VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
INFORMATION TECHNOLOGIES STUDY PROGRAM

Problem-Based Project

Lost Animal Platform

Done by:
Edvinas Gerdvila
Renaldas Narbutas
Arsenij Nikulin

Supervisor:
dr. Agne Brilingaitė

Vilnius
2025

Contents

Abstract **4**

Santrauka **5**

Introduction **6**

1 Related Works **8**

2 Analysis **8**
2.1 Similar systems analysis 8
2.2 Color histograms 9
2.3 Texture histograms 10
2.4 Edge detection 11
2.5 Structural similarity index measure 11
2.6 Machine learning 11

3 Design **12**
3.1 Use cases 12
3.2 System requirements 13
3.2.1 Functional requirements 13
3.2.2 Non-functional requirements 14
3.2.3 Software requirements 15
3.3 Architecture 15
3.4 Database 17
3.5 Sitemap 18
3.6 Managing Pets 19
3.7 Lost and found pet reports 20
3.7.1 Report structure 20
3.7.2 Operations on reports 21
3.7.3 Searching for reports 21

4 Implementation **22**
4.1 Tools 22
4.2 Deployment 23
4.3 Automated installation 23
4.4 Database 25
4.5 Authentication 26
4.6 Multi-language support 27
4.7 Image 27
4.7.1 Image processing 27
4.7.2 Image comparison 27
4.7.3 Color histogram 28
4.7.4 Texture histogram 29
4.7.5 Histogram comparison 30
4.8 Email and poster generation 30

Conclusions and Future Work **31**

References **32**

Appendices **34**

Abstract

This paper outlines the development and implementation of StraySafe, a web-based lost animal platform. There have been many similar platforms developed but not many with image comparison algorithms and community based crowd sourcing functionalities. We integrate HCI principles to make the website user friendly and rewarding to use. Images are compared using color and texture histograms. The distribution of colors are separated into bins and compared using Battacharyya comparison algorithm. Texture histograms are calculated into bins using LBP pattern algorithm and compared using Chi-squared technique. There is a list of shelters that you can look up on a map. Each report has a comment section where users can talk, which can be translated to the preferred language (Lithuanian or English). When users generate reports, they get an email with a poster which they can print and post around the neighbourhood. Users are categorized into three groups: default users, moderators, and shelters, each with unique functionalities. This structure aims to create a supportive and effective community for reuniting lost pets with their owners.

Santrauka

Darbo pavadinimas kita kalba

Šioje dokumentacijoje aprašoma kūrimas ir įgyvendinimas "StraySafe" internetines platformos, skirtos pamestiems gyvūnams surasti. Tačiau jau yra sukurta daugybė panašių platformų, nedaugelyje jų naudojami vaizdų palyginimo algoritmai ir bendruomenės pagrindu veikiantys funkcionalumai. Siekdami padaryti svetainę patogią ir malonią naudoti, taikome HCI principus. Vaizdai lyginami naudojant spalvų ir tekstūrų histogramas. Spalvų pasiskirstymas skaidomas į dėžutes ir lyginamas naudojant Battacharyya algoritmo metodą. Tekstūrų histogramos apskaičiuojamos į dėžutes naudojant LBP algoritmu ir lyginamos naudojant Chi-Squared techniką. Platformoje yra prieglaudų sąrašas, kurį galima peržiūrėti žemėlapyje. Kiekvienoje ataskaitoje yra komentarų skiltis, kurioje naudotojai gali bendrauti, o komentarai gali būti išversti į pageidaujamą kalbą (lietuvių arba anglų). Naudotojams sukūrus skelbimus, jie gauna el. laišką su plakatu, kurį gali atsispausdinti ir paskelbti apylinkėje. Naudotojai skirstomi į tris grupes: paprastus naudotojus, moderatorius ir prieglaudas. Kiekviena turi unikalių funkcionalumų. Ši struktūra siekia sukurti palaikančią ir efektyvią bendruomenę, padedančią susigrąžinti pamestus augintinius jų savininkams.

Introduction

Searching for lost pets is an important and challenging task that many pet owners face. When a pet is lost, it becomes a real stress and source of anxiety for the owners. Losing a pet is not only an emotional trauma, but also a great responsibility as the owner feels an obligation to bring the pet home as soon as possible.

When a pet goes lost, owners take all sorts of steps to find them. The first step is often to post notices in the neighbourhood. These reports contain a description of the pet, a photo, the owner's contact details and sometimes the promise of a reward. The posters are placed on poles, public transport stops, shops and other places where there is a large number of passers-by. Some owners also post flyers in mailboxes in the hope that a neighbour has seen their pet.

In parallel, owners contact specialised services and organisations, such as local animal shelters, veterinary clinics and animal trapping services to see if any animals matching their pet's description have been reported.

The motivation for creating this platform is to reduce stress for those whose pets have been lost. Our goal is to create a universal platform where the user can quickly and clearly obtain the information he needs from other users, shelters, veterinary clinics, get advice and help, and all this without leaving the site!

Currently, there are many existing web platforms designed to assist in the search for lost pets. These platforms serve as convenient tools for distributing and presenting information to users. However, they often require users to visit multiple sites and resources, which can be time-consuming and overwhelming during an already stressful time. In this paper, we want to describe and demonstrate a new platform that should speed up and automate the search process. By simplifying and enhancing the search process and centralizing information, we hope to make it easier and faster for pet owners to find their lost pets.

To achieve these goals, we have implemented advanced image comparison techniques using color histograms [11] and texture histograms [17]. These algorithms compare pictures of animals from reports that users create to announce the lost or found pet. By analyzing the color and texture patterns in images, our platform can match similar-looking animals, increasing the chances of reuniting pets with their owners. This method enhances the search accuracy and reduces the time spent manually sifting through numerous reports.

Additionally, we have incorporated several user-friendly search options. Users can search by uploading an image of their pet, allowing the platform to find potential matches using our image comparison algorithm. Two more ways to search is grid view list and map.

Our platform aims to bring together a community of pet owners, shelters, veterinary clinics, and volunteers, creating a collaborative network dedicated to finding lost pets.

This paper outlines the thought process, design and implementation of our lost animal platform. The next two sections describe analysis of both existing web-based platforms for solving the missing pet problem, as well as analysis of algorithms and other approaches such as ML. In the design segment the general ideas and structure of the platform is described, while implementation section presents our specific realisation of the proposed design.

This paper outlines the thought process, design and implementation of our lost animal platform. Section 1 provides the insights into the background. Section 2.1 explains the results of previous attempts of tackling a similar problem that our product strives to solve. Sections 2.2 through 2.6 describes the various variables that make certain algorithms for image analysis better fitting for our platform. Sections 3.1 displays the use cases that fit default users, moderators, shelters and

anonymous users. Section 3.2 thoroughly outlines the functional and non-functional requirements for our platform. Section 3.3 and 3.4 dwells more deeply into the architecture of the StraySafe platform, design of the database and provide visual reference such as architecture, ER diagrams. The 3.5 section provides a thorough explanation of navigation on the StraySafe platform with visual reference in the form of a flow diagram. Section 3.6 talks about the main object of the lost pet platform which is the pet object, the section thoroughly explains the choices made when designing the pet object and operations that are possible with the pet object. Section 3.7.3 thoroughly explains every nook and cranny of the report object, the purpose, possible operations and how it is used for other parts of the StraySafe platform. Sections 4.1 displays the tools used for the implementation of the lost pet platform. The 4.2 section explains the purpose behind the setup of our system, what each machine is used for, what logic is performed on which machine. Section 4.3 thoroughly defines the process behind automated deployment how each virtual machine is set up what tools are used how the tools are configured to fit the StraySafe platform. The 4.4 section thoroughly displays a relational schema of the lost pet platform database and the logic of setting up the structure. Section 4.5 provides insights of how the authentication flow is implemented in the StraySafe platform explains the communication of frontend and backend servers during the authentication process. Section 4.6 thoroughly describes the process of multi-language implementation, how separate parts of the lost pet platform use different implementations of multi-language support. Section 4.7 thoroughly describes the systems implementation for image processing, image comparison, implementation of various histograms and their analysis. The 4.8 explains the implementation of an email service used to notify users of successful events that occur on the StraySafe platform and it also explains the practices used for generating a PDF format poster for the user of the StraySafe platform.

1 Related Works

There have been many attempts to create an application for finding lost pets in communities far larger than Vilnius [23]. In a big city the key features that people love to have include pet tracking, health tracking and nearby veterinaries. As sweet as these tracking features sound, people are hesitant to chip their pets with GPS or even the mandatory microchip for cost and privacy concerns [24]. It has been said that over 35% of pets in the UK do not have mandatory microchips and the existing ones usually fail to update the microchip record [18]. This leads to pet shelters taking care of stray pets without being able to find the owners.

Our approach relies more heavily on the community instead of sole individuals that has bought expensive tracking gear [23]. It has been said that globally in the world 70% of people have phones, including developing countries [21]. Since most people carry a smartphone in the world, it does not require much to take a photo of a stray pet and upload it.

Another aspect of the platform is image comparison. While there are many studies coming out with new AI models that can differentiate pets, there are not many that do so with traditional algorithms [13]. We have had to come up with our own solutions on that end. This include various histograms that is written about in the book Histograms and Image Statistics [1]. We have also looked edge detection algorithms[19], machine learning approaches[25] and similarity indexes.

GPS on the collar is also one of the ways people can track where their pets are [2]. The system would show where the pet is any time and when he strays away from the region. It can send sms or emails with google map location specified as well as track the path the pet took. Although this is promising, people would need to buy very specific collars and would need to integrate with the system which can complicate things for the users.

There has also been work on crowd sourcing apps that focuses on IoT and finding items[5, 8]. This effect could be very powerful by using a lot of people and interconnecting them in various ways.

There has also been progress in pet biometrics, focusing on methods such as capturing images of an animal's nose to identify unique patterns [9, 4]. However, these technologies are still in the developmental stages and face implementation challenges. Moreover, the reliability of these methods is limited, as they often rely on machine learning algorithms, which fundamentally make probabilistic predictions.

2 Analysis

2.1 Similar systems analysis

There are several online services for finding lost pets, including **PetRadar**[16], **Tagmefy**[22], **PetcoLove**[15], and **PawBoost**[14] that illustrate the basic mechanisms of such systems. Typically, these platforms allow users to create detailed reports of lost and found pets, use local alerts, and utilise social media integration to increase the visibility of lost pets. Some features of these services, such as alert systems and social media integration, will be included in our project.

In some regions, Facebook groups are the most popular and active way to search for lost pets, for example in Lithuania there is a facebook group for lost items[12]. This group demonstrates the importance of community involvement in the pet search process, but lacks the structured approach and comprehensive capabilities of specialised platforms.

However, many of these existing services are designed to search for various missing items, and pet searches are only a small part of their functionality. In addition, some of these platforms offer paid features, such as adverts and prioritised social media posts, which may not be available to all users. There are also limitations in terms of reach, as not all services operate globally, limiting their effectiveness in some regions. Our project aims to address these shortcomings by creating a specialised, comprehensive and accessible platform specifically designed to reunite lost pets with their owners.

2.2 Color histograms

Color histograms provide an approach to summarize the color distribution of an image, which helps in distinguishing and comparing the visual characteristics of different pets.

A color histogram represents the frequency of occurrence of various color values in an image [11]. For the purpose of our platform, we generate histograms for the RGB (Red, Green, Blue) color model. Each image uploaded to the platform is processed to create a three-dimensional histogram. This process involves:

- Convert the image from its original color space to the HSV color space to effectively separate color information (hue and saturation) from lighting (value), thus reducing variations due to different lighting conditions.
- Divide each HSV channel (Hue, Saturation, Value) into a predefined number of bins to capture the frequency distribution of each component, enhancing the ability to differentiate based on color and brightness characteristics.

To determine the similarity between two images, we employ the Bhattacharyya distance, which measures the overlap between two statistical samples. A smaller Bhattacharyya distance indicates a higher degree of similarity between the two histograms.

A visual representation can be seen in Figure 1

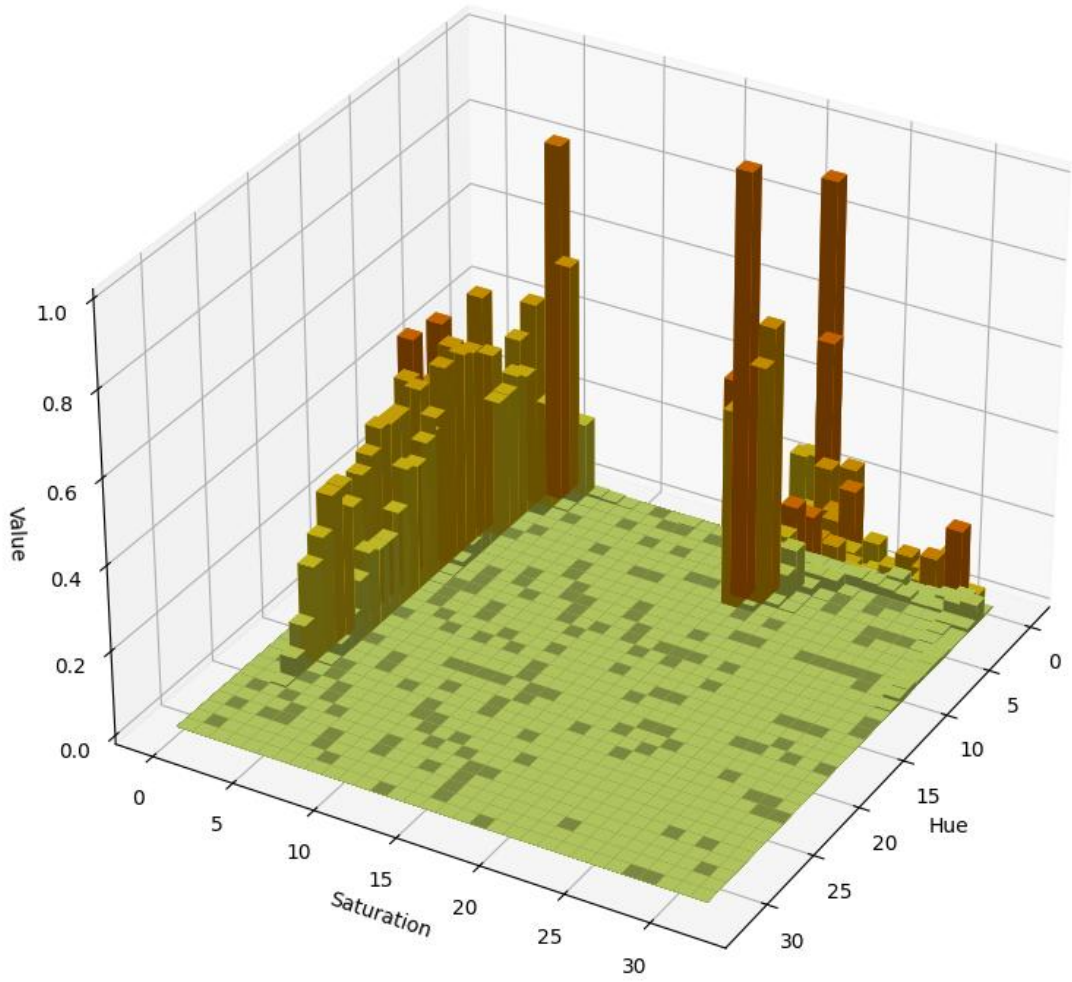

Figure 1. Visualization of color histogram in 3D space

2.3 Texture histograms

It matches images based on textural features, such as fur patterns and unique markings. Texture histograms summarize the distribution of local texture patterns in an image.

In our platform, texture histograms are utilized to represent the local binary patterns observed in an image, categorizing these patterns into various bins for histogram comparison using Chi-squared statistical test [17].

- Converting the image to grayscale to emphasize the texture information without the influence of color.
- Calculating the Local Binary Pattern (LBP), which involves comparing each pixel in the image to its surrounding neighbors to form a binary number based on whether the neighboring pixels are brighter or darker than the center pixel.
- Creating a histogram of the LBP values which represents the frequency of each binary pattern within the image.

From the extracted binary patterns, we construct texture histograms. For comparing the histograms we use the Chi-square comparison statistical test. A lower Chi-square value indicates a greater similarity between the textures of two images.

2.4 Edge detection

It identifies pets by their structural outlines, such as shapes of ears, tails, and fur patterns. This method focuses on detecting boundaries and edges in the images of pets. By utilizing the Canny edge detection algorithm [19] we can capture edges with precision. The process involves several steps:

- Applying a Gaussian blur to the image to reduce noise and detail, focusing on significant transitions.
- Calculating the gradient of the image intensity at each pixel, determining the direction and intensity of edges.
- Applying non-maximum suppression to thin out the edges, ensuring that only the most pronounced edges are detected.
- Using double thresholding to distinguish between strong and weak edges, and linking them to detect complete edges.

After processing with the Canny edge detector, the result is an edge map that highlights the boundaries and outlines of objects within the image. Each detected edge is represented by white lines on a black background, clearly delineating the structural features of the pet.

The method proved to be unreliable for our purposes because of different angles and positions of photos.

2.5 Structural similarity index measure

The SSIM[26] is a method for measuring the similarity between two images, which is particularly effective in assessing the quality of visual structures between the images of pets. SSIM is designed to improve on traditional methods like mean squared error, which may not correlate well with visual perception. It considers changes in texture, luminance, and contrast.

The SSIM index provides a measure from -1 to 1, where 1 indicates perfect similarity. This method allows to detect high levels of visual similarity in terms of brightness, contrast, and structure.

Unfortunately this method proved to be too rigid for our platform. A person may upload an image from the side, front or any other angle and this algorithm would not work well.

2.6 Machine learning

For our platform Convolutional Neural Networks (CNNs) [25, 10] offers great features and could do many tasks related to image analysis, comparison challenges. These networks are particularly well-suited for image classification, object detection, and feature extraction tasks.

- **Image Classification:** the network can be trained to classify images of pets, discerning between different species and breeds. This could be used for filling in the report automatically. A bonus point is that the user would not need to think deeply on what breed the pet is. It could also filter out bad actors.

- **Object Detection:** CNNs also excel in detecting specific objects within an image. In our platform this would be great for automatic cropping of the image to remove all the background and make comparison algorithms more accurate.
- **Feature Extraction:** it can tell apart different features such as color patterns, fur textures, and unique markings, creating a detailed profile for each pet. These profiles could then be used to match lost pets with found or seen reports more accurately.

Unfortunately, we could not make the approach viable due to lack of resources and knowledge on the subject.

3 Design

3.1 Use cases

The lost animal platform may be used for the following purposes: Searching for lost pets, owner of pets, spreading the word, volunteering and support or socialization.

Users can use the platform to search for their missing pets by browsing through reports of lost animals, filtering based on various attributes, and getting notifications if a similar report is near to them. People (or shelters) who have found lost pets can use the platform to search for the owners by creating reports with descriptions and images of the found animals. The web platform also offers several options for spreading information related to pets, such as publishing posts on social networks, automatically generating posts, and notifying nearby users. Volunteers who are not directly involved with lost or found pets can freely join in the search for lost pets to increase the chance of finding them. The platform also provides support contacts, tips related to this topic and possibility to write comments below the posts.

To achieve these goals, but also to maintain order on the platform, a system was developed that demonstrates the rights and capabilities of the users (Figure 2).

There are 4 actors in the diagram. They represent the roles on the website.

1. **Anonymous**: an unauthorised user. He has access only to view content: read articles related to the topic, view lists and maps of lost and found pet reports, and view details of the report, which includes the pet's attributes, comments, and contact details of the report's author. This functionality will help this role not to be able to modify any content on the site, but it will allow access to all the necessary information.

2. **Default user**: inherits all the functionality of anonymous, as well as includes CRUD operations with reports. In addition, this role can post and delete comments under other reports. The most important difference from the previous role is the presence of a profile and the ability to customise it (change nickname, first and last name, etc.).

3. **Shelter**: a modification of the default user, including all its functionality. The peculiarity of this role is that it has a shelter icon in place of its address on the map. This is done because shelters often have quite a large number of pets, so all reports created by the shelter are united under one icon on the site for the convenience of users and to prevent their disentanglement.

4. **Moderator**: is a role that is intended for content moderation. This role can delete incorrect comments, and block reports and users. Also, the user with this role inherits all the functionality of the default user.

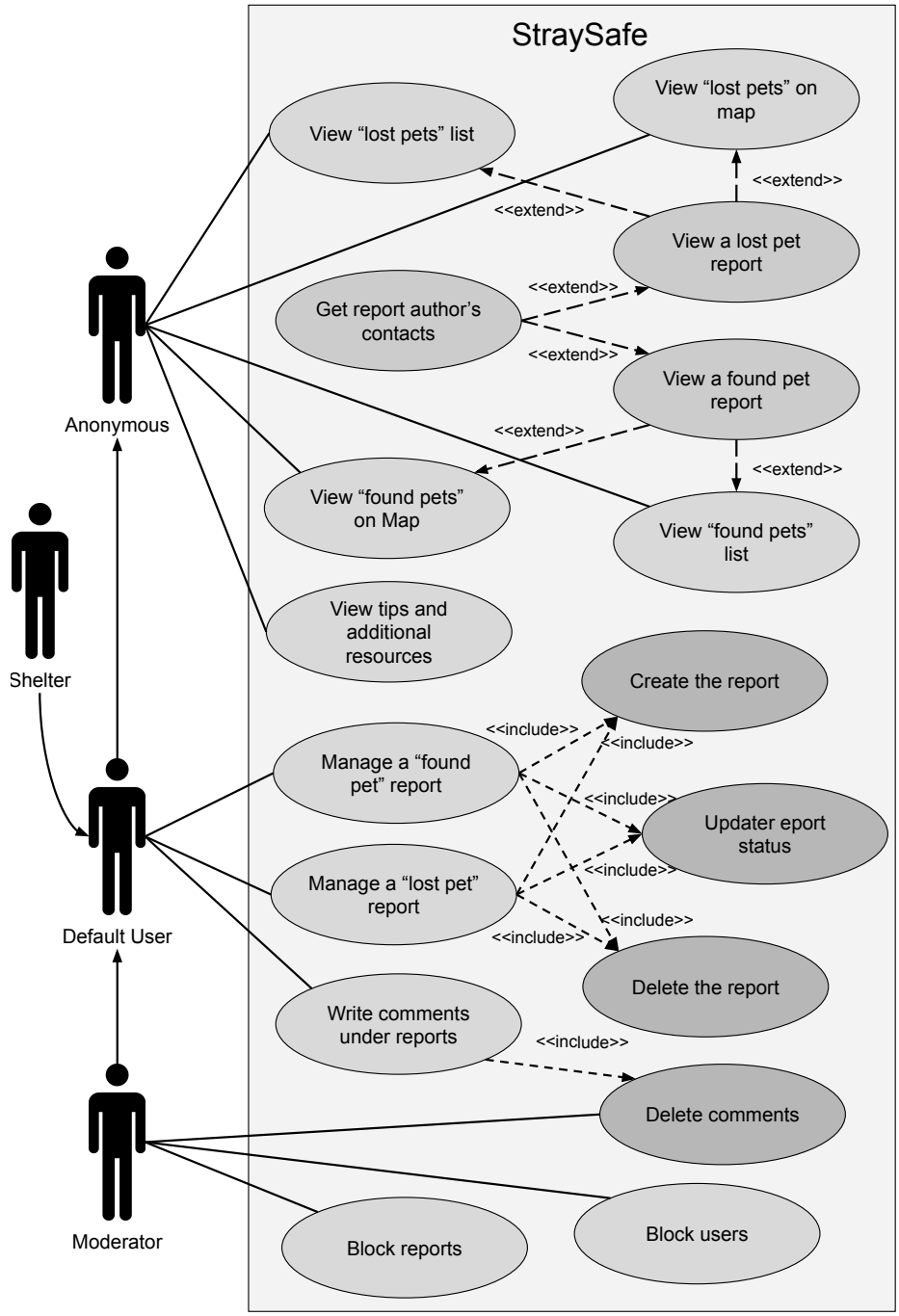

Figure 2. Use Case Diagram

3.2 System requirements

3.2.1 Functional requirements

We formulated 24 requirements for the system. These requirements are divided into functional, non-functional, and software requirements. Functional requirements are divided into deployment and hosting (1-3), user interaction and account management (4-5), report creation and management (6-7), comments and email notifications (8-10), search and algorithms (11-14), user experience (15-16), and tips and moderation (17-18). Non-functional requirements address the performance, security, and documentation of the system. The software requirements specify compatibility with browsers based on Angular 17.

1. The system should be deployed on the VU MIF Cloud.
2. Automated installation using Ansible playbooks.
3. The web application will be hosted on a virtual machine with the Apache2 web server.
4. The account system should allow users to register and log in, and view other users' profiles, pets, and reports.
5. In the account settings, you should be able to change your email, user password, and notifications.
6. Users who lost or found a pet must be able to create a report and input necessary data: the image of the pet, attributes based on keywords, and general location.
7. The User should be able to update the status of the report or delete it he has created.
8. Each report shall have a comment section where a user can post comments.
9. The comments shall be filtered using a keyword-based analysis.
10. Users will get email notifications when registering or uploading a lost pet report.
11. There shall be two views: map and grid list.
12. Both views need to be able to filter out information that the user is interested in.
13. The map view shall show spots where lost pets often get found.
14. Image comparison to find pet similarities.
15. The website shall support Lithuanian and English languages to cover a diverse user base. Users should have the option to select their preferred language.
16. The interface shall be designed for various devices, including desktops, tablets, and mobile phones.
17. Have pages on how to handle having lost a pet (e.g., "How to find a lost pet") to ensure accuracy and reliability.
18. There will be a moderator system where they can delete bad actor reports.

3.2.2 Non-functional requirements

1. Every layer has its own README.md file where there is documentation of this segment.
2. Communication between devices communicates using REST APIs, and HTTP protocol.
3. Pages should load in up to 5 sec.
4. The database should follow the first three normalization forms for designing the database.
5. Security Measures such as spam protection, DDoS attacks, data leaks, Protection from CSRF and XSS attacks, and SQL injections.

3.2.3 Software requirements

Based on the angular 17 version, the platform should work on these browsers from the specified versions and higher shown in Figure 3.

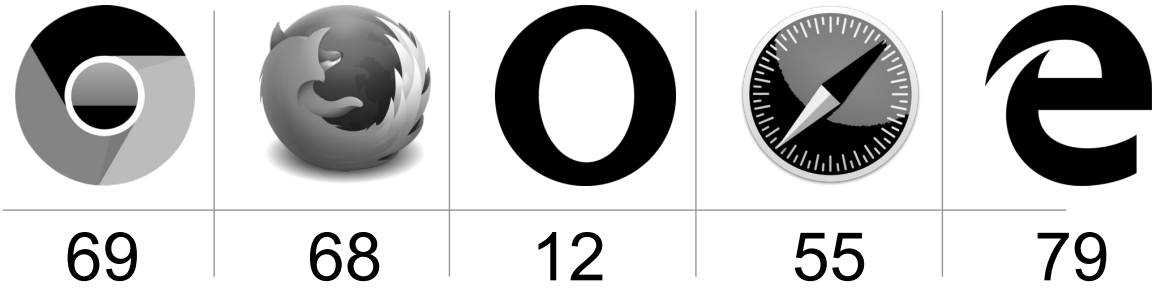

Figure 3. Browser Compatibility Chart

3.3 Architecture

The system is built upon a layered architecture designed for scalability, security, and efficiency. It can be seen in Figure 4. It comprises of five different layers communicating with defined interfaces and have their own specific responsibilities. Web layer is the entry point for the website and Translation layer is an on-demand layer. The main logic communicates through the business layer. This design principle ensures a clean separation of concerns.

- **Web Layer:** Utilizes the Angular JavaScript framework, using its server-side rendering ensures that web pages are pre-processed on the server, allowing for faster loading times and improved user experience.
- **Business Layer:** Acts as the system's logic center, processing incoming requests from the presentation layer. Tasks such as image analysis and language translation are handled here.
- **Analysis Layer:** This layer serves as an intermediary for data access and management. When the business layer requires reading or writing from the database, such as user authentication or fetching reports on lost and found pets, it delegates these operations to the persistence layer, which in turn communicates with the database.
- **Database layer:** physically separated from Tier 1 in order to optimize data security. This separation enhances security measures by isolating data. More can be seen in figure 5.
- **Translation layer:** Hosted on separate device because it demands a lot of resources which cannot fit in our configuration. It is separate from the whole other system. It translates dynamic content such as comments, notes, descriptions.

Web, business and analysis layers are clustered into one machine for ease of development and deployment. However, this configuration does not impose constraints on scalability or distribution. Given that each layer has been designed with distinct responsibilities and communicates through well-defined interfaces, separating them onto different machines would be a straightforward task.

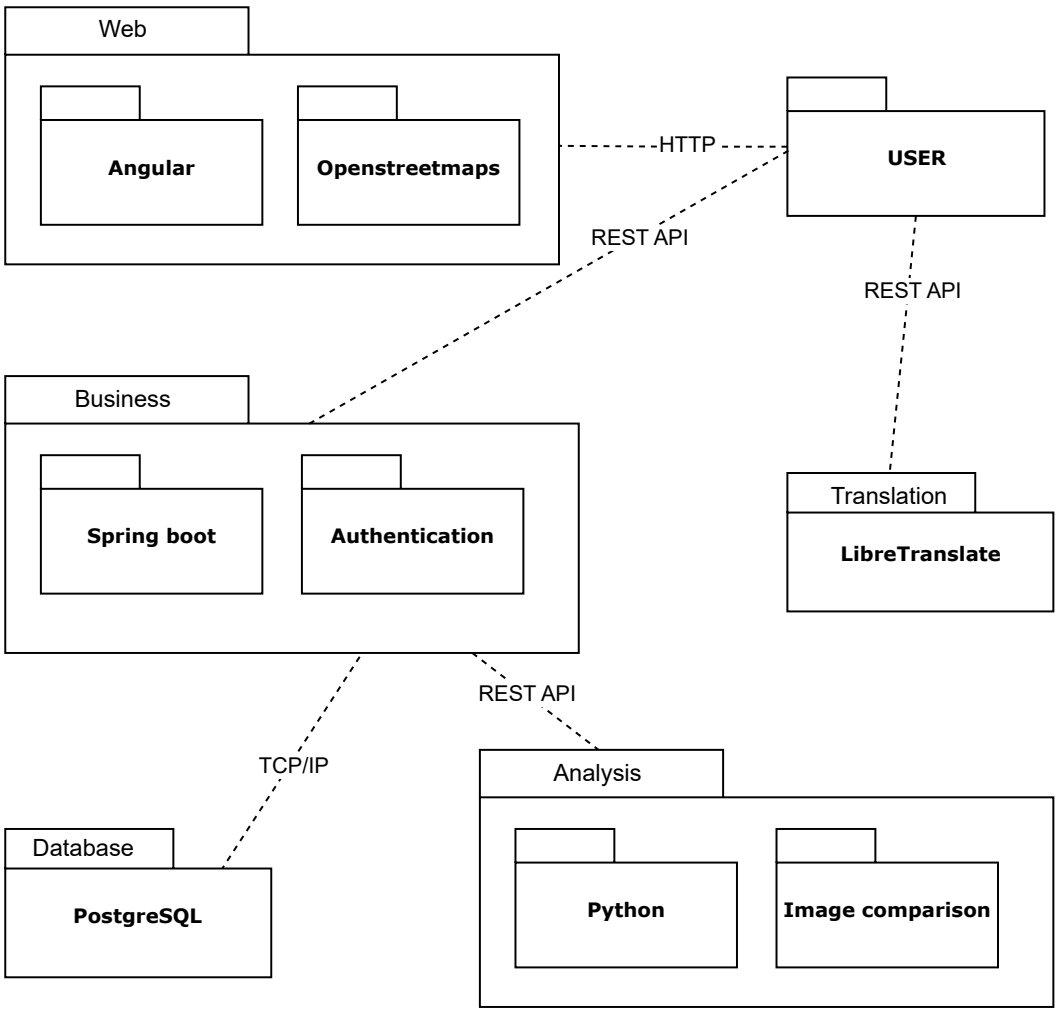

Figure 4. StraySafe Architecture

3.4 Database

To logically store all system data such as users, pet information, reports, comments, etc. an Entity Relationship model was developed. The description below describes the ER model presented in 5.

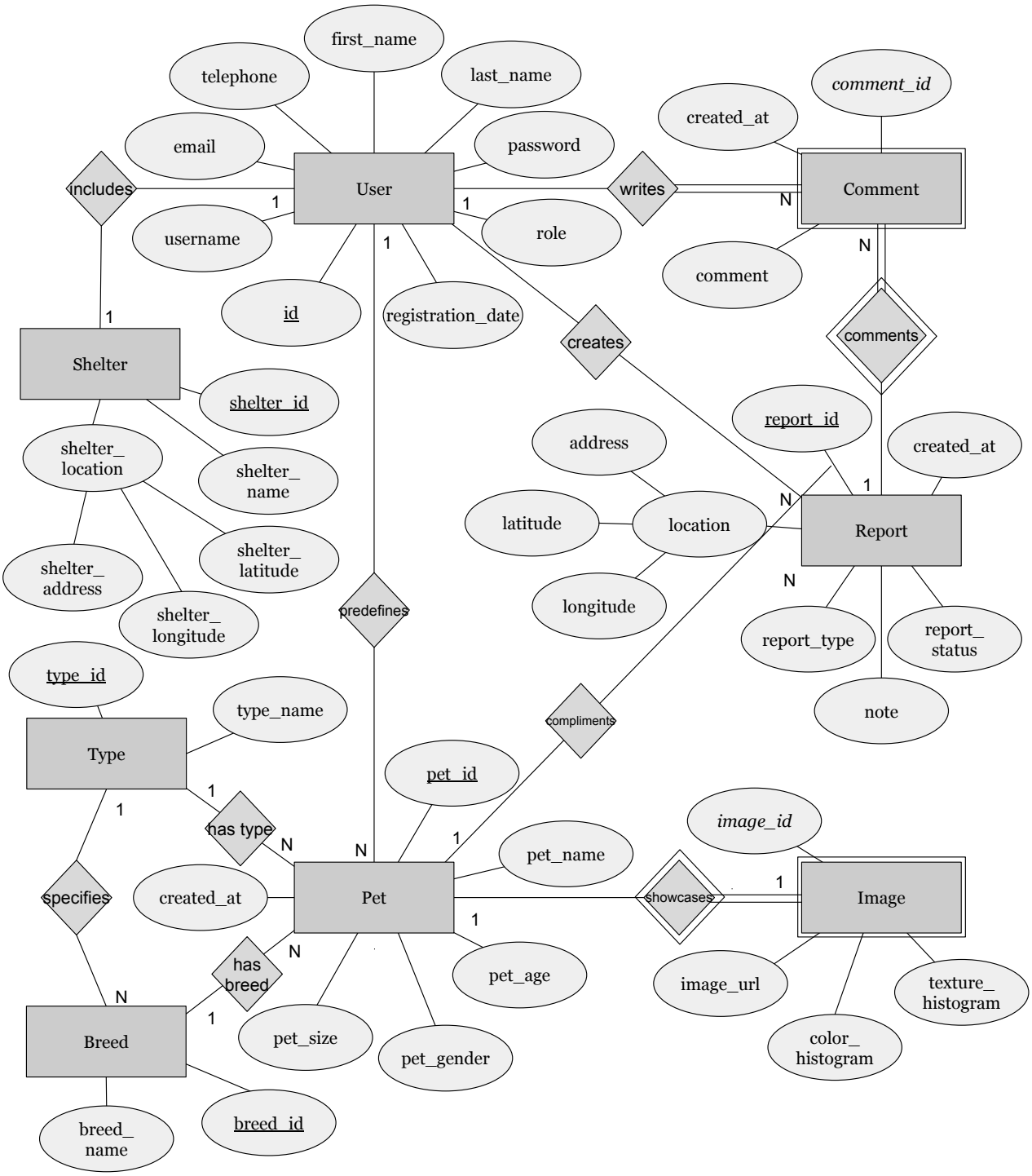

Figure 5. Entity Relationship Model

User is by far the most important entity in this model, as only authenticated users have access to create new reports. It has the following list of attributes: id (primary key), username, password, first and last name, email and telephone, registration_date, and role. User has a one-to-many relationship with the **Pet** entity, which means that a single user can specify multiple pets. The **Pet** entity has the following attributes: pet_id (primary key), pet_name, pet_size, pet_gender and

pet_age. The **Pet** entity also relates important entities such as **Type** and **Breed** in the many-to-one cordiality, meaning that one pet is associated with only one type and breed. These two entities only have keys (type_id and breed_id) and names (type_name and breed_name). In addition, **Type** has a one-to-many relationship with **Breed** because each breed refers to a different type of animal. One more relationship with **Pet** has **Image** weak entity. It has image_id as a foreign key and 3 more attributes - image_url for image path storing, texture_histogram, and color_histogram for the analysed histograms storing.

Back to **User**, this entity has a one-to-many relationship with the Report entity named "creates", since a report can have only one author. In turn, the report describes a lost or found pet, for which there is a many-to-one relationship with the **Pet** entity, for example, the same pet can be lost several times. The **Report** entity contains the following attributes: report_id (primary key), report_type, report_status, created_at, note, and a combined location attribute consisting of address, latitude, and longitude.

The weak entity, which relates both **User** and **Report** in this model is **Comment**. It is related to **Report** by a many-to-one relationship, and also each comment has an author from the **User** entity. The comment entity has a comment_id, a publication date as created_at, and the comment itself.

A separate entity **Shelter** was designed to simplify the creation of shelter reports. It contains the following attributes: shelter_id (primary key), shelter_name and a combined shelter_location attribute consisting of shelter_address, shelter_latitude and shelter_longitude. This will allow shelters to avoid specifying locations when creating reports, as well as marking them separately on the map.

3.5 Sitemap

Sitemap, presented in Figure 6, represents the hierarchy of the platform pages and sections with different content.

The start of this diagram is **Home** page (dashed in the diagram) - the main page where placed inside 4 components: Create Report, Search, Statistic, Tips and Resources. The **Login** page also goes from the Home page, but using a special button. After successful login user should be able to view his profile in **Profile** page, view his reports in **Reports** page, and change settings in **Settings** page.

Create Report component suggests for the user 2 buttons "I Lost a Pet" and "I Saw a Pet". The report creation mechanism is similar for both types, so the process, which is divided into 2 steps, is created using the same components. The first step is on **Input Address** page, where the user provides the location, where he last time saw the pet. The second step is **Report Summary** page, where the user should select a pet from the pet list, add notes, and submit the report. In case if pet does not exist in the list, the user should go to **New Pet** page to create a new pet. Then when the user creates a new pet he can select it in **Report Summary** page.

Search component have inside 3 different ways to search among existing reports. The first variant is on the **Map** page, where the user can comfortably look at report locations. The second way is **Report List** page, where reports are placed in a grid view list, and the user can narrow his search using filters. The third variant is **Find by Image** page, where the user should be able to input an image and get several reports, where the pet image is similar.

Tips and Resources component also link to the 3 pages: **Lost Pet Tips**, **Found Pet Tips**, **Volunteering information**. This page content advises on different types of users to support them and increase the chances of a successful search for a pet.

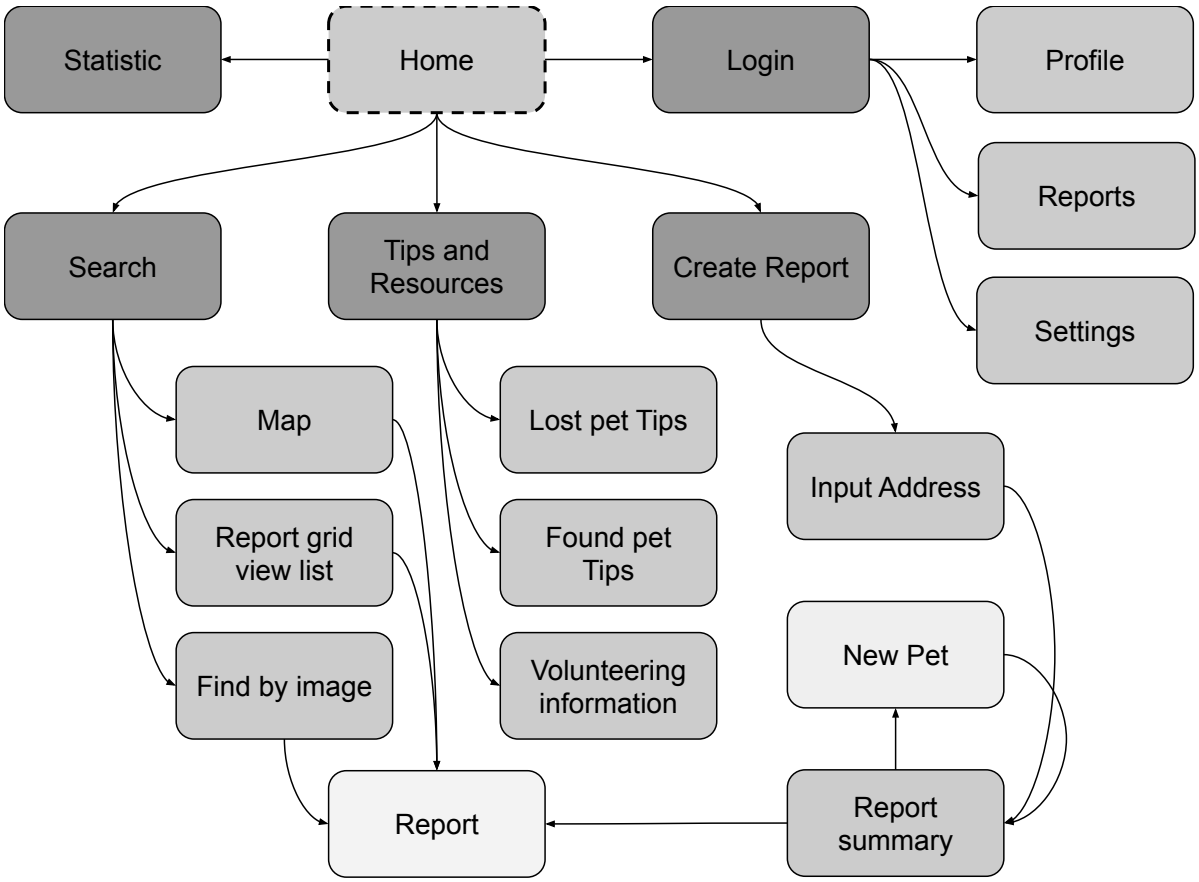

Figure 6. Sitemap

Statistic component shows the performance of the platform: the number of registered users, shelters, created reports, and reunited percentage.

The last element in the site map is **Report**. It has all the information like location, image, attributes, status, author data, etc. Also, there is a discussion section, where users can write comments. This pop-up element can be reached from any search page as well as when the report was created on the **Report Summary** page.

3.6 Managing Pets

The pet object is the main one for the platform, so it was important to structure it correctly and not overload it with unnecessary elements. We have selected seven attributes to include: pet image and pet type (both required), pet name, age, size, gender, and breed . The last 5 elements also required but can be selected as *Unknown*. Pet attributes are predefined therefore user input error whilst creating a pet is reduced. While additional attributes like fur color or collar color could be useful, they were excluded to simplify the creation process and avoid potential user confusion. Users can specify such detailed information in report notes if needed.

Each user has a public list of their created pets displayed in their profile. A pet object is necessary for creating new reports and can be referenced in multiple reports. While users cannot update the information of an existing pet, they have the option to remove it from their list.

As we could not find a standard for breeds we have found different websites that contain comprehensive lists of breeds which can be included in our platform. For dog breeds dogbreedinfo.com[6] was chosen, and for cat breeds catbreedlist.com[3] was chosen.

3.7 Lost and found pet reports

The main element of this platform is the report. By creating reports, users share data that directly contributes to the search for missing pets and their owners. This means that this element must be precisely constructed.

3.7.1 Report structure

The ER diagram 5 report entity will help you visually understand what is described below.

1. **Location**: has data such as address, latitude and longitude. This is needed so that other users can quickly understand where the event occurred.

2. **Pet selection**: Each report describes a pet that has been seen or lost. The user must select one animal from the list of their pets.

3. **Report type**: has 3 possible options: lost, seen and found. They give an understanding of their purpose, what kind of problem they are solving.

4. **Report Status**: has this options - *Active, Non-Active, Found, Deleted, Blocked*. When a report has been created, it is assigned status *Active*. Only reports with this status are visible to users when searching. After two weeks. the report automatically changes its status form *Active* to *Non-Active*. If the user indicates in the report that his pet is found or he found the owner of the lost pet, then it will accept status *Found*. Also, user able to delete report, then the status will be changed to *Deleted*. If moderation found bad content in the report, then it can block the report by changing its status to *Blocked*. In report status state diagram 7 shows changes in report statuses depending on events.

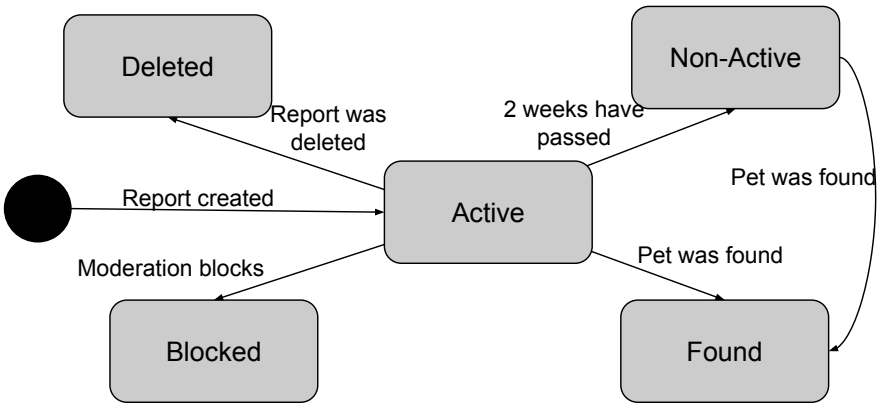

Figure 7. Report Status States

   Report status helps in two ways at once: to show other users only relevant reports and to compile statistics for the platform.

5. **Note**: the user is able to add additional notes to the report in free text. The user can provide additional details not seen in the attributes or a reward that the person who finds the pet will get.

6. **Date of publication**: this information gives other users an idea of how long ago a pet was lost or spotted. Also, reports that have been posted for more than two weeks automatically change their status to *Non-Active*.

3.7.2 Operations on reports

As there is a lot of data that needs to be entered to create a report, according to the HCI principles the process of creating a report have 4 user input steps described below. It is also important that only authorised users can create reports, as the report must have author details, which are specified in the user's profile. The system checks if the user is authorised. If not, it asks the user to log in. Report creation process shown in flowchart (Figure 8).

1. **Report type**: To proceed to report creation, the user must select a report type (Figure 15). For any type of report, the creation procedure will be the same.
2. **Location**: the user specifies the location where he or she last saw the pet (Figure 17).
3. **Pet selection**: The user must select one animal from the list of his pets. A Pet ID will be assigned to the report. If there is no pet in the list, a new pet should be created.
4. **Agreement**: Before publishing the report, the user must agree to the platform rules. You can also add notes to the report at this stage.

Then, when all the data has been entered, the submit process forming the reporting data, then data writes into database.

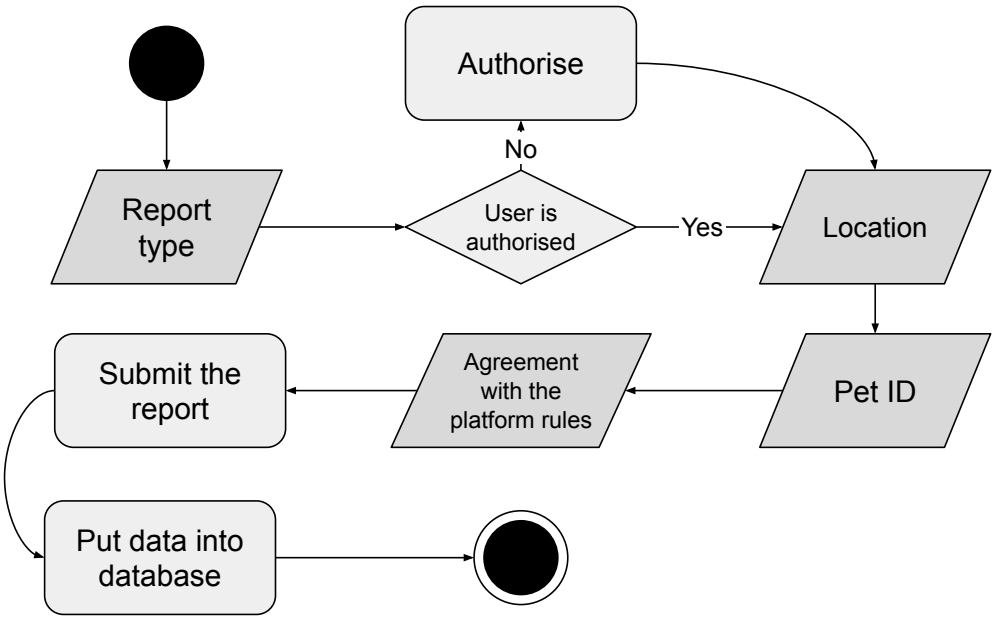

Figure 8. Report Creation Flow

In addition to creating a report authorised users have the opportunity to update report status and delete report (for report author only), block reports (for moderation only), post and delete comments under the reports.

3.7.3 Searching for reports

Users can only search among reports, where `report_status` equals to *ACTIVE*. To help the user find the appreciate search method, a general search page have been designed (Figure 27).

The search option where reports are located in grid view allows users to view each report in its own box, as depicted in Figure 23. By clicking on a report box, users can access all the information specified in it.

Each report box is a container that displays very general information about the report and the pet, including the pet image, report type, pet type, and address. Users can apply various filters to discard reports they are not interested in. The available filter options include report type (lost, seen, found), search distance (where the user specifies a center point address and a distance radius to show only reports within this radius), publication date (allowing the user to specify reports from a certain date onward), animal type (dog, cat, other), breed (where the user can specify one specific breed), gender (male, female, other), and user nickname (to see active reports created by a specific user).

To simplify and speed up the process of searching for lost and found pet reports, visualization through a map has been chosen. Markers have been placed on the map where other users have indicated the location when creating a report. There are four different types of markers on the map: three markers for different types of reports (lost, seen, and found) and one marker for shelters, as depicted in Figure 32. When you click on a marker, all information about that particular report is shown, or if you click on a shelter icon, it will open the personal page of that shelter. If there are a lot of markers in one place, they are combined into a cluster that shows how many reports are in that area, as seen in Figure 33. To discard unnecessary reports, simple filters are available and divided into three categories: pet type (dog, cat, other), report type (lost, found, and seen), and shelter. These items from each category can be selected at the same time, allowing users to focus on relevant reports (for example, if you have lost a dog, you can filter by dog, seen, and found reports).

The "Search by image" option allows the user to find reports, photos of pets that are relatively similar using a pet photo and location. Texture and color histograms are used for analysis. As a result, the user receives 8 reports that are most similar. This option can also be used by an unauthorised user.

4 Implementation

4.1 Tools

- **Apache2** (Web server): Widely-used, open-source web server known for its robustness, flexibility, and security features.
- **Angular** (Front-end): Popular front-end framework developed by Google. Angular was chosen for its powerful features, comprehensive tooling, and strong community support.
- **Tailwind CSS** (Styles): CSS framework that provides low-level utility classes to build custom designs directly in your markup. It was chosen for its flexibility, ease of use, and ability to rapidly prototype and style applications without leaving HTML and creating .css style files.
- **Python** (Image comparison): This programming language has many libraries, which simplify work for comparing images.
- **Spring boot** (Business Logic): Spring Boot is an open-source Java framework used to create stand-alone, production-grade Spring applications.

- **PostgreSQL** (Database): Open-source relational database system known for its stability, extensibility, and standards compliance.
- **OpenStreetMaps (OSM)** (API): Free and open source map. We use the leaflet dependency which allows easy implementation.
- **Nomanatim geocoder** (API): Use OpenStreetMaps data to convert addresses into geographic coordinates and vice versa.
- **OpenCV** (Library): Library for image comparison and processing algorithms.
- **LibreTranslate** (Library): Open-source machine translation that supports multiple languages. It can be self hosted. It was chosen for dynamic content (comments, notes, descriptions) translation.
- **Ansible** (Deployment): Automated scripts to setup and update the platform.

4.2 Deployment

The deployment diagram, shown in Figure 9, illustrates the setup involving four machines, including the end user.

UserClient - The end user that will be using our website, this can be an android, desktop, IOS or any other device that uses a browser. Web server - The main logic of the application, it has three programs running as systemd services: Apache, Java, Python. Apache is the entry point of the application and serves HTML, css, java script that angular generates when building. Java handles the back-end logic of StraySafe and runs the Spring boot service. Python integrates with the java back-end to provide image comparison. Database server - All of our data storage on one device. Images are stored as URL paths to the image. Translation server - A separate machine to translate report notes and comments on demand. It does not save these translations anywhere.

4.3 Automated installation

Ansible was used to automate deployment (CD). The process works using one central device (ansible machine) and three others (web-server, database and translation). The scripts are pushed from ansible machine to the deployed machines.

There is ansible_vm_setup.sh which is used create all necessary virtual machines on OpenNebula. It downloads necessary packages, login to OpenNebula, instantiates machines, copies public ssh, pings machines, sets vault variables, hosts file, launches playbooks.

Ansible playbooks have 3 setup scripts for each machine and one download playbook which takes artifacts from gitlab CI/CD jobs and sends it to webserver machine.

- Database setup creates user in database straysafe with superuser privilages and grant connection access. Run tables.sql and breeds.sql files. The update playbook renames the current database to straysafe_v2 or v3, etc. and creates a new database straysafe, runs updated sql scripts.

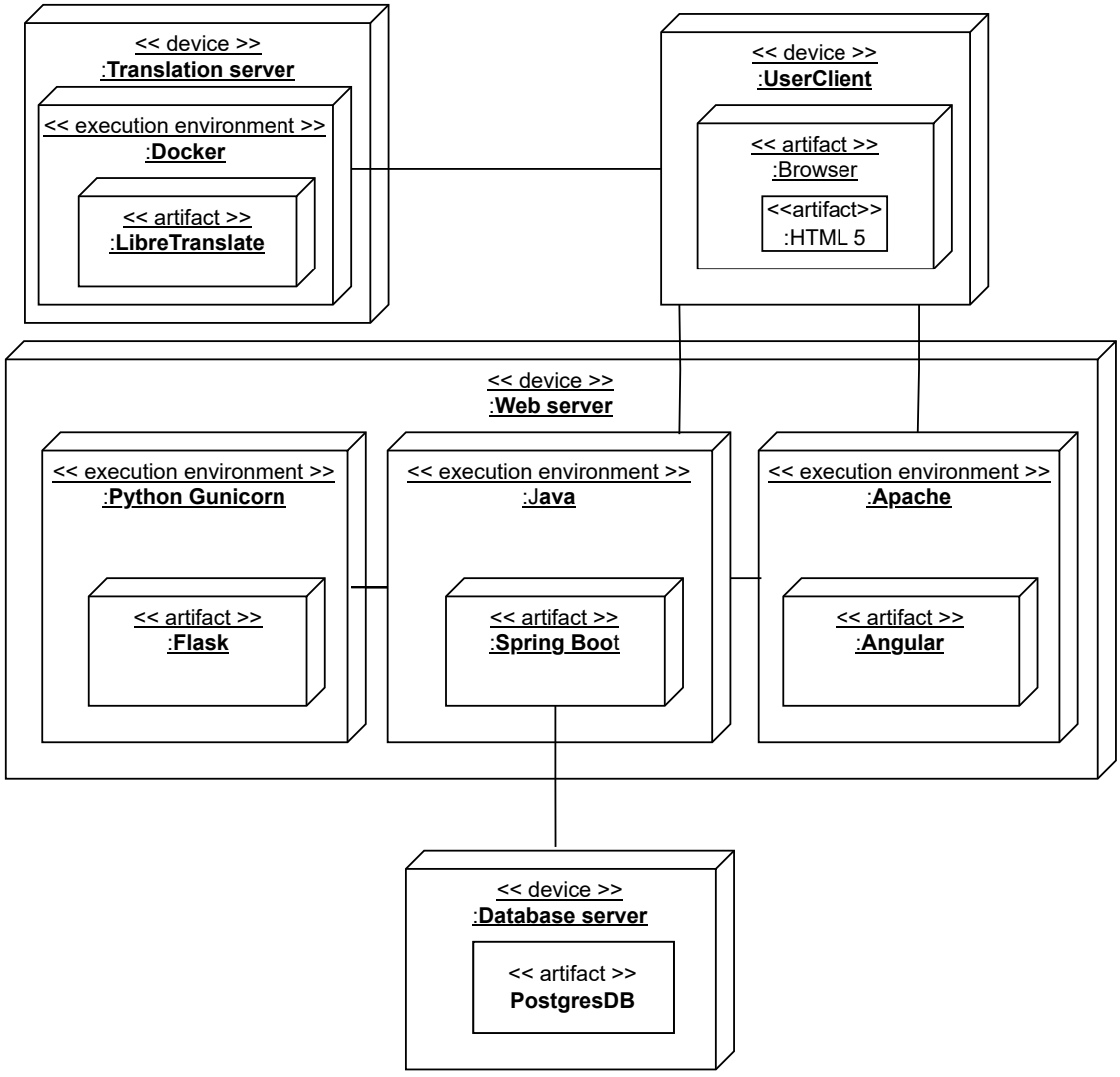

Figure 9. StraySafe Deployment UML diagram

- Webserver setup is split into two parts. The setup script sets apache configs, creates systemd services, installs necessary packages. The download playbook fetches artifacts created from gitlab CI/CD jobs - gets last pipeline id, finds the job ids associated, downloads artifacts based on job ids. Python is downloaded using pipx virtual environment, .env is added to the installed environment. Spring uses one jar file to launch the program. Angular built files are added to /var/www/html and changed permissions to www-data owner and group.
- Libretranslate is configured by cloning the libretranslate repository, creating python venv, installing gunicorn in order to launch multiple instances of the program. English to Lithuanian and vice versa language models are installed. Nginx config and systemd service is added to launch it on startup and track the program process.

4.4 Database

Relational schema (Figure 10), which is built according to relational modeling entity relational model presented in Figure 5, shows tables, attributes, data types, keys, and references. A few key points are outlined below:

- Images `image_url`) and files for the image analysis (`texture_histogram` and `color_histog` are storing in web-server files, not inside tables. In tables stores only the path to the files.
- Most attributes that have a limited list of possible options (for example `report_type`, `report_status`, `pet_type`, `pet_breed` and others) have **integer** data type. In front-end and back-end written enumerations for these values, for example tables 1 and 2.

0	lost
1	seen
2	found

Table 1. `report_type` enumeration

0	dog
1	cat
2	other

Table 2. `pet_type` enumeration

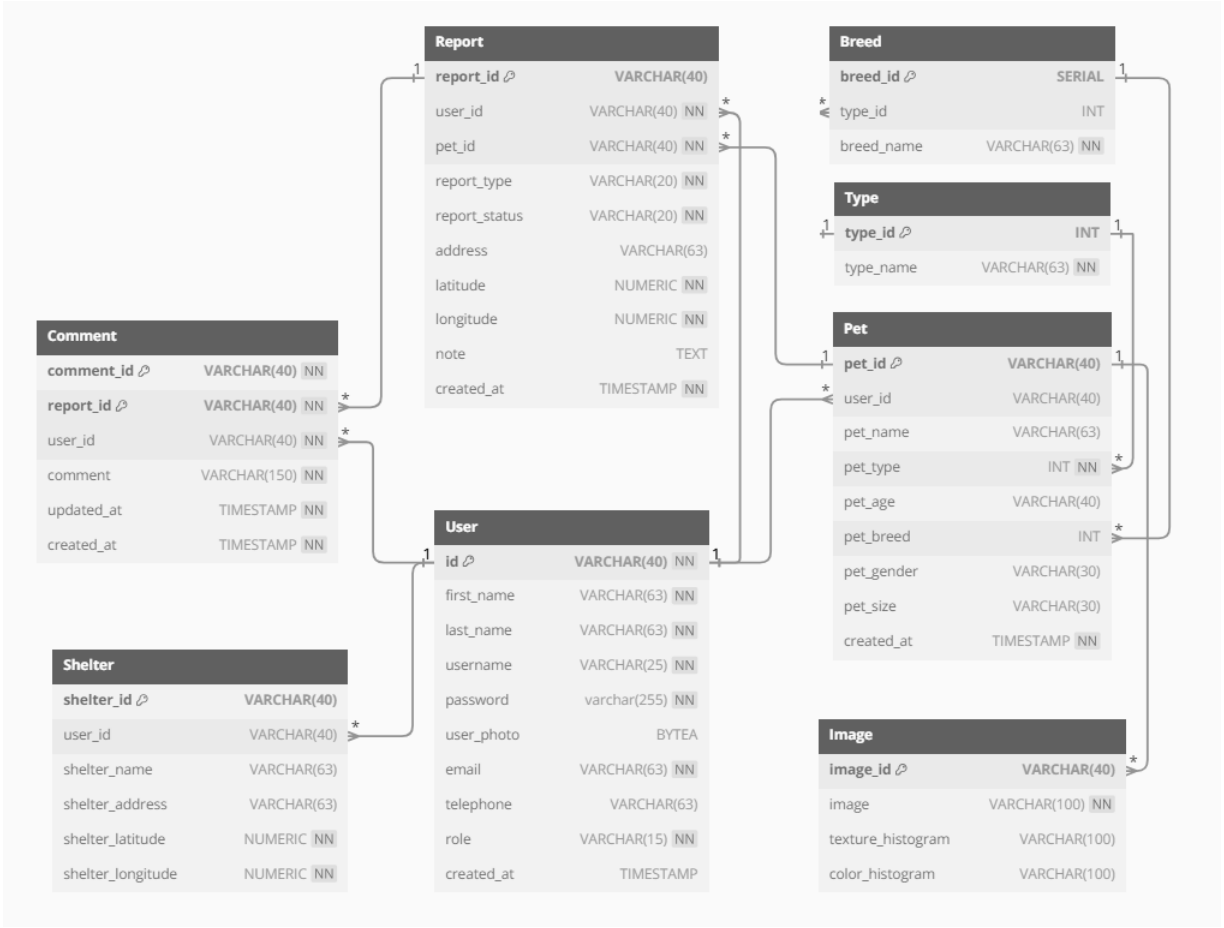

Figure 10. Relational Schema

All SQL files related to table creation, data insertion, functions, and triggers are located in the textttsql_scripts directory. The `tables.sql` file contains statements for creating database tables, `breeds.sql` and `data.sql` files contains statements for inserting all breeds and data (e.g. moderator account). The file contains dummy statements for inserting data. The The `trigger.sql` file contains a trigger that automatically updates the report status if 2 weeks have passed since it was created.

4.5 Authentication

The authentication in our system relies on JSON Web Tokens(JWT) which are used for securely managing user sessions. JWTs are made of three parts: header, payload and signature.

The header specifies the token type JWT and the algorithm used for signing the JWT token. The payload contains the claims, which are statements about the user and additional metadata. The information in payload is encoded but not encrypted. Thus, it does not contain any sensitive data because it can be decoded easily. The most crucial part for security is the signature which is generated by applying the signing algorithm specified in the header in our case HMAC-SHA256 to the encoded header, payload, and a secret key that is kept on the server-side. This process verifies that the sender of the JWT token is actually the user is the one he says he is.

Figure 13 shows the flow of how the JWT gets handled during a request to an endpoint requiring authentication.

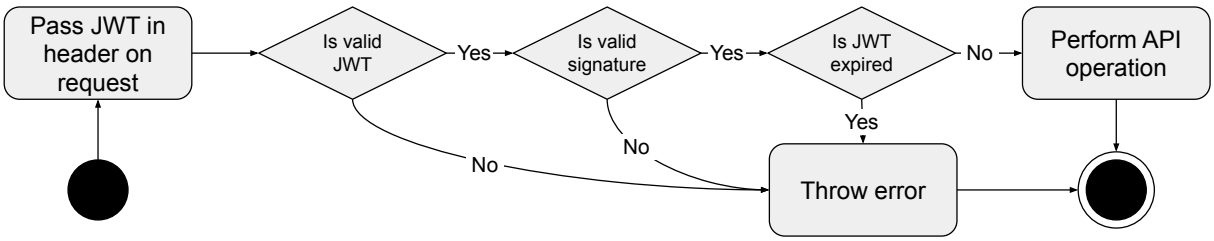

Figure 11. Authentication flowchart

4.6 Multi-language support

By offering content in multiple languages, the site can cater to users from different language groups. To change the language user should select it from navigation bar. The language to load initialy will be chosen according to the browser preferences.

The website interface translated using `@angular/localize` package. HTML tags with interface text which is marked with special `i18n` attribute are extracted into `assets/messages.xlf` source translations file. Another file (e.g. `messages.lt.xlf`) based on source file have inside translations.

Comments and notes created by users that are not static and cannot be pre-translated are handled by LibreTranslate. The dynamic translations are not saved and can just be fetched on demand. To translate report element user should press special button. LibreTranslate will automatically detect the source language, when the target language passing to the request based on the URL path.

4.7 Image

4.7.1 Image processing

For displaying images for pet reports first of all a mechanism to store and process these images had to be implemented, to initially store the pet photo it had to be cropped and compressed to do so `ngx-image-cropper` package was utilized, the `imageQuality` parameter was used to compress and crop the images while keeping the same aspect ratio. Upon cropping the images they are converted to `PNG` format and returned as blob file format. Upon sending request to save the image in the database the image is converted to `Base64` format and sent in the request headers sent to an endpoint responsible for saving pet data afterwards it is converted to a byte array to check integrity of the image, the first 8 bytes of the array are then checked to make sure the signature fits that of a `PNG` format. After that the image is saved locally and its path is stored in the database. To display the images for the user, the images are read using the path stored in the database, converted to `Base64` and returned to frontend for displaying.

4.7.2 Image comparison

The comparison algorithm uses color and texture of the pet to figure out similarities between pets. For the convenience of the user we provide a way to avoid searching through hundreds of different reports and just find the most similar to the pet that is being searched for. The full comparison flowchart can be seen in Figure 12.

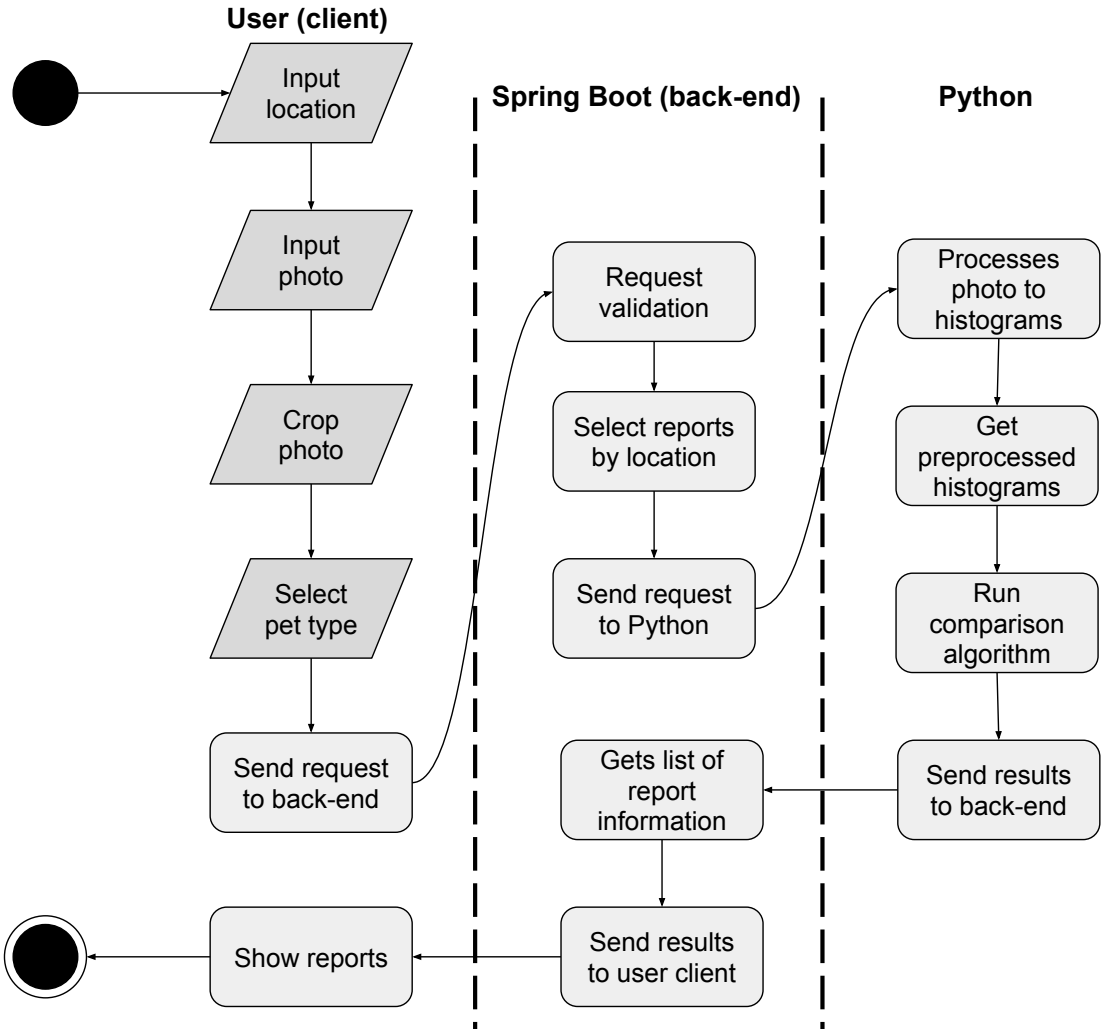

Figure 12. Image comparison flowchart

To use the system, the user needs to follow these steps: First, specify the general location where the pet was last seen. Next, add a photo of the pet to the system. Finally, crop out the background from the photo to ensure a clearer image of the pet.

On the backend, Spring Boot receives and validates the request from the user. It then narrows down the list of report IDs that need to be compared based on the specified location and forwards this request to Python. Python processes the preprocessed histograms of the report IDs or generates new histograms if they do not already exist. It then compares these histograms using color and texture analysis methods, as detailed in the analysis and implementation sections. Once the comparison is complete, Spring Boot receives the report ID and its similarity score relative to the uploaded photo. Finally, Spring Boot returns the relevant report information to the frontend based on the similarity results.

4.7.3 Color histogram

The color histogram calculates the distribution of colors within the image. It computes the histogram by dividing the image's color channels (blue, green, and red) into 32 bins each, creating a multidimensional histogram. The histogram is then normalized to a range of 0 to 1, ensuring that the values are scaled appropriately for comparison purposes. This normalized histogram is then compared using the Battachryya comparison algorithm.

4.7.4 Texture histogram

Texture histogram is calculated using the Local Binary Pattern (LBP), which involves comparing each pixel in the image to its surrounding neighbors to form a binary number based on whether the neighboring pixels are brighter or darker than the center pixel. The algorithm works as follows:

```
   Input: Grayscale Image Pixel Array - grayArray
   Output: LBP Image Pixel Array - lbpArray
1  Initialize lbpArray with zeros, the same size as grayArray;
2  for i ← 1 in grayArray.height − 1 do
3      for j ← 1 in grayArray.width − 1 do
4          centerPixel ← grayArray[i, j]
5          for k ← −1 to 1 do
6              for l ← −1 to 1 do
7                  if k = 0 and l = 0 then
8                      continue
9                  if grayArray[i + k, j + l] > centerPixel then
10                     binaryString ← 1
11                 else
12                     binaryString ← 0
13         lbpArray[i, j] = convert binaryString to pixel value;
```

We take the surrounding 3x3 pixel matrix and compare surrounding pixels. If they are larger than the middle they get 1, else 0. The binary output gets converted into a number. This number is then assigned to a bin that is between 0 and 255 because there are 2^8 possible values. 1 or 0 means two possible values and 8 different binary numbers gives 256 available bins.

A more understandable visualization can been seen in Figure 13 [7, 20].

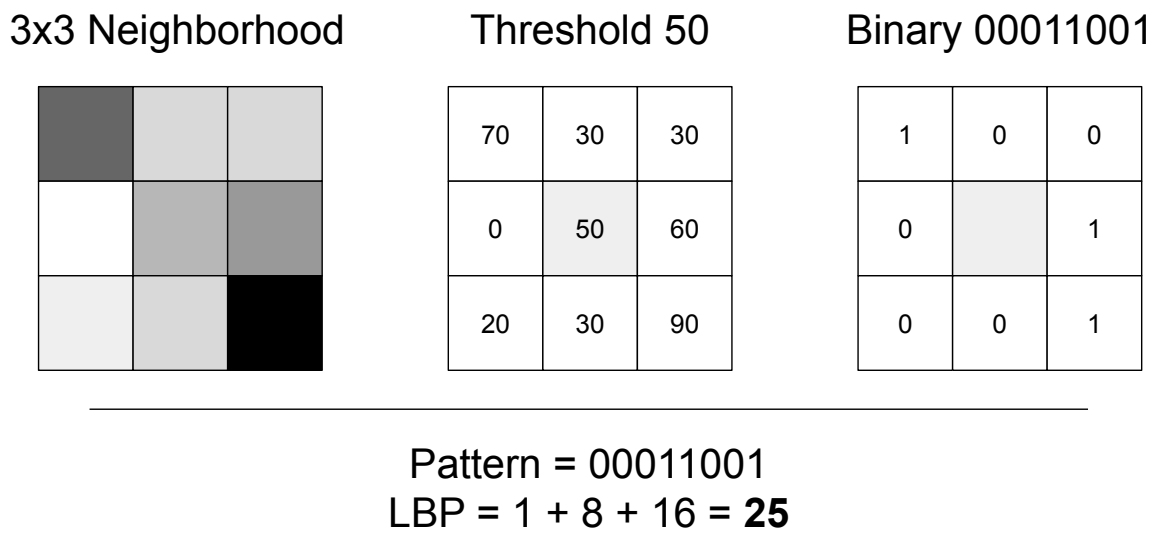

Figure 13. LBP visualized

4.7.5 Histogram comparison

The histograms are processed once and stored in a .npy file. The database holds the path to the .npy file and when they need to be compared it uses these preprocessed, normalized npy arrays.

Both color and texture histograms ranges from 0 (most similar) and 1 (least similar). We take these and combine them 0.5 * color-hist + 0.5 * texture-hist. The end result is from 0 (most) to 1 (least) similarity.

Input: Color and texture histogram list - histList, Original image histograms - origHists
Output: Similarity array - similarity

Initialize $similarity$ array
if $histList.length != 0$ **then**
    for row ***in*** $histList$ **do**
        $colorMetric \leftarrow compareWithBhattacharyya(origHists.cHist, row.cHist)$
        $textureMetric \leftarrow compareWithChiSquared(origHists.tHist, row.tHist)$
        $textureNormalized \leftarrow textureMetric/(1 + textureMetric)$
        $similarity[row.imageId] \leftarrow 0.5 * colorMetric + 0.5 * textureNormalized$

4.8 Email and poster generation

An email service was implemented for notifying users of successful events such as welcoming email after registering and providing users data of their submitted reports. The mailing service also allows attaching files in byte array format therefore it was also used to attach a generated PDF report on the users `LOST` pet. To implement the email service `springframework.mail` package was used to implement a mailing service with a combination of an app password of google gmail.

Poster generation for the `LOST` pet report email involves several steps to convert HTML content into a PDF document, these steps include: parsing the HTML, calculating the layout for the content inside the HTML to fit the PDF pages. After that the renderer converts each part of the HTML into elements suitable for a PDF document, as the HTML is being processed asynchronoussly the PDF is being generated. Several pet attributes are being used and stored in the pdf report poster Figure 14 such as: animal name, breed, lost since date, last seen at address and contact information of the report creator.

Conclusions and Future Work

In this project, we introduced StraySafe a lost animal platform for pet owners to find their lost pets by crowd-sourcing and utilizing the solutions provided, which are based on human-computer interaction principles. The application has alternative ways for pet owners to find their pets either by submitting reports for other community members to see, uploading a pet image to compare with seen pet reports in the user's area, or utilizing the map functionality to search through community posted reports or pet shelter pages.

Straysafe can be improved in the future with image comparison algorithms by using machine learning models to better compare and filter the images. The platform could also add support for users who are using pet collar trackers. Heat maps could be generated by aggregating data on found pet locations. The crowd-sourcing effect could be further improved by delivering notifications to people walking near the lost pet report location. Moreover, the platform itself still has many issues left to address both for users and shelters.

References

[1] Wilhelm Burger and Mark J Burge. Histograms and image statistics. In *Digital image processing: An algorithmic introduction*, pages 29–48. 2022.

[2] Anthony Morán Cabezas, Andrés De La Torre Macias, Jefferson Morán Cabeza, and José Tubay Vergara. Analysis and implementation of a satellite tracking system applied to pets using free software with gps and gsm technology. *Espirales Revista Multidisciplinaria de investigación*, 7(47):48–56, 2023.

[3] Cat Breeds List. All cat breeds, 2024. Accessed: 2024-06-16.

[4] Meo Vincent Caya, Emmanuel D Arturo, and Chezjon Q Bautista. Dog identification system using nose print biometrics. In *2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, pages 1–6, 2021.

[5] Lien-Wu Chen and Jun-Xian Liu. Easyfind: a mobile crowdsourced guiding system with lost item finding based on iot technologies. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 343–345, 2019.

[6] Dog Breed Info Center. Italian bichon, 2024. Accessed: 2024-06-16.

[7] Aihala Halapathirana. Understanding the local binary pattern (lbp): A powerful method for texture analysis in computer vision, 2019. Accessed: 06-16-2024.

[8] Emily Harburg, Yongsung Kim, Elizabeth Gerber, and Haoqi Zhang. Crowdfound: A mobile crowdsourcing system to find lost items on-the-go. In *Proceedings of the 33rd annual ACM conference extended abstracts on human factors in computing systems*, pages 1537–1542, 2015.

[9] Santosh Kumar and Sanjay Kumar Singh. Monitoring of pet animal in smart cities using animal biometrics. *Future Generation Computer Systems*, 83:553–563, 2018.

[10] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.

[11] Nikhil Naik, Sanmay Patil, and Madhuri Joshi. A scale adaptive tracker using hybrid color histogram matching scheme. In *2009 Second International Conference on Emerging Trends in Engineering & Technology*, pages 279–284, 2009.

[12] Pamesti/Rasti Daiktai Lietuvoje. Pamesti/Rasti Daiktai Lietuvoje Facebook Group. https://www.facebook.com/groups/PamestiRastiDaiktaiLietuvoje/, 2024. [Accessed: 06-15-2024].

[13] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505, 2012.

[14] PawBoost.com. Pawboost: Helping lost pets, 2024. Accessed: 06-14-2024.

[15] PetcoLove.org. Petco love: Finding lost pets, 2024. Accessed: 06-14-2024.

[16] PetRadar.org. Petradar: Lost pet finding service, 2024. Accessed: 06-14-2024.

[17] Jianfeng Ren, Xudong Jiang, and Junsong Yuan. A chi-squared-transformed subspace of lbp histogram for visual recognition. *IEEE Transactions on Image Processing*, 24(6):1893–1904, 2015.

[18] Christie Siettou. Evaluating the recently imposed english compulsory dog microchipping policy. evidence from an english local authority. Report, University of Nottingham, 2018. Accessed: 05-21-2024.

[19] Renjie Song, Ziqi Zhang, and Haiyang Liu. Edge connection based canny edge detection algorithm. *Pattern Recognition and Image Analysis*, 27:740–747, 2017.

[20] N. Srilatha and V. Lokeswara Reddy. Image texture analysis with local binary patterns: A review. In *2021 Innovations in Power and Advanced Computing Technologies (i-PACT)*, pages 1–6, 2021.

[21] Statista. Global smartphone penetration per capita since 2005. https://www.statista.com/statistics/203734/global-smartphone-penetration-per-capita-since-2005/, 2024. Accessed: 05-21-2024.

[22] Tagmefy.com. Tagmefy: Pet tagging and recovery, 2024. Accessed: 06-14-2024.

[23] Songsri Tangsripairoj, Parit Kittirattanaviwat, Kamonwan Koophiran, and Lalita Raksaithong. Bokk meow: A mobile application for finding and tracking pets. In *2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 1–6. IEEE, 2018.

[24] Dirk Van Der Linden, Matthew Edwards, Irit Hadar, and Anna Zamansky. Pets without pets: on pet owners' under-estimation of privacy concerns in pet wearables. *Proceedings on Privacy Enhancing Technologies*, 2020(1):143–164, 2020.

[25] Changqing Wang, Jiaxiang Wang, Quancheng Du, and Xiangyu Yang. Dog breed classification based on deep learning. In *2020 13th International Symposium on Computational Intelligence and Design (ISCID)*, pages 209–212, 2020.

[26] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

Appendices

Figure 14. Generated pet poster

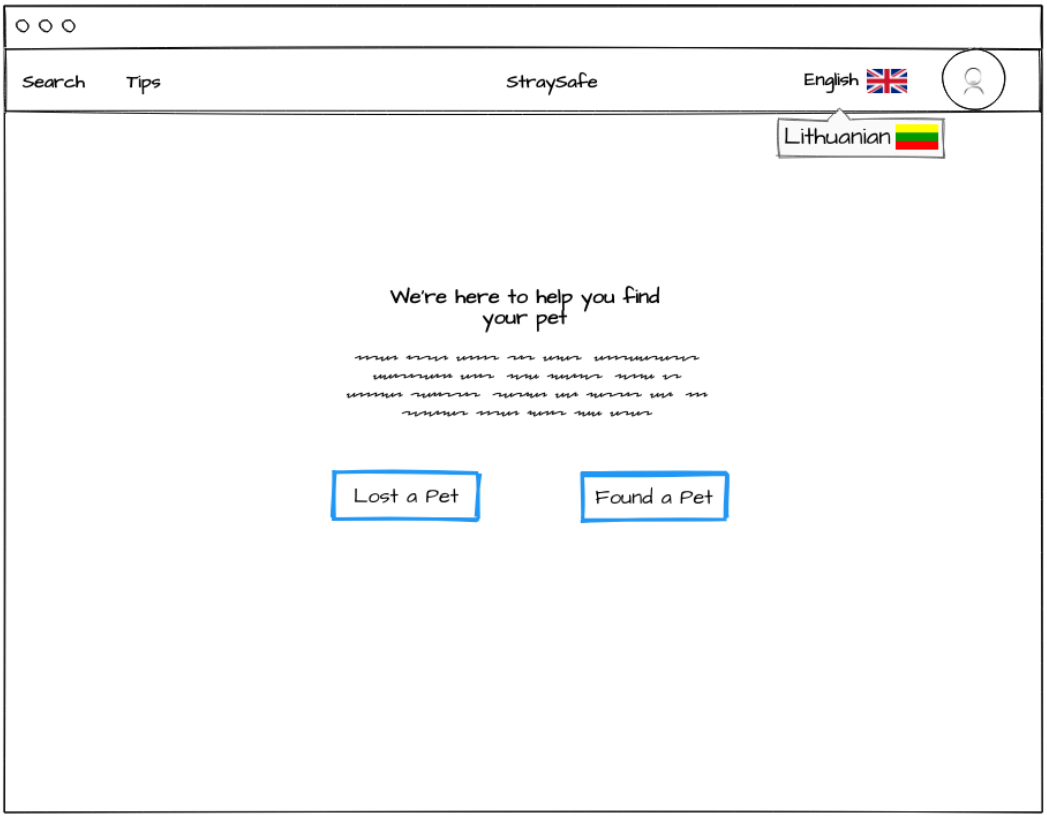

Figure 15. Welcome page (*wireframe*)

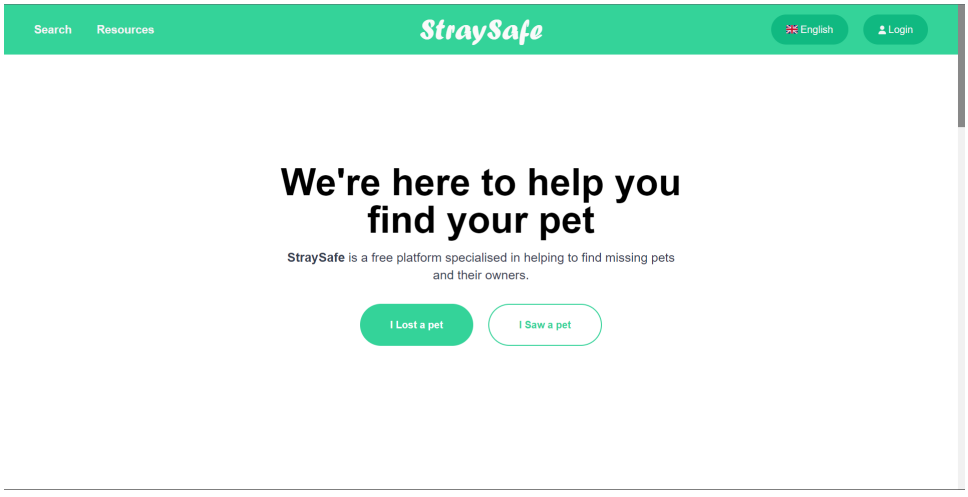

Figure 16. Welcome page (*implemented*)

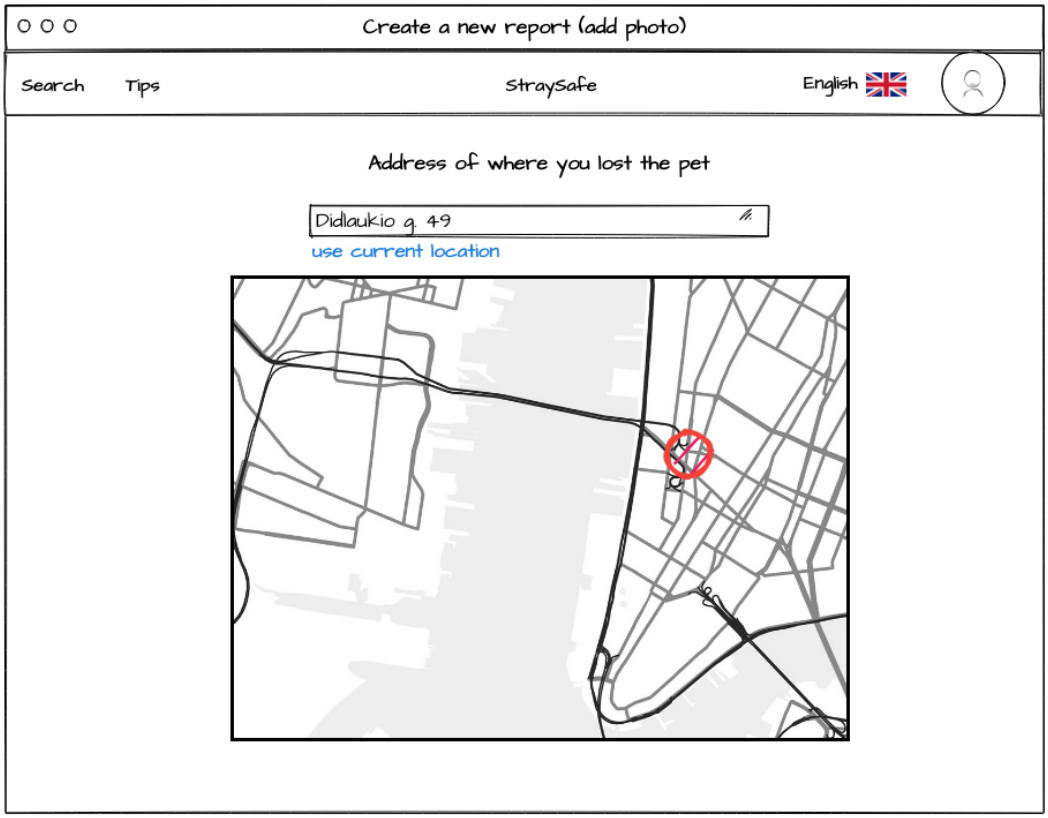

Figure 17. Create a new report: specify the location (*wireframe*)

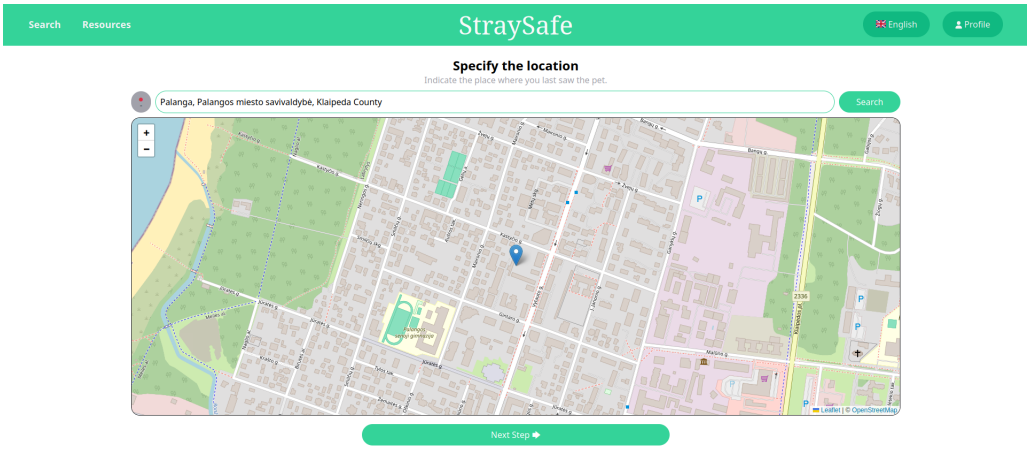

Figure 18. Create a new report: specify the location (*implemented*)

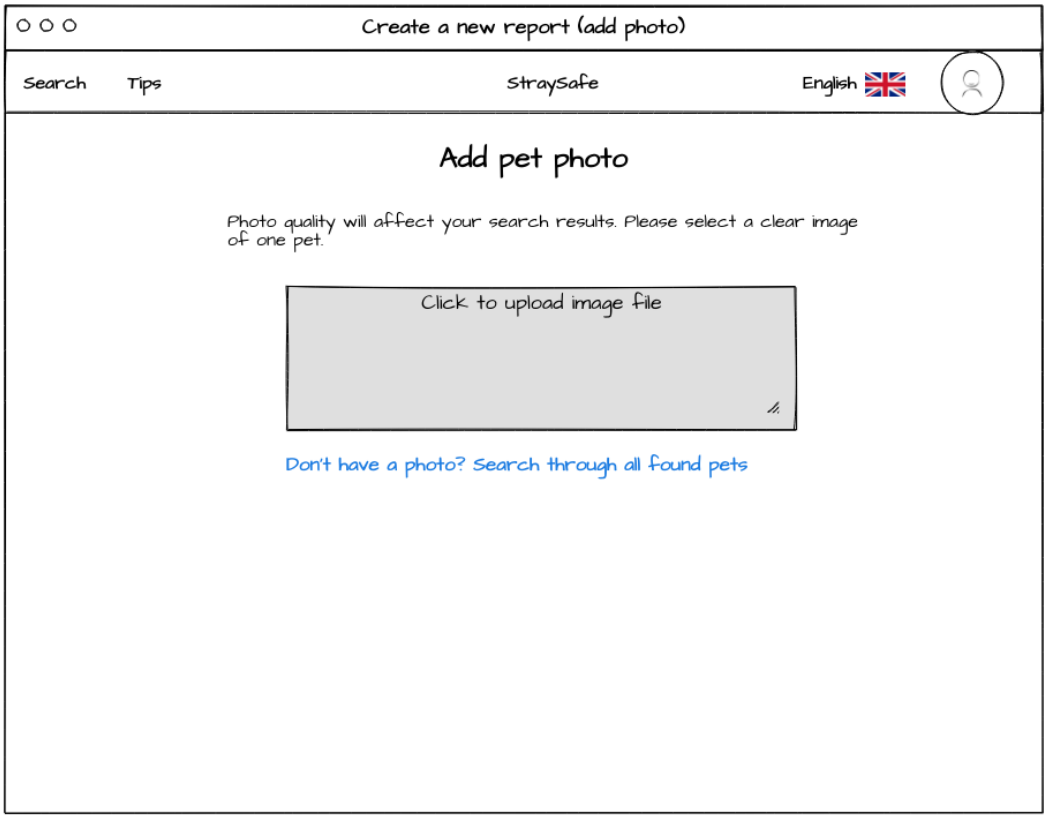

Figure 19. Create a new report: image upload (*wireframe*)

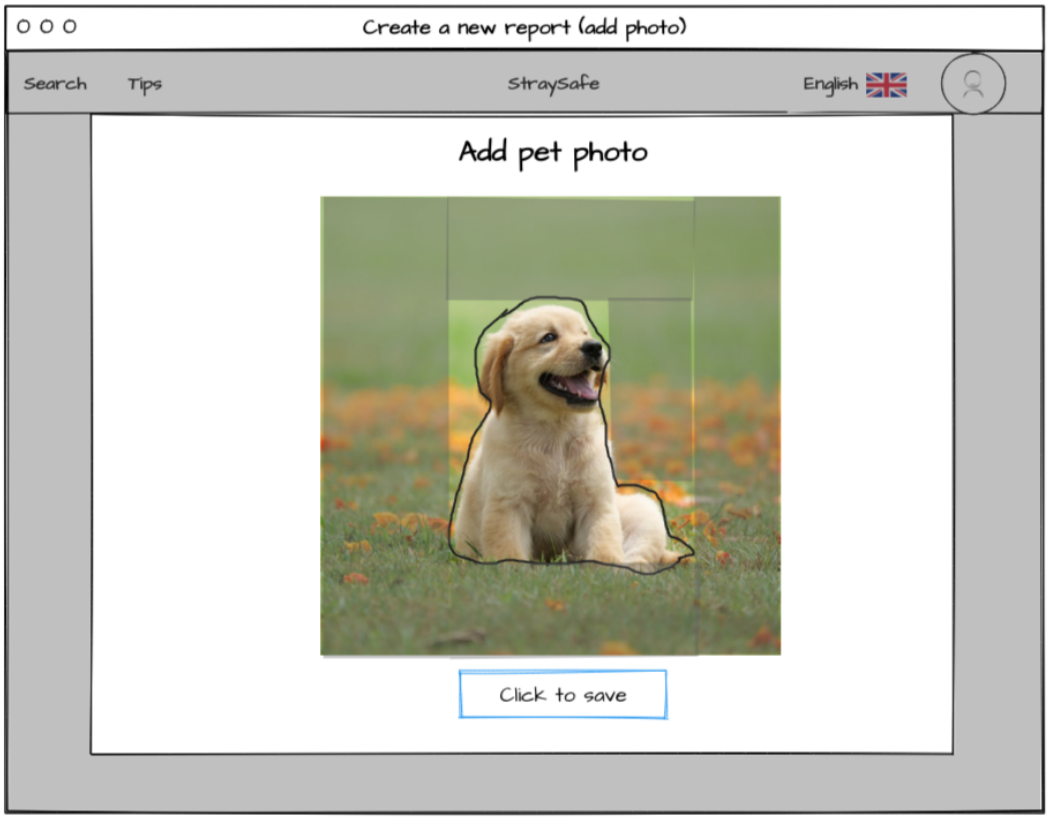

Figure 20. Create a new report: crop the photo (*wireframe*)

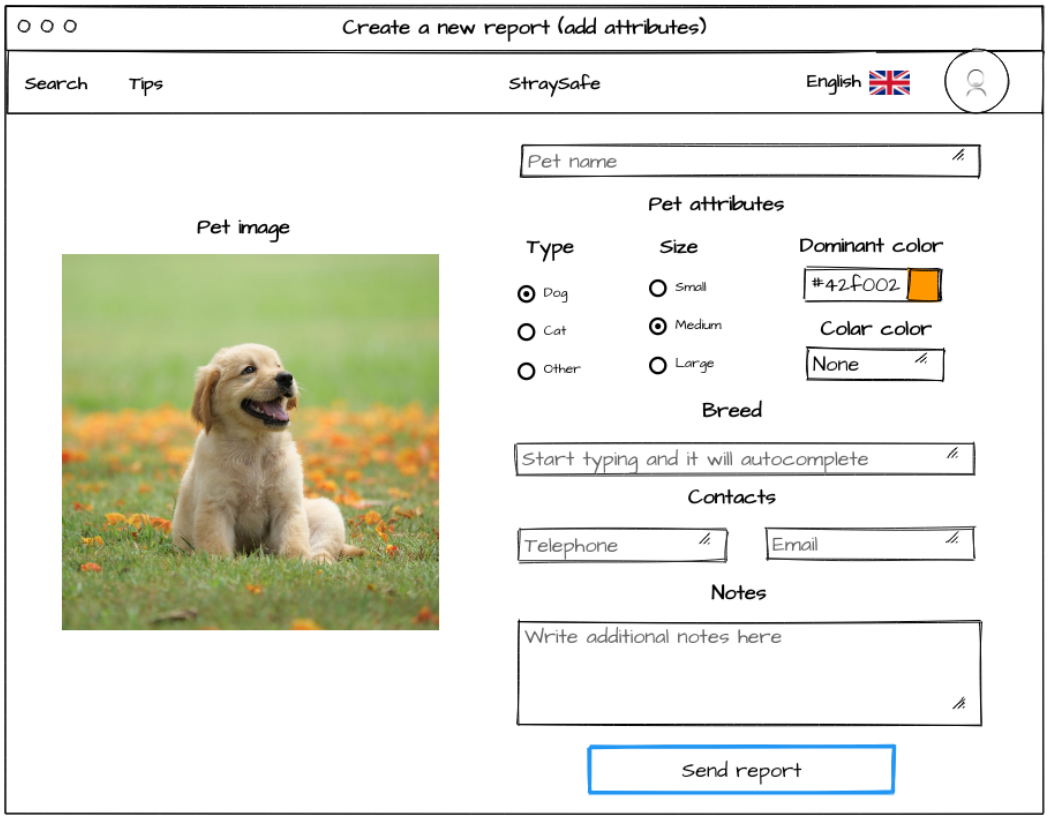

Figure 21. Create a new report: fill pet attributes (*wireframe*)

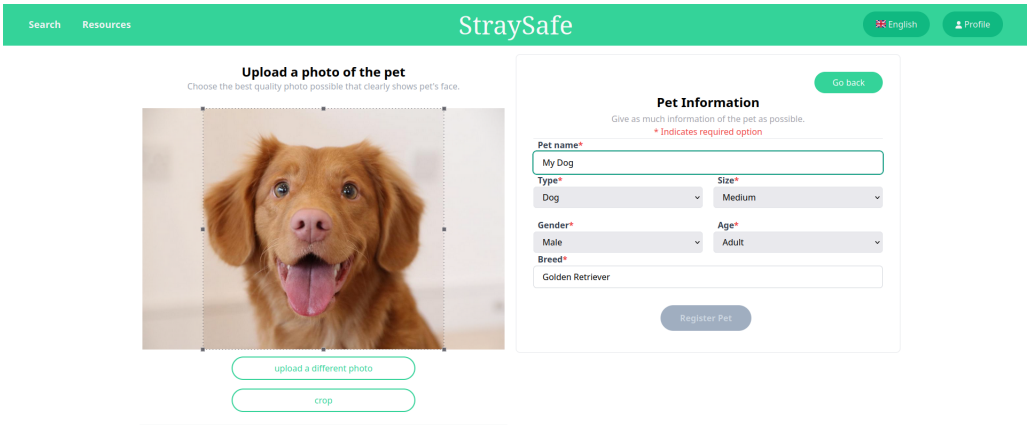

Figure 22. Create a pet (*implemented*)

Figure 23. Report (*wireframe*)

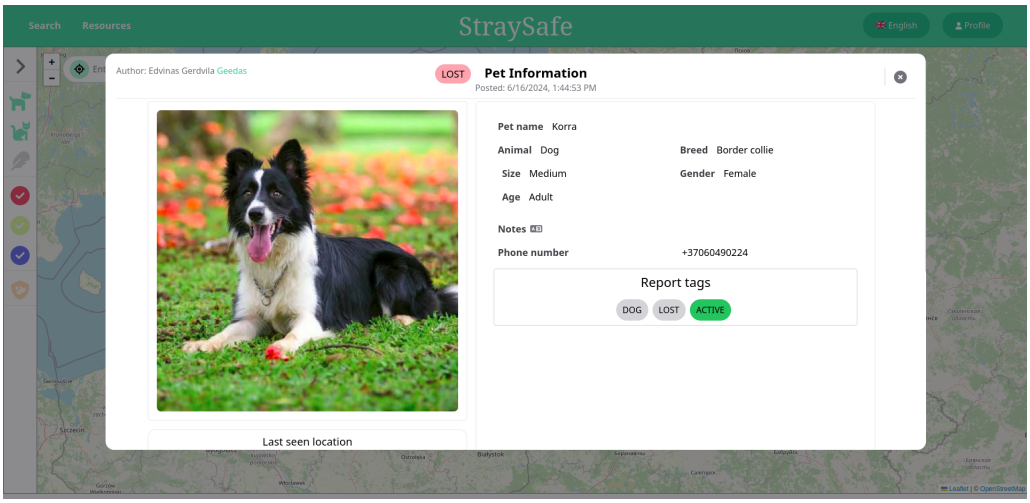

Figure 24. Report (*implemented*)

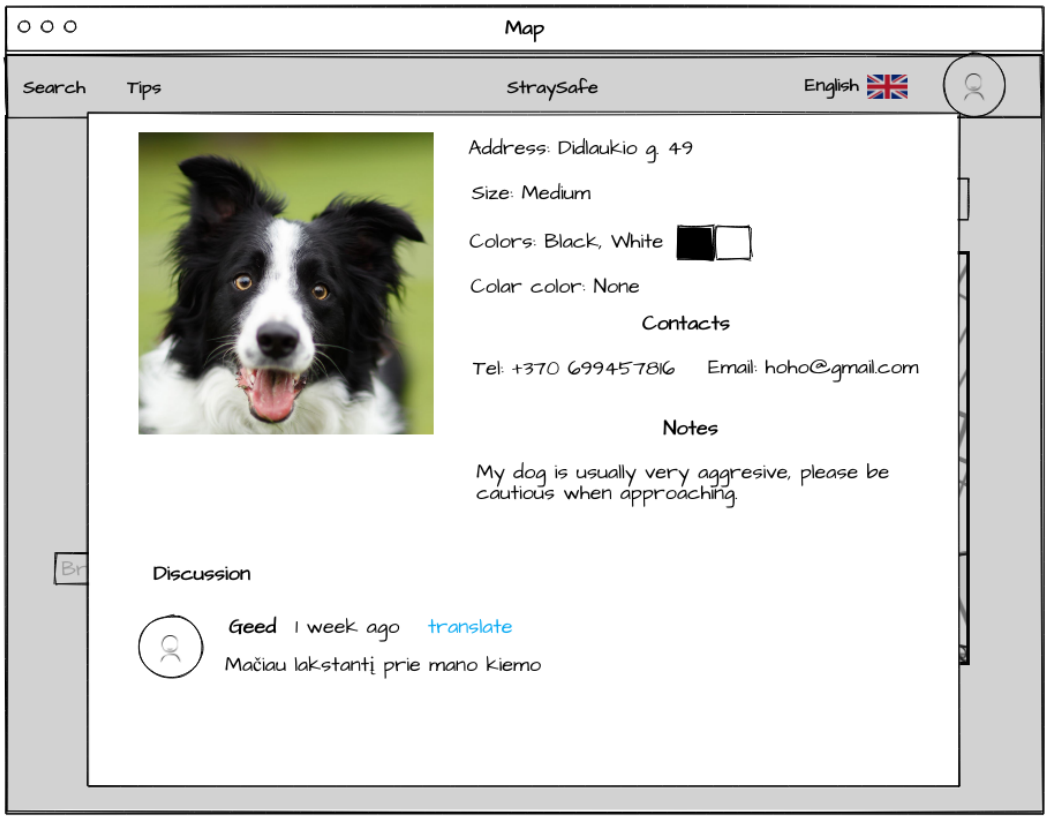

Figure 25. Report comments (*wireframe*)

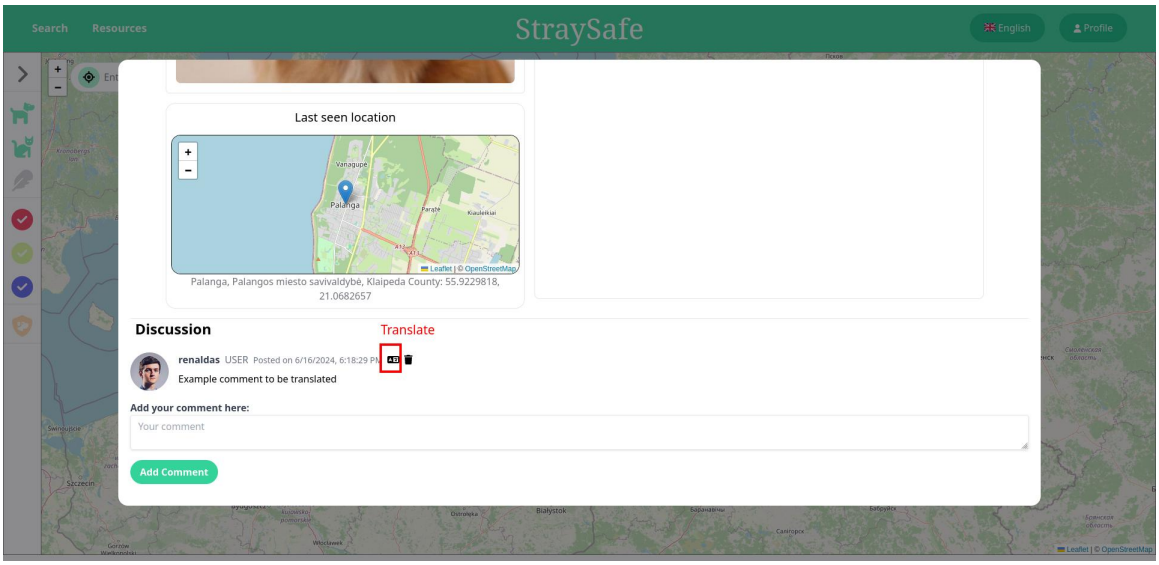

Figure 26. Report comments with translation (*implemented*)

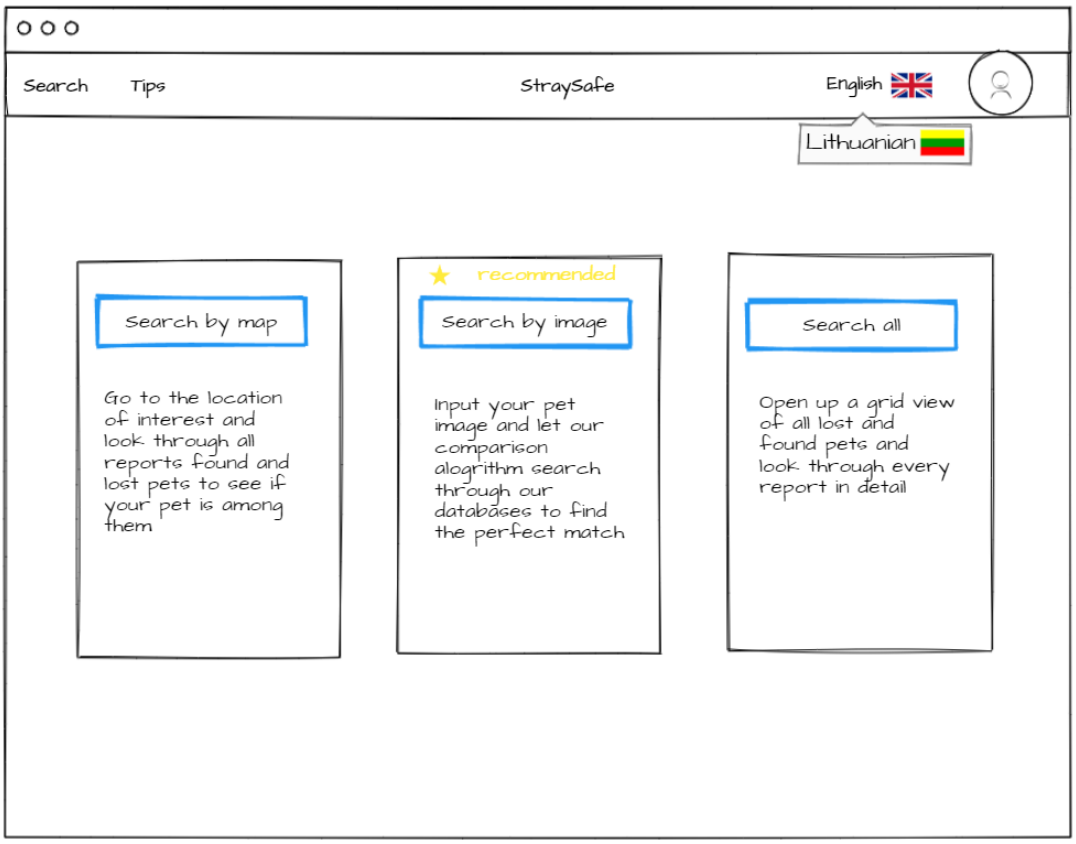

Figure 27. Choose a search method (*wireframe*)

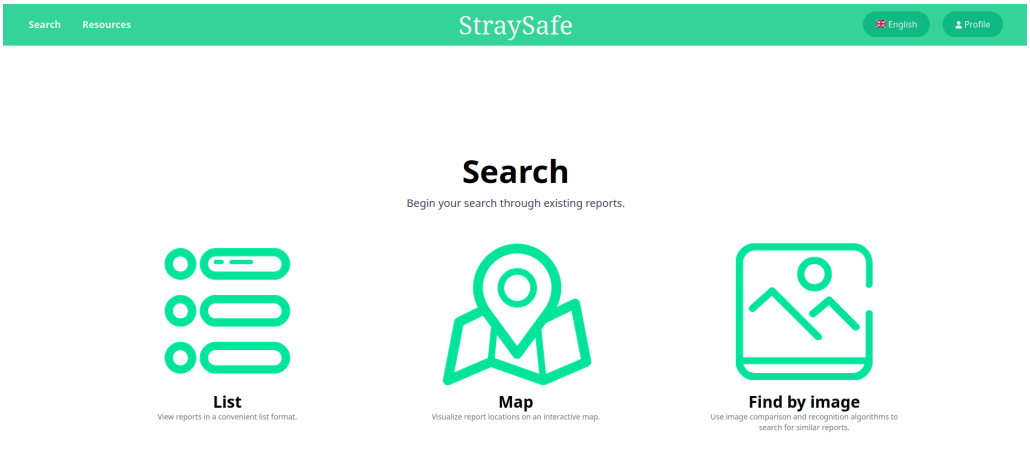

Figure 28. Choose a search method (*implemented*)

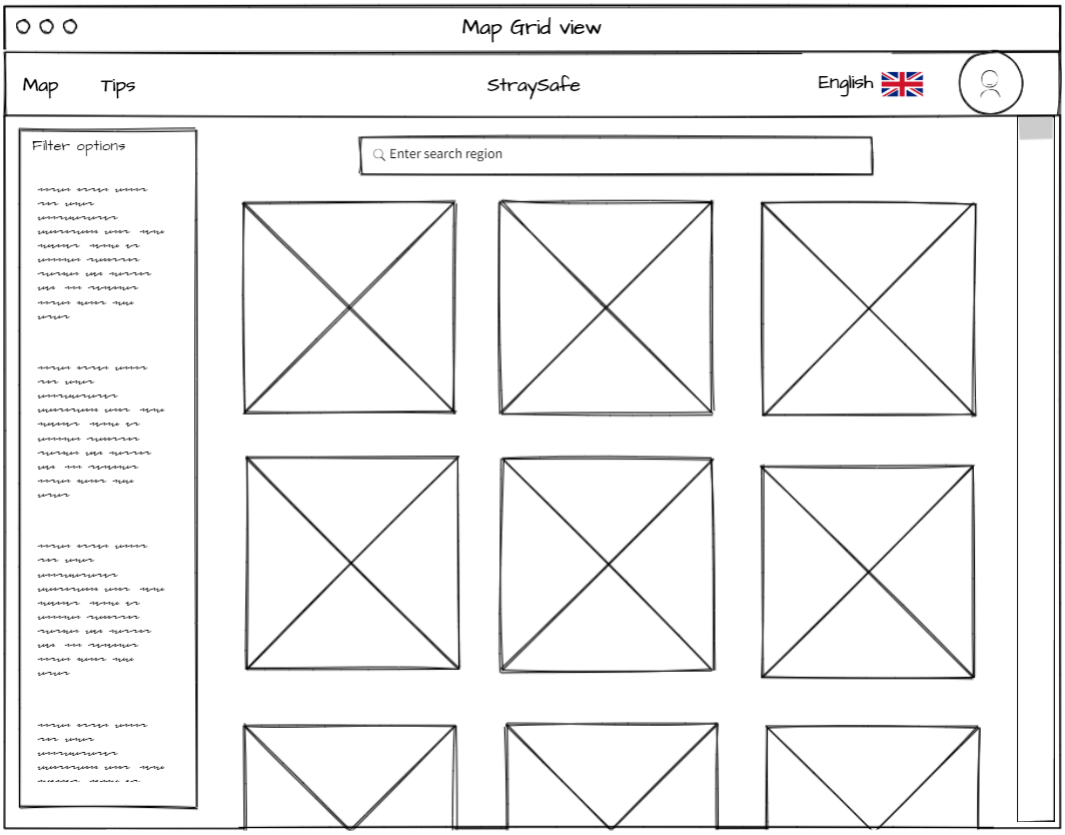

Figure 29. Report list (*wireframe*)

Figure 30. Report list (*implemented*)

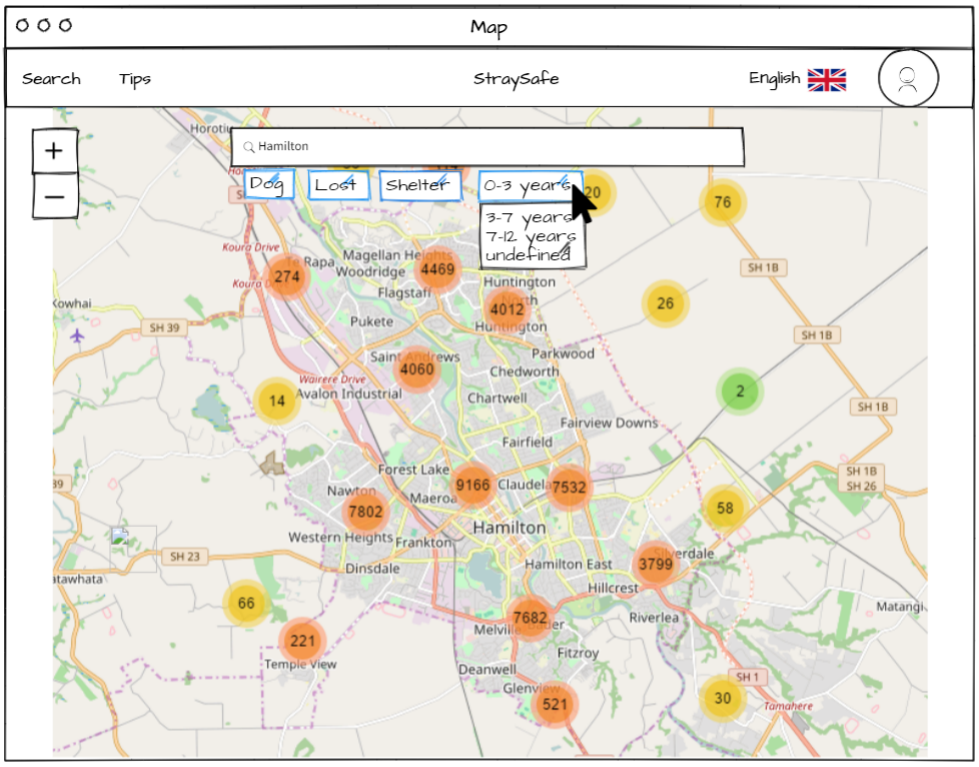

Figure 31. Map (*wireframe*)

Figure 32. Map: markes (*wireframe*)

Figure 33. Map: markers clustered (*wireframe*)

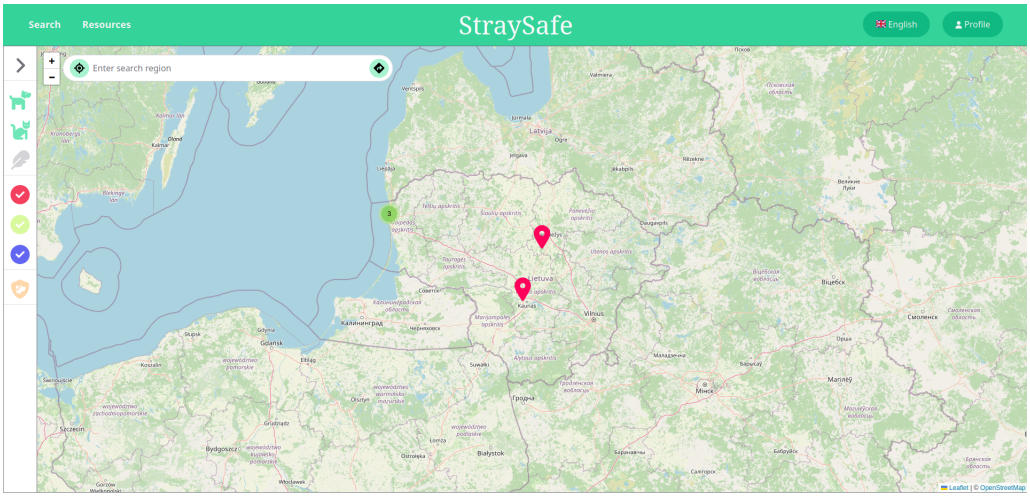

Figure 34. Map (*implemented*)

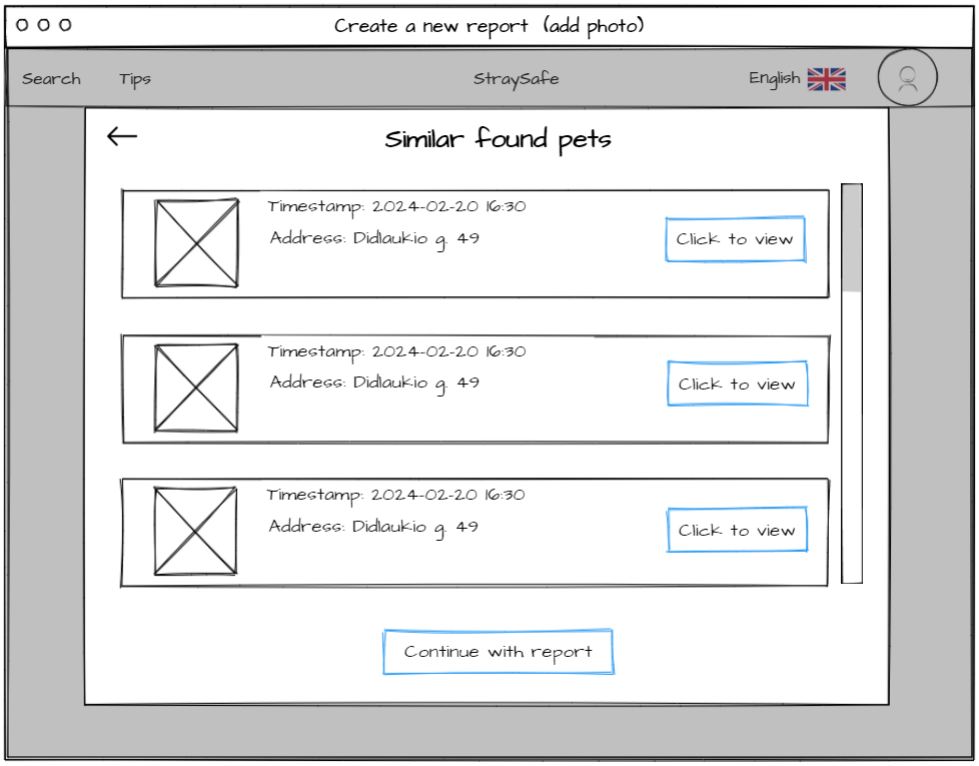

Figure 35. Similar reports (*wireframe*)

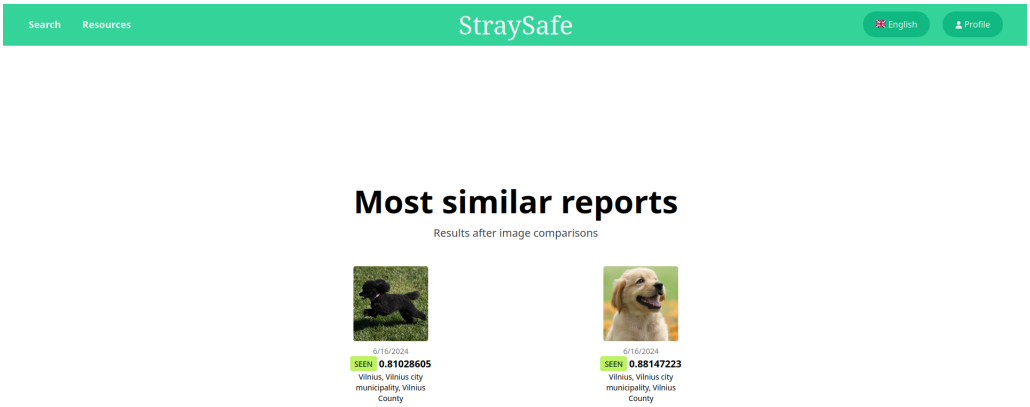

Figure 36. Similar reports, comparison by image (*implemented*)