

TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Department of Software Science

Kristjan Variksoo, Artjom Zarva, Veronika Tamm

IAIB193800, IAIB193420, IAIB195216

DETECTION OF PETS WANDERING IN A RESIDENTIAL

AREA

Bachelor Thesis

Supervisor

Raul Savimaa

PhD

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Tarkvarateaduse instituut

Kristjan Variksoo, Artjom Zarva, Veronika Tamm

IAIB193800, IAIB193420, IAIB195216

ELUPIIRKONNAS JÄRELEVALVETA UITAVATE LOOMADE

TUVASTAMINE

Bakalaureusetöö

Juhendaja

Raul Savimaa

PhD

Tallinn 2022

Author's declaration of originality

I hereby certify that We are the sole authors of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Kristjan Variksoo, Artjom Zarva,
Veroonika Tamm

.....

(signature)

Date: May 30, 2022

Annotatsioon

Käesoleva bakalaureusetöö raames on tehtud platvorm, et tuvastada elupiirkonnas vabalt liikuvaid lemmikloomi (koeri, kasse) ja sellest vajadusel SMS- või e-maili teel teavitada eelnevalt kokkulepitud adressaadile (operaator, omanik). Rakendus aitab loomaomanikel leida oma kadunud loomi ühelt platvormilt.

Rakenduse kasutajaliidesel on vastavalt sisselogimise andmetele kaks erinevat vaadet – omanikuvaade ning operaatori vaade.

Omanikul on võimalik registreerida uus loom, eemaldada enda loomad andmebaasist, anda tagasisidet masinõppe programmile ning jälgida kadunud loomi.

Operaatoril on võimalik lisada platvormile uus loomaomanik ning lisada, kustutada, muuta andmeid lemmikloomade tuvastamise andmebaasist ja eemaldada loomaomanike.

Rakendusega seotud veebilehed on nähtaval lisas 5, 6.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 35 leheküljel, 7 peatükki, 11 joonist, 2 tabelit.

Abstract

This bachelor thesis reviews a web application that serves as a platform for identify pets (dogs, cats) moving freely in the residential area and, if necessary informing the previously agreed addressee (operator, owner, etc.) by SMS or e-mail. The platform helps pet owners find their lost animals using one platform.

The application consists of two views — pet owner and operator.

The owner has the possibility of registering new pets, removing pets from the database, giving feedback to the machine learning algorithm and keeping track of lost animals.

The operator has the possibility of adding new pet owners to the platform and adding, deleting, editing data from the pet identification database as well as removing pet owners.

The web pages related to the application are seen in appendices 5, 6.

This application is finished at the time of writing this thesis. The thesis is in English and contains 35 pages of text, 7 chapters, 11 figures, 2 tables.

List of abbreviations and terms

Back-end	The server-side part, which is located away from the client, and generally deals with queries sent by the user interface.
Bootstrap	Open-source CSS framework
CSS	Cascading Style Sheets The markup language used to mark up the design of web pages.
Django	Python-based open-source web framework
Figma	Online collaboration tool used to build digital products
Front-end	Converting data into a graphic interface using HTML, CSS, JavaScript, so that users can. interact with and view the data.
Keras	Open-source software library that provides a Python interface for artificial neural networks.
Python	High-level, interpreted, general-purpose programming language
TensorFlow	Open-source software library for machine learning
UI	User Interface
UX	User Experience
CPU	Central Processing Unit
GPU	Graphics Processing Unit

Table of Contents

List of Figures	viii
List of Tables	ix
1 Introduction	1
2 Background	3
2.1 The idea	3
2.2 Teams responsibility for developing the solution	3
2.3 Existing solutions and competitors	4
2.3.1 Social media groups	4
2.3.2 Websites	5
2.3.3 FindMyPet	6
3 Functional requirements	7
3.1 User roles	7
3.2 User authentication	7
3.3 Role 1: Pet owner	8
3.3.1 Submitting photos to the pet database	8
3.3.2 Asking for pet information	8
3.3.3 Getting information about a lost pet	9
3.3.4 Deleting a pet from the database	9
3.4 Role 2: Owner of a missing pet	9
3.4.1 Reporting a lost pet (pet is not in database)	10
3.4.2 Reporting a lost pet (pet is in database)	10
3.4.3 Request for missing pet movements	11
3.4.4 Getting information about a lost pet	11
3.4.5 Reporting animal as found	11
3.5 Role 3: Individual in residential area	12
3.5.1 Verification of the correctness of the processing of personal data	12
3.6 Role 4: Pet identification system operator	12
3.6.1 Submitting data and photos to the pet database	13
3.6.2 Reporting a lost pet	13
3.6.3 Getting information about a lost pet	14
3.6.4 Getting information about a stray pet	14
3.6.5 Operator requests for pets movements	14

3.6.6	Operator deletes a pet from the database	15
3.6.7	Verification of the correctness of the processing of personal data .	15
4	Petfind development	16
4.1	Implemented technologies	16
4.1.1	Python and Django	16
4.1.2	Bootstrap	17
4.1.3	TensorFlow and Keras	18
4.2	UX and UI design	18
4.2.1	Sitemap	19
4.2.2	Personas	19
4.2.3	User flow	19
4.3	Machine learning	20
4.3.1	Object detection	21
4.3.2	Data collection	21
4.3.3	Model training	23
4.3.4	Model evaluation	24
4.3.5	A model that develops over time	27
4.4	Limitations	28
4.5	Deployment	29
5	Results and discussion	30
5.1	Quantitative validation	30
5.1.1	Differentiate between a user’s pet and a similar animal seen on cameras	30
5.2	Privacy and confidentiality requirements	32
5.3	Feedback after testing from the supervisor	33
5.4	Key lessons	33
6	Proposed future improvements	34
7	Summary	35
	Bibliography	36
	Appendices	38
	Appendix 1 - Non-exclusive license for reproduc- tion and publication of a graduation thesis	38
	Appendix 2 - Website wireframe model	39

Appendix 3 - User Personas	40
Appendix 4 - Used icons	42
Appendix 5 - Client views	43
Appendix 6 - Pet identification system operator views	46
Appendix 7 - Database relational model	47
Appendix 8 - Testing feedback from the supervisor views	48
Appendix 9 - References	49

List of Figures

1	Website sitemap.	19
2	Process chart	20
3	Training/testing data image being labeled using labelImg.	22
4	Development of loss (smoothing = 0.6).	24
5	Development of mAP (smoothing = 0.6).	24
6	Example of model evaluation based on camera trap images (1). The left image is the model's prediction, the right image is the ground truth.	26
7	Example of model evaluation based on camera trap images (2). The left image is the model's prediction, the right image is the ground truth.	27
8	Image labeling tool in the application	28
9	Architecture diagram of the proof of concept deployment used for testing	29
10	Example of a page "Recently seen animals"	31
11	Example of a page "Images possibly showing my lost animal"	32

List of Tables

1	Number of labeled images with classes used in dataset for model training	23
2	Evaluation results run with TensorFlow 2 Object Detection API	26

1. Introduction

The aim of this bachelor thesis is to develop and implement a prototype that integrates sensors and data processing systems to identify pets (dogs, cats) moving freely in the residential area and, if necessary, to inform the previously agreed addressee (operator, owner, etc.) by SMS or e-mail. The platform is aimed at pet owners who do not have a convenient tool for finding their lost pet.

At the moment that there is no system for obtaining an overview of pets moving in an area without supervision (including missing and stray pets).

The most widely used existing solution in Estonia is a completely manual process of informing community members through social media posts and asking them for information about animals in the area. For example, the most popular Facebook group in Estonia is Kadunud ja leitud loomad ning koduotsivad hoiukodude/varjupaikade loomad. [1] There have been start-ups abroad that have created machine learning mobile apps, but these have failed.

Choosing to use track cameras as the sensors comes down primarily to their wide spread use as the industry standard solution for tracking of wild animals in a certain area along with the relatively recent move towards more camera based solutions as opposed to many other sensor types in the autonomous driving industry by example of Tesla.

Previous research to identify animals from track cameras has focused purely on machine learning and has excluded the surrounding application and its sustainability, including the collection of feedback and the improvement of the quality of the system over time.

The use of a neural network that develops over time results from the fact that training a highly accurate machine learning system requires a lot of data which has not been previously collected. As such a system was conceived that would in parallel collect that and improve itself to enable quicker deployment without a long winded separate data collection and tagging stage. This also enables constant improvements and the collection of data for future data-sets to improve the technology even further.

Expected outcomes of the thesis are:

- A neural network model that develops over time based on feedback and is able to distinguish pets from other objects (eg humans, wild animals)
- A web application running on a computer and mobile phone that allows you to see the animals detected by the system and notify the operator or people who sent the search query, whether or not a match was found
- A web application that can provide feedback to the neural network
- The system being developed must comply with the requirements governing the protection and confidentiality of personal data. (For example, the solution should include hiding the faces of passers-by)
- If possible, a complete real-time prototype solution that can read input data based on e-mails or SMS sent from the camera, as well as data from an agreed directory, such as a memory card, for a period of time (sensor and camera hardware provided by the customer).

In order to achieve the goal, the process will be split into four phases:

1. Analyzing existing solutions and competitors (Section 2.3)
2. Building a user interface (Section 4.2)
3. Data collection and model training (Section 4.3.2-4.3.3)
4. Experiment: testing our solution in a real environment (Section 5.3)

Validation of the developed system will be carried out through testing in a real environment.

This paper consists of six sections. The introduction composes the first part. In the second part, the theoretical background is introduced. The functional requirements in form of user stories are presented in the third part. The fourth part is the core of the paper: the application development process is described, the process of data generation and preparation is explained, the developed convolutional neural networks models are introduced, methods for scene classification, as well as model training and evaluation are described. Results of the trained network model and scene classification methods on the example of the scenarios described in Functional requirements are analysed in the fifth part. Lastly, a summary is presented in the sixth part.

2. Background

At the moment that there is no system for obtaining an overview of pets moving in an area without supervision (including missing and stray pets). In Estonia social media groups are used to find missing pets, but these are ineffective and time consuming. Solution to this problem is the Petfind web application.

This is a licensed application. To read the application code contact the author.

2.1 The idea

The idea to make a platform for pet owners came from our client and supervisor Raul Savimaa. The chance to solve a real life problem that affects us and acquire a deeper understanding about machine learning made us choose the given topic.

Our team consists of:

- Veroonika Tamm
- Artjom Zarva
- Kristjan Variksoo

2.2 Teams responsibility for developing the solution

As a development methodology, we introduced agile development, where after each development cycle, incremental functional development towards the desired end result is achieved as a result of close cooperation between the team and the customer. Waterfall is a static methodology, so it is not suitable for our project where client requirements may change during development.

Within the team, roles were divided as follows:

- Veroonika Tamm: UI and UX design, database management, connection management, prototyping
- Artjom Zarva: backend logic, machine learning

- Kristjan Variksoo: team leading, architecture, DevOps

We applied the principles of agile development by practicing pair programming, stand-ups, planning sessions, and sprints. We used Toggl Track to track our activities and time spent on tasks and meetings as this was recommended by the faculty and we had previous experience with the platform.

The potential to finish short of project objectives due to time constraints was identified early on due to a late start and as such during all stages of planning getting the most actual usable value out of all development time was a key requirement. As such many comfort features were cut and a lot of functionality was provided in a manner unsuitable for larger scale deployments but that fully satisfied the client requirements for the limited user base it would be deployed to. As such we successfully avoided over-engineering and over-optimization too early on in the product life-cycle. This was largely due to our past experiences with projects that spent a long time making different plans but actually failed to provide any usable iterations to the client until the end and as such could not benefit from the frequent feedback at the heart of the Agile methodology.

The suitability of the result was checked and validated at the end of each development cycle in a meeting, based on feedback from the client.

2.3 Existing solutions and competitors

The most widely used existing solution in Estonia is a completely manual process of informing community members through websites or social media posts and asking them for information about animals in the area.

There is a start-up abroad called FindMyPet that uses reporting and instant notification system that is designed to find lost animals.

We will analyze these competitors and compare them with our own Petfind platform.

2.3.1 Social media groups

In Estonia the most popular groups are in the social media platform Facebook. There are lost and found groups for pets as well as specified groups for every big city. Area specific Facebook groups (for example "Muuga rahvas") get posts about a variety of topics with lost animals being just one of them. There is a lot of traffic in Facebook groups, because

there is no initial setup and it can be accessed from mobile or desktop. These groups are however geographically limited and invite-only meaning information from people finding a pet might never reach the person who lost the animal.

The most popular social media group in Estonia is a Facebook group called "Kadunud ja leitud loomad ning koduotsivad hoiukodude/varjupaikade loomad".[1] The group was created in 2013 and it has 27.3 thousand users. It has posts about missing and found pets, as well as pets looking for a new home.

The group is very active and has a lot of users, but there are a lot of drawbacks:

- Information overload when trying to find anything in the group. Pictures of lost, found pets and newborns are intertwined making it hard to find specific information and lengthening the search time.
- Can not search specifically by location or identifiers like color, type, breed
- User needs to keep reposting in order to stay at the top of the page, otherwise their post will be forgotten.

Currently, this is one of the biggest competitors, because of the amount of traffic in the group.

2.3.2 Websites

There are two main providers of such a platform in Estonia - Lemmikloomaregister and Loom24.

Lemmikloomaregister is a nationwide voluntary pet identification database in Estonia [2]. Their focus is on contacting owners whose animals have microchips. If a stray or abandoned animal is found, vets will search the animal for microchips and use this site to contact the owner. Users need to contact the admin if their animal has gone missing or search the database by microchip id.

Lemmikloomaregister has a lot of advantages:

- Available 24 hours a day (24 hours online service).
- Find owners of microchipped animals quickly and securely.

Loom24 offers targeted advertising opportunities on a portal that connects all animal lovers in Estonia [3]. On this site clients can register an animal they have seen or search for their

missing pets. At the time of writing the thesis there are 115 posts on the site - 20 found and 95 lost pets. Loom24 has many services available and has a straightforward user interface.

Both website have drawbacks:

- Information overload when trying to find a specific animal.
- User cannot search by a specific location or identifier like color, type which lengthens the search time.
- Finding lost animals is not a priority for them hence they will not improve their current solution at all or if then marginally.

These websites are not big competitors mostly, because the process of finding their pet is not automated and is quite time consuming. Also, there are no mobile alternatives and the websites have outdated user interfaces.

2.3.3 FindMyPet

Find My Pet is a reporting and instant notification system that helps all pet owners find their missing animals. This app is available on both Android & iPhone platforms. [4]

The application uses crowd sourcing to help pet owners find their lost or stolen pets. Users can generate an alert, which will get sent to other members using this app with a 25km geo-fence. If other users capture the pet, they can report they have it and contact owner directly via the app. As the pet moves, and people take more pictures, the GPS coordinates of each photo will enlarge or shrink the search area. "Find My Pet" uses volunteers and donations to keep developing their application.

FindMyPet has a lot of advantages:

- Available on mobile and desktop.
- Makes it possible to see the pets route.

Find My Pet is not a big competitor, because their website does not have user friendly design and it is not available in Estonia. The application has no automatizing and only relies on users to report pets.

3. Functional requirements

Based on the analysis of competitors and our vision and research, we developed functional requirements for the owner and operator views. For the initial visualisation of the requirements, we designed the owner and operator views using Figma. See Appendix 2 for wireframe of the initial vision of the application 7.

The user stories outlined below were provided to us by our client. After the entire team had thoroughly acquainted themselves with the stories we discussed them with the client. The user stories were well written and provided us with a lot of room for different implementations with varying levels of technical difficulty. As such the user stories themselves did not need to be altered but implementation was discussed with the client. For example, regarding the identification of the users real life identity being done by the administrator manually.

3.1 User roles

The system describes the following roles:

- Pet owner - A user registered in the system
- Owner of a missing pet - A user registered in the system
- An individual in the residential area - A user who is not registered in the system and who has access to limited pages.
- Pet identification system operator - A user who has the authority to manage the entire application (administrator).

3.2 User authentication

1. The user logs in to the application or contacts the administrator, who will create a new user.
2. The authenticated user's details are stored in the browser cache and on the page. User has to re-authenticate every 12 hours after logging in.
3. The user can log out - the browser cache data is deleted.
4. User's ID must be unique.

3.3 Role 1: Pet owner

Pet owner required activities:

1. Provide photos of the pet to the identification database so that if the pet is found wandering on its own, owner will be informed. Data to be submitted includes: photo of the pet, the pet's species and breed, the pet's name, a description of the pet, pet's home address, owner's name, owner's telephone number, owner's e-mail address. Consent to the system being able to process user data.
2. If needed ask the pet identification operator for information or, if possible, ask directly the database if their pet has been seen and/or identified.
3. Wants to be informed if their pet is seen on its own.
4. Wants to delete a pet from the database.

3.3.1 Submitting photos to the pet database

Prerequisite: existence of a pet database.

Chain of events:

1. User identification and/or creation of a user ID.
2. Entering the pet owner's details: name, phone, e-mail.
3. Enter pet details and photo: species, breed, name, description, photo, home address and, if available, the chip number.
4. Consent to the processing of the data.

Consequence: user details successfully entered, pet details successfully entered. Note: One person can have more than one pet and they can be in different locations.

3.3.2 Asking for pet information

Prerequisite: existence of a pet database.

Chain of events:

1. User identification (identified access).
2. Entering the pet owner's details: name, phone, e-mail.
3. Enter pet ID and search period.

4. The system will provide a response when this or a similar animal has been seen.

Consequence: the user has received information whether the system has seen him or a similar animal.

3.3.3 Getting information about a lost pet

Prerequisite: existence of a pet database.

Chain of events:

1. The system finds a pet in the database based on an incoming image or video.
2. The system sends a message to the user and operator indicating the location and time of the animal.
3. The user checks the image and reports/gives feedback if it was his/her pet or not.

Consequence: the user has received information that his animal or similar animal has been seen. The system receives feedback on whether the decision was correct (should also receive feedback on how similar the animal actually was).

3.3.4 Deleting a pet from the database

Prerequisite: existence of a pet database.

Chain of events:

1. User identification.
2. Entering the pet's ID and a request to be removed from the database.
3. The system deletes the animal from the database.

3.4 Role 2: Owner of a missing pet

Required activities for an owner of a missing pet:

1. If pet is not yet in the database, report the pet missing and provide photos of their pet to the identification database for identification purposes in order to be notified if pet is found missing. Data to be submitted: Photo(s) of the pet, pet species and breed, pet name, pet description, other relevant information, pet's home address, owner's

name, owner's telephone number, owner's e-mail address. Consent to the system to process data.

2. If pet is already in the database, report the pet missing and, on request add additional photos of their pet in the identification database in order to be notified if his pet is found missing. Data to be submitted: Pet photo(s), pet species and breed, pet name, pet description, other relevant information, pet's home address, owner's name, owner's telephone number, owner's e-mail address. Consent to the system to process data.
3. To request information from the pet identification operator or, if possible, directly from the pet owner. If requested, to ask the pet owner's pet identification operator whether his pet has been seen and/or identified.
4. To be informed if pet is sighted.
5. Wishes to be notified when an animal is found so that it can be removed from the wanted/missing list.

3.4.1 Reporting a lost pet (pet is not in database)

Prerequisite: existence of a pet database.

Chain of events:

1. User identification and/or creation of a user ID.
2. Entering the pet owner's details: name, phone, e-mail.
3. Enter pet details and photo: species, breed, name, description, photo, home address and, if available, the chip number.
4. Marking an animal as lost, entering the time of loss.
5. Consent to the processing of the data

Consequence: user details successfully entered, pet details successfully entered, the animal is marked as lost.

3.4.2 Reporting a lost pet (pet is in database)

Prerequisite: existence of a pet database.

Chain of events:

1. User identification and/or creation of a user ID.

2. Marking an animal as lost, entering the time of loss.

Consequence: the animal is marked as lost.

3.4.3 Request for missing pet movements

Prerequisite: existence of a pet database; the animal has been reported missing.

Consequence: the user has received information that a missing animal or similar animal has been sighted. System gets feedback on whether the decision was correct (should also get feedback on how similar the animal actually was).

3.4.4 Getting information about a lost pet

Prerequisite: existence of a pet database; the animal has been reported missing.

Chain of events:

1. The system finds the pet in the database based on the received image or video.
2. The system sends a message to the user and the operator with the animal's location and date and time with a note that the animal has been marked as missing.
3. The user checks the image and provides feedback on whether or not it was the lost pet.

Consequence: the user has received information that a missing animal or similar animal has been sighted. System gets feedback on whether the decision was correct (should also get feedback on how similar the animal actually was).

3.4.5 Reporting animal as found

Prerequisite: existence of a pet database; the animal has been reported missing.

Chain of events:

1. The user sends an e-mail about finding a pet to the operator.
2. The operator enters the data.

Consequence: the animal is marked as found and the time of finding the animal is entered.

3.5 Role 3: Individual in residential area

An individual moving within the residential area wishes to do the following activities:

1. Check whether his/her data and photos/videos of him/her have been recorded and how they have been recorded.

3.5.1 Verification of the correctness of the processing of personal data

Prerequisite: existence of a pet database.

Chain of events:

1. The individual sends his/her name by e-mail to the operator and is asked to provide an overview of what data is available. Individual also gives consent to the processing of the data.
2. Identification of the operator.
3. The operator makes a query to the database.
4. The system outputs the personal data, e.g. the owner's details: name, phone, e-mail, etc. the details of the pet associated with owner: species, breed, name, description, photo, home address, and if available, the chip number.
5. The operator sends the data to the person who made the request.

Consequence: the data have been properly issued.

3.6 Role 4: Pet identification system operator

The pet identification system operator wishes to do the following:

1. Have the possibility to add photos and other information to the pet identification database manually or automatically as new camera data is added. Data to be submitted: pet photo(s), pet species and breed, pet name, pet description, other relevant information, pet's home address, owner's name, owner's telephone number, owner's e-mail address.
2. Include information about animals and their disappearance (when, where - information provided by the notifier)
3. Receive a notification if an animal has been seen.
4. Receive a notification when the system has detected a pet and possible matches

(species, animal identifiers, animal recordings, etc.) As well as the possibility to add this animal to the system in order to identify the same (previously unknown) animal at different times and places.

5. Extract data on sightings of specific animals or similar animals in the system.
6. Delete the animal from the database.
7. Retrieve photos, videos and other data on individuals in the system.

3.6.1 Submitting data and photos to the pet database

Prerequisite: existence of a pet database.

Chain of events:

1. User identification and/or creation of a user ID.
2. Entering the pet owner's details: name, phone, e-mail.
3. Enter pet details and photo: species, breed, name, description, photo, home address and, if available, the chip number.
4. Consent to the processing of the data

Consequence: user details successfully entered, pet details successfully entered.

3.6.2 Reporting a lost pet

Prerequisite: existence of a pet database.

Chain of events:

1. User identification and/or creation of a user ID.
2. If the animal is not in the database, enter the owner's details: name, telephone, e-mail, then enter the pet's details and photo: species, breed, name, description, photo, home address and, if available, chip number.
3. Marking an animal as lost, entering the time of loss.
4. Consent to the processing of the data

Consequence: user details successfully entered, pet details successfully entered, the animal is marked as lost.

3.6.3 Getting information about a lost pet

Prerequisite: existence of a pet database.

The chain of events is identical to the chain of events in 3.3.3, if necessary, the animal shall also be reported as missing.

Consequence: the operator has received an indication that an animal or similar animal has been seen. The system receives feedback on whether the decision was correct (should also receive feedback on how similar the animal actually was).

3.6.4 Getting information about a stray pet

Prerequisite: existence of a pet database.

Chain of events:

1. Based on the received image or video, the system detects the animal.
2. The system sends a message to the operator stating the following details about the detected animal: species, location and time.
3. The system provides possible matches (species, animal identification data, animal descriptive data).
4. The operator adds an animal to database if desired to detect the presence of the same animal (previously detected or unidentified) at different times and places.

Consequence: the database has been updated at the choice of the operator.

3.6.5 Operator requests for pets movements

Prerequisite: existence of a pet database.

The chain of events is identical to the chain of events in 3.3.2

Consequence: the operator has received an overview that an animal or similar animal has been seen. The system receives feedback on whether the decision was correct (should also receive feedback on how similar the animal actually was).

3.6.6 Operator deletes a pet from the database

Prerequisite: existence of a pet database; the animal is in the database.

Chain of events:

1. Operator identification.
2. Entering the pet ID and the request for pet to be removed from the database.
3. The system deletes the animal from the database.
4. In case the owner no longer has any animals in the database, it asks if it also wants to delete the personal data or not. According to the operator's decision, deletes the personal data.
5. The operator sends a notification to the user (owner).

Consequence: the animal's data has been deleted from the database. The photos remain in the system to support machine learning algorithm.

3.6.7 Verification of the correctness of the processing of personal data

Prerequisite: existence of a pet database.

The chain of events is identical to the chain of events 3.5.1, but in addition the operator has the possibility to print out all personal data for all persons.

4. Petfind development

In this chapter, we will describe how the development started, what technologies and solutions were used, and how the application was finally completed and made available. The app was designed and developed for desktop and mobile platform.

4.1 Implemented technologies

In this work, programming languages Python 3.9 and C++ are used. The TensorFlow library is used for training the network.

4.1.1 Python and Django

Competence of the team / ease of language

Python is often the first programming language to be learned because of its simplicity. Therefore, it is certainly within the reach of our team, who have completed 3 years of Informatics undergraduate studies at TalTech. In the first semester of these studies, Ago Luberg's course ITI0102 "Introduction to Programming" already covered Python programming. Python has often been considered a disadvantage because of its slow speed, but since we use external hardware accelerated libraries for more complex computations (like machine learning) this is not a problem.

Python's popularity in machine learning.

"PyPI is the most popular software package ecosystem for ML libraries to publish the officially-maintained bindings. Figure 2 shows that 88% (129 out of 146) of the popular ML libraries support an officially-maintained binding in PyPI. Moreover, we noticed that 65% (95 out of 146) of the popular ML libraries use Python as their primary programming language." [5]

Java has a better performance than Python but Python requires lesser code and can compile even when there are bugs in your code. One of the main reasons why Python is widely used is its ease to adopt for people who are beginners. It is also more suited for quick

prototyping. In short, Python is preferred over Java for AI projects including developer productivity, language flexibility, library support, community support, and ease of learning. [6]

Django

Django was chosen because it enables rapid development of the application, as it already includes all the essentials needed, such as the Admin UI, authentication, security etc. Django proved essential for the project development. Without it we would have not been able to deliver such a comprehensive, fully functional, secure, future proof and maintainable application. The ability to modify all database objects in a user friendly manner within the Admin UI is an amazing out-of-the-box feature. It allowed us to provide fully fledged administrating capabilities from day one. The ability to empower the administrator and ensure that during development no new data is added that remains off limits to the admin user was a key requirement that was met with the aid of Django.

While Flask is highly suitable for small projects that necessitate experimentation we chose Django, because it stands out as one of the best frameworks for building sophisticated applications.

4.1.2 Bootstrap

Bootstrap is a front-end framework used to create dynamic and responsive web applications. It is one of the most popular frameworks among designers. Bootstrap consists of an HTML- and CSS-based structural design template that allows easy creation of web development interfaces using pre-built components. By updating CSS, it is possible to quickly integrate modern trends and create stunning websites. Bootstrap is perfect for the team as it's easy to use and speeds up our development process with ready-to-use templates and teasers. In addition, thanks to Bootstrap's popularity and community in the development world, solutions can be found quickly when problems arise. Bootstrap is compatible with all major browsers and is customizable to the team's needs.

We chose Bootstrap over React, because we did not want to increase the code complexity. Bootstrap has better documentation and is easier to debug than React.

4.1.3 TensorFlow and Keras

TensorFlow is a low-level machine learning engine, while Keras is a high-level deep learning API for easy deployment and computation of neural networks and runs on top of TensorFlow. Both of these frameworks were used in the development, as they are related to each other and many functions and methods are taken from both frameworks.

We chose TensorFlow over PyTorch, because TensorFlow offers better visualization, which allows us to debug better and track the training process. PyTorch, however, provides only limited visualization.

4.2 UX and UI design

We applied User Interface design standards and laws when building our user interface. We used the following laws and effects to make the site more usable: Jakob's Law, Hick's Law and Aesthetic-Usability Effect. Jakob's Law was applied to make our site familiar and easy to use by using standard elements, common patterns and visual cues for choices available to the user. Hick's Law was applied by minimizing choices given to users and simplifying the user interface. The Aesthetic-Usability Effect was used to make the Users perceive the design and more usable and pleasant.

We decided not to use Neumorphism in our user interface, because there is a very small margin of colors and contrast. Also, it does not have a user friendly design to visually impaired users, because Neumorphism uses very subtle contrast.

Visual hierarchy:

- We used 4 different styles of Bootstrap buttons: primary, secondary, danger, light to highlight certain actions. Primary actions are obvious: "Logi sisse" button has solid, high contrast color. Secondary actions are clear but not prominent: "Minu loomad" button has lower contrast background color. Tertiary actions are discoverable but unobtrusive: "Meist", "KKK" buttons are less noticeable.
- We used Helvetica typography font throughout the user interface. We chose this font, because we wanted a popular font that looks natural and has a lot of different weights.
- For color palette we used Bootstraps original colors. Our primary colors were: blue, white, black and yellow.
- Layouts and grids: throughout the user interface we used asymmetric grids, also

called broken grids, that do not have any center line or point. We also used responsive grid that allowed our grid layout to be scalable with different screen sizes.

4.2.1 Sitemap

A sitemap is a visual representation of a site's structure. We decided to create a sitemap to get a full picture of our system.

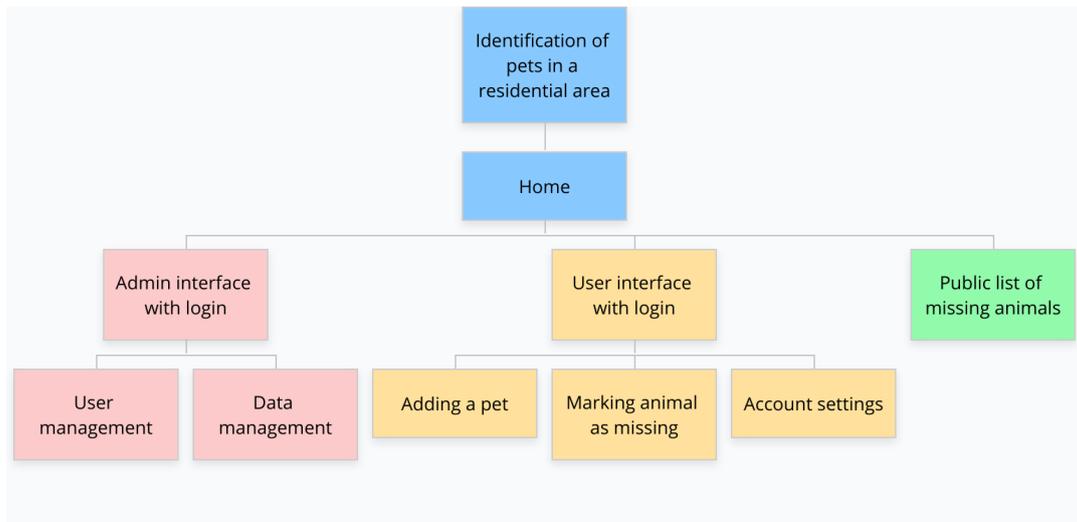


Figure 1. Website sitemap.

4.2.2 Personas

Personas are fictional characters, which we created based upon research in order to represent the different user types that might use our platform. We decided to use personas to prevent self-referential design as well as prevent designing for a elastic user.

First persona is Desire, who intends to use our platform to have a system in place for when her pet goes missing. Full persona is seen in appendix 2.

Second person is Zoe, who intends to use our platform to give her pet more freedom whilst being able to find her in case pet goes missing. Full persona is seen in appendix 2.

4.2.3 User flow

A user flow is a diagram that shows the path our users will take through our websites. Our user flow has two main tasks user wants to execute: user wants to find out information about lost pet and user wants to add pet to the system.

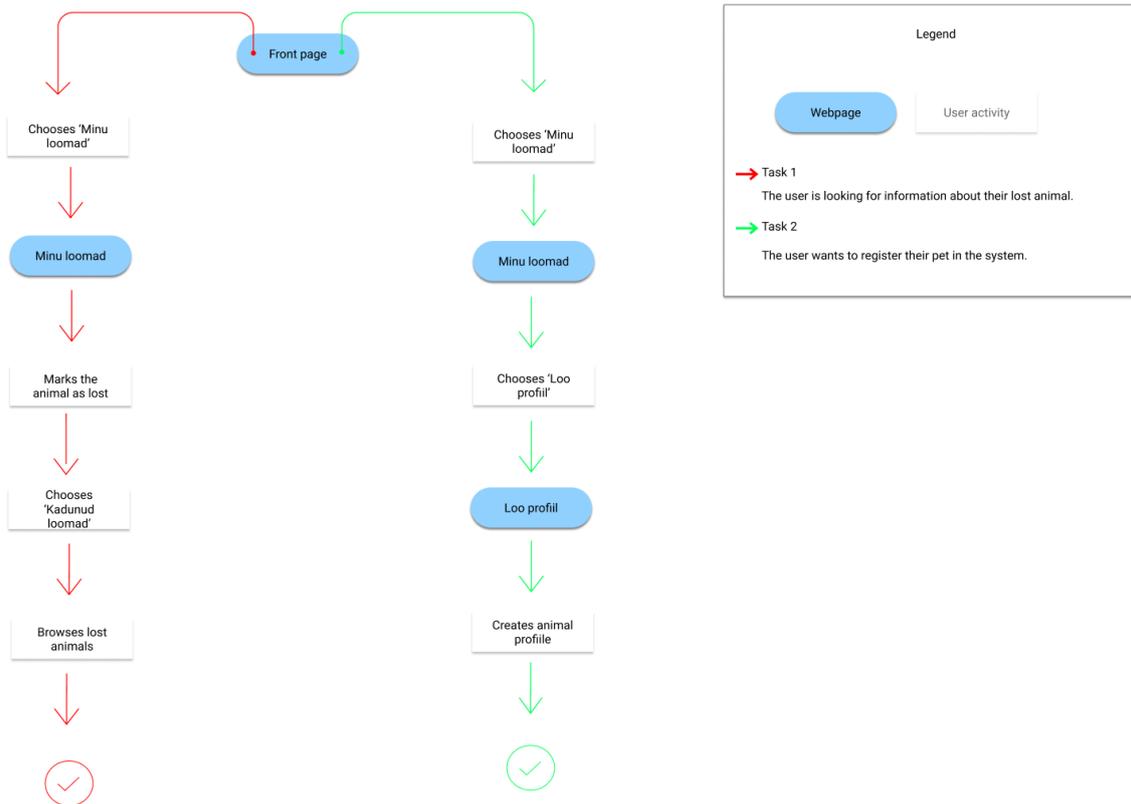


Figure 2. Process chart

4.3 Machine learning

Since none of the team has much experience in working with neural networks and especially with object detection, it was decided to take a pretrained neural network that was designed to solve similar tasks such as object detection. A method has been chosen to implement object detection and training, which is recommended in case of a lack of images for training, lack of a large margin of time and computing power. Namely apply transfer learning / fine-tune a pretrained neural network from pretrained model selection (model zoo). In this case, we can already get good results for our own classes without having a huge dataset quickly. This method was tested in the bachelor's thesis "Object detection in sports: TensorFlow Object Detection API case study", in which this solution showed relatively good results on a small set of data [7].

The choice of the model from the TensorFlow Object Detection API fell due to the fact that it is well documented and has a large selection of different models for our object recognition task. Other model zoos such as PyTorch also offer models of different architectures, but mainly for an image classification task that involves predicting the class of a single object in an image, whereas we needed to classify one or more objects in an image. The model

architectures for object detection included in the PyTorch Model Zoo are also included in the TensorFlow 2 Detection Model Zoo.

4.3.1 Object detection

TensorFlow Object Detection API provides a collection of pretrained object detection models that were trained on the COCO 2017 dataset. All these NN models have different architectures and therefore provide different performance, but there is a trade-off between execution speed and bounding box placement accuracy. Pretrained models can be useful for out-of-the-box inference for detection of categories already included in that dataset, or for initializing your own models when training on your own dataset.

One of the advantages of using this particular library is that we can choose a neural network for our tasks, whether it is detecting objects in a video or working only with pictures. Also, we can easily test different architectures to evaluate which is better, even though this has already been done by the developers and the results have been given in a table [8], from where we could roughly make a choice of model architecture based on speed and accuracy.

Before fine-tuning the model, we tested several architectures on images from camera traps. All of them showed an unsatisfactory result, so we chose the model according to the table.

In this project, the "CenterNet Resnet101 V1 FPN 512x512" model was used as it combines good performance in both speed and accuracy. The speed is 34 ms and the COCO mAP is 34.2.

4.3.2 Data collection

In our case, we did not have a large dataset with any bounding box annotation, so all the data collected had to be labeled in appropriate format. The tool used to create bounding box annotations is called "labelImg tool" [9]. Annotations are saved in VOC Pascal format (.xml format).

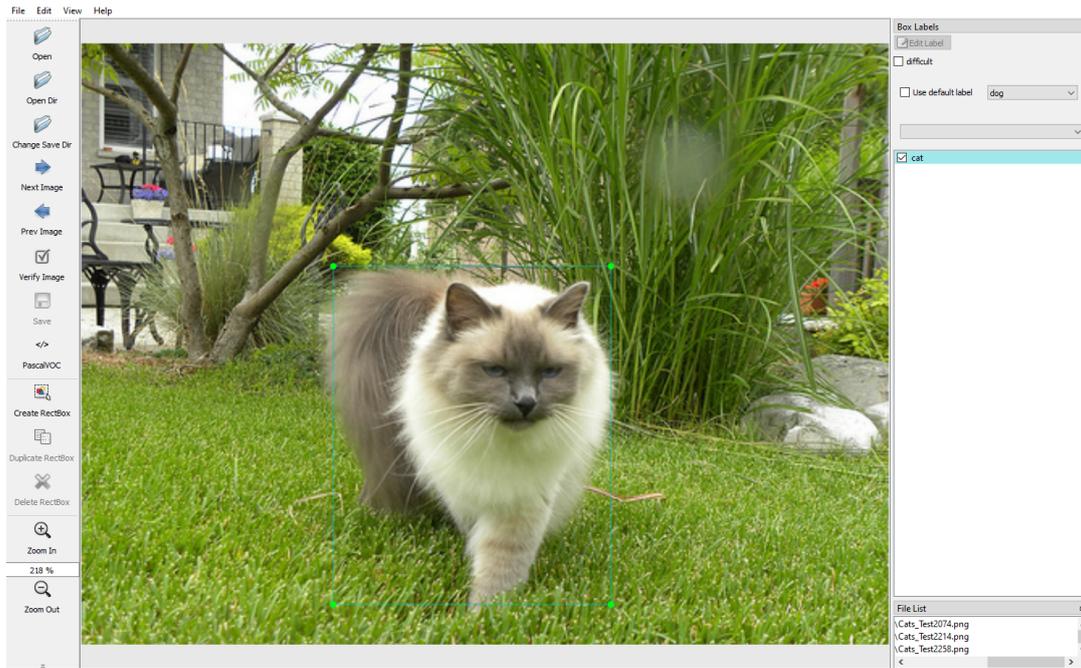


Figure 3. Training/testing data image being labeled using labelImg.

The data was collected from the following sources:

- Provided directly from the camera trap by our supervisor
- Partial images from the dataset of cats and dogs [10]. These images had to be relabeled manually, as they were not suitable for our task (animal faces were labeled in the dataset, not the whole object).
- Separate camera trap images from the Internet for model evaluation

After annotating our image dataset it was divided in two parts. It is a general convention to use only part of it for training, and the rest is used for evaluation purposes. The ratio used was 9:1, i.e. 90% of the images are used for training and the remaining 10% are used for testing. However, after it was decided to add data for testing to make the estimate more plausible. The train/test set is available in our repository.

Class name	Train	Test
cat	196	56
dog	216	50
bird	25	5
beaver	1	0
wild goat	2	2
ferret	1	1
wild boar	1	0
other animal	2	1
Total	444	115

Table 1. Number of labeled images with classes used in dataset for model training

There is also a support for such classes as elk, fox, rabbit, bear, wolf, raccoon, otter and lynx. However, the images with these classes were not included in the dataset.

These classes were agreed with the client/supervisor. There was also discussion about focusing on dog and cat breed recognition, but this was not a viable solution for a number of reasons. Firstly, existing datasets are intended for a different type of task, namely an image classification task, and even if we took this dataset, we would have to manually label the entire dataset for our object detection task. Secondly, collecting animal breeds from other image sources takes a significant amount of time. Lastly, we would need to enlist the help of a dog breed expert, because it's very hard to label each image without much knowledge of breeds. This potential solution was left for future improvement of the existing system.

We focused largely on model improvements by means of fine-tuning instead of improving the quality of the collected data. Data collection is handled by the client with their cameras and as such is outside of our control. We could have experimented with pre-model image processing techniques, but decided against it because of our limited experience and fear that we may create artificial patterns in the data which the neural network may pick up on. The end result of which would be a convoluted and incredibly difficult to debug setup.

4.3.3 Model training

The pre-trained "CenterNet Resnet101 V1 FPN 512x512" model was fine-tuned for our dataset using manually labeled images and Jupiter Notebook Training_Model.ipynb file that is included in our project repository. The instructions are based on official tutorial [11]. The training was stopped after 50000 timesteps.

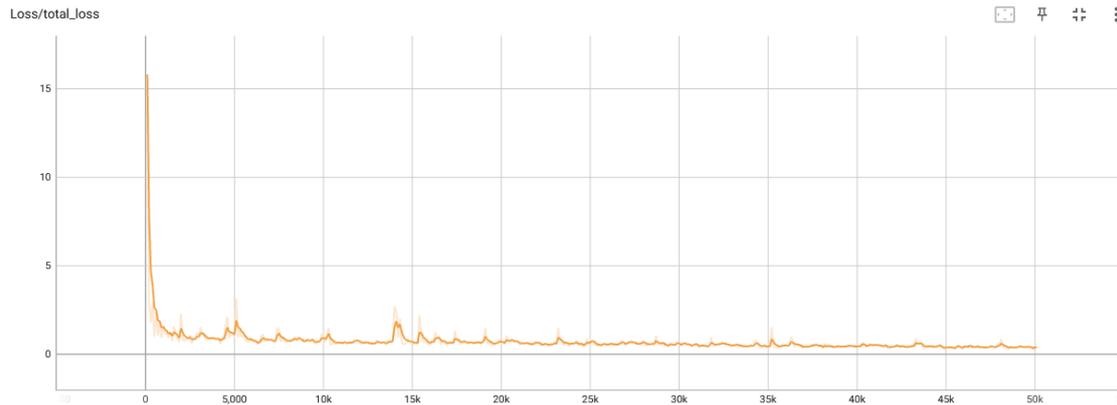


Figure 4. Development of loss (smoothing = 0.6).

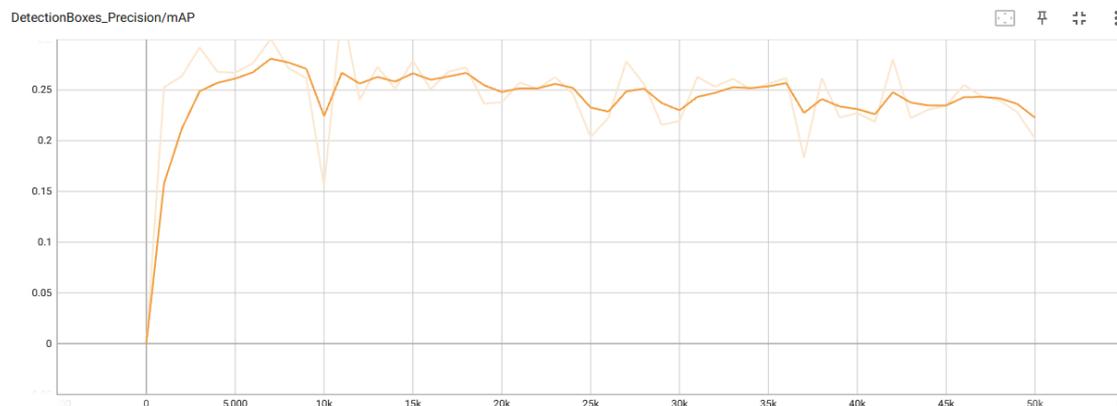


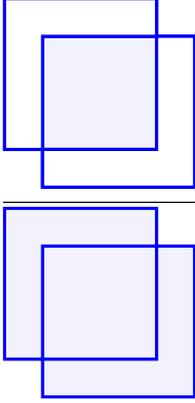
Figure 5. Development of mAP (smoothing = 0.6).

4.3.4 Model evaluation

The TensorFlow Object Detection API uses "COCO detection metrics" by default for model evaluation. The following metrics were obtained after validation on the test dataset.

Important definitions

- Intersection Over Union (IoU):** a measure that evaluates the overlap between two bounding boxes. It requires a ground truth bounding box and a predicted bounding box. By applying the IoU we can tell if a detection is valid (True Positive) or not (False Positive).

$$IoU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} = \frac{\text{area of overlap}}{\text{area of union}} \quad (4.1)$$


The equation (4.1) states that IoU is given by the overlapping area between the predicted bounding box (B_p) and the ground truth bounding box (B_{gt}) divided by the area of union between them.

- **True Positive(TP):** A correct detection. Detection with $IoU \geq \text{threshold}$
- **False Positive (FP):** A wrong detection. Detection with $IoU < \text{threshold}$
- **False Negative(FN):** A ground truth not detected
- **Precision:**

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}} \quad (4.2)$$

Precision (4.2) describes how relevant the detection results are.

- **Recall:**

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truths}} \quad (4.3)$$

Recall (4.3) describes the percentage of relevant objects that are detected.

Metrics

- **Precision x Recall curve:** The Precision x Recall curve shows the trade-off between precision and recall for different thresholds. The curve is formed by sorting all the predicted bounding boxes (over all images in the evaluation set) assigned to an object class by their confidence rating, and then computing a recall and precision value for each prediction.
- **Average Precision(AP):** It is calculated using area under the curve of the Precision x Recall curve. AP is the precision averaged over all recall values in range from 0 to 1.
- **Mean Average Precision(mAP):** The mAP score is calculated by taking the mean

AP over all classes **and/or** over all IoU thresholds. [12]

Eval metrics name	IoU	area	max detections per image	value
Average Precision (AP)	0.50:0.95	all	100	0.202
Average Precision (AP)	0.50	all	100	0.293
Average Precision (AP)	0.75	all	100	0.257
Average Precision (AP)	0.50:0.95	small	100	-1.000
Average Precision (AP)	0.50:0.95	medium	100	0.184
Average Precision (AP)	0.50:0.95	large	100	0.201
Average Recall (AR)	0.50:0.95	all	1	0.275
Average Recall (AR)	0.50	all	10	0.331
Average Recall (AR)	0.75	all	100	0.334
Average Recall (AR)	0.50:0.95	small	100	-1.000
Average Recall (AR)	0.50:0.95	medium	100	0.440
Average Recall (AR)	0.50:0.95	large	100	0.301

Table 2. Evaluation results run with TensorFlow 2 Object Detection API

The resulting mAP value of 0.202 can be considered not so bad, given the amount of data for training, training time and the dataset given for evaluation. The evaluation set consisted of camera trap images as well as simpler images of cats and dogs.



Figure 6. Example of model evaluation based on camera trap images (1). The left image is the model's prediction, the right image is the ground truth.



Figure 7. Example of model evaluation based on camera trap images (2). The left image is the model's prediction, the right image is the ground truth.

Moreover, we did not set ourselves the task of training the model very well, but on the contrary, to create an environment and all the necessary tools to improve the existing model.

4.3.5 A model that develops over time

The concept of a constantly learning neural network is quite difficult to implement, since the learning process involves the involvement of a person in any case. The so-called "Supervised learning", because there are a huge number of factors that affect the effectiveness and efficiency of training [13]. Putting it on automation or somehow training without human intervention will be quite difficult, because it is necessary to create this system from scratch, plus, besides, this is not implied by the TensorFlow library itself.

To improve the accuracy and/or number of detection classes of our and future neural networks, we need to either train on a huge dataset or train longer. That is why we came to the conclusion that it would be a good idea to implement an image labeling tool in the application itself in order to collect data for the next training instead of using external tools.

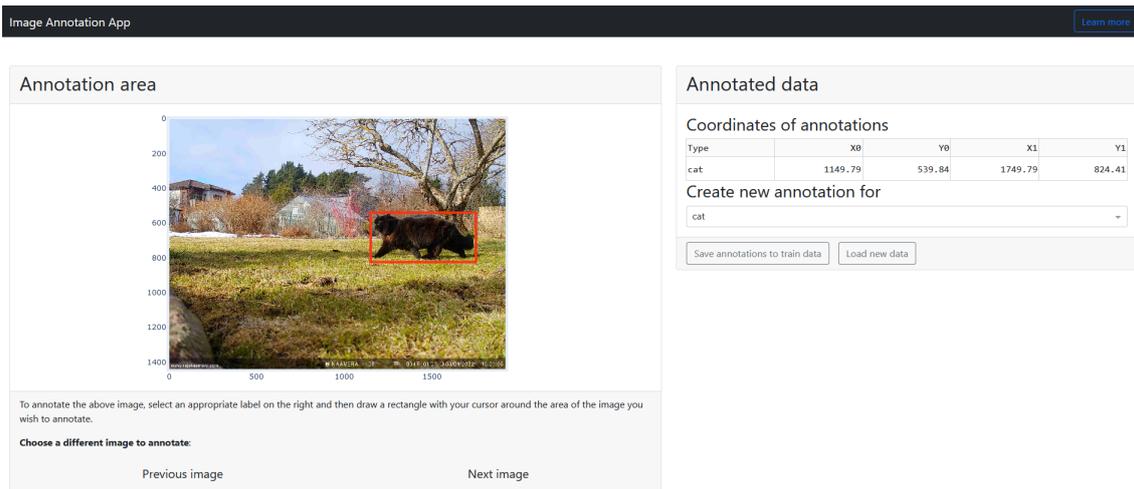


Figure 8. Image labeling tool in the application

According to the idea, we use our custom model to detect objects (at this stage, no matter how accurate these predictions are) and correct incorrect predictions. Each picture in which incorrect or inaccurate predictions were noticed by the administrator is marked as incorrect and sent for manual labeling, thereby collecting data for the next training. Thus, we will need to involve the person directly in this process, which is a disadvantage, but hopefully a small one.

The cycle of data acquisition, data labeling and model training in Google Colaboratory (Colab) is shown clearly in the instruction video "[Petfind] Colab Detection Model Training" [14]. Colab was chosen as it is a free cloud service which allows anyone to write and execute Python code through a browser (with the support of Jupyter Notebooks) and offers GPU and CPU processing power especially well suited for machine learning, data analysis, and training jobs.

4.4 Limitations

Since a mailbox was used to receive images from the trap cameras (each photo is attached in a separate letter), we may have problems with the fact that the email provider limits the capacity of the mailbox (the mailbox allocated to us for testing has a disk space limit of 8GB). However, the problem can only arise if the mail is not cleaned in a certain period of time. The usual photo size is just over 300KB, so the mailbox can hold about 25,000 images before it needs to be cleaned.

4.5 Deployment

The original goal was to provide by the end of the project a working software solution hosted on the clients infrastructure. The infrastructure the client had procured however did not allow this as it was a shared hosting service which did not allow for the installation of the external machine learning packages that are a prerequisite for our project.

As such the current deployment is temporary and simplistic. It has however had no issues despite both user testing and being available publicly for multiple weeks. The simple architecture diagram of the proof of concept deployment used for testing can be seen below.

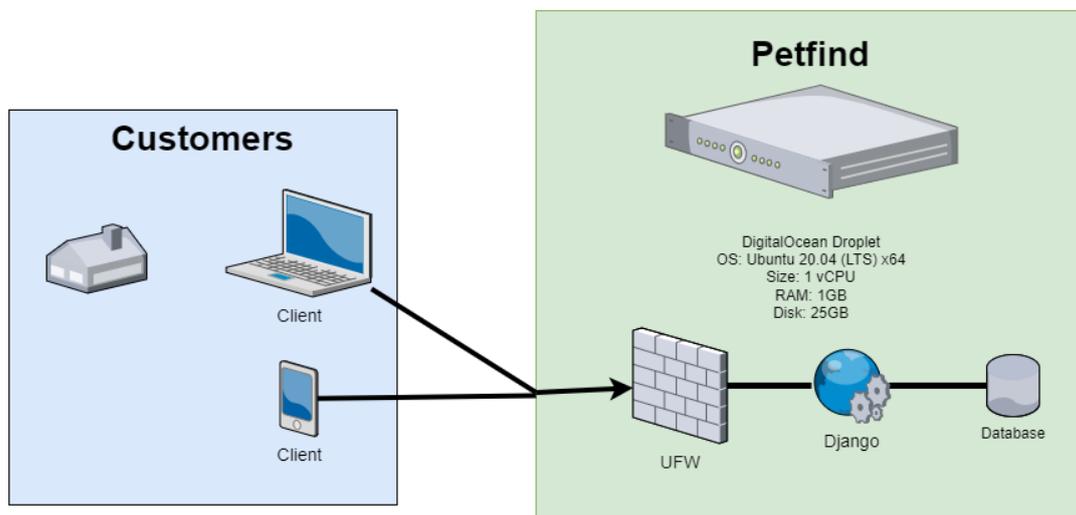


Figure 9. Architecture diagram of the proof of concept deployment used for testing

5. Results and discussion

This section analyses a quantitative validation of the fine-tuned model, as well as the analysis of the final product and the assessment given by our supervisor on the basis of the tests carried out in the real environment.

5.1 Quantitative validation

It is extremely difficult to evaluate our solution quantitatively, since how accurate our neural network model trained for animal recognition can only be tested and analyzed with a huge sample of data, various weather conditions, lighting etc. A rough estimate of our model has been made and the results are recorded in the table Evaluation results run with TensorFlow 2 Object Detection API.

As a result of testing on the provided data set, it can be assumed that it is likely that animals should be provided from a large number of angles, since the animals do not necessarily get on camera trap completely and not necessarily from a good angle. It is necessary to include as many views as possible in the training set (from the sides, back, front, etc.) to improve the accuracy of the model. Also important were the times of day. At night, it was usually harder for the model to recognize objects, probably because the dataset consisted mainly of images with good lighting.

It can also be assumed that fine-tuning for our classes came in handy and it was the right decision to simply not use this pre-trained model out of the box. Since the data provided consisted of classes of animals from different angles, the model was more accurate with such predictions.

5.1.1 Differentiate between a user's pet and a similar animal seen on cameras

The process for a user to receive notifications about a pet in the database is as follows. First of all, the pet must be marked as lost OR marked that the user wants to receive notifications. Next, the model makes predictions from the received photos. If the same species of animal was identified in the photo as the user's pet, then an sightings pair is created and offered to

the user for comparison/feedback. This process is limited by the accuracy of our model, so there is a second way that the user can view images from trap cameras and animals detected on them - through "recently seen animals" on the home page.

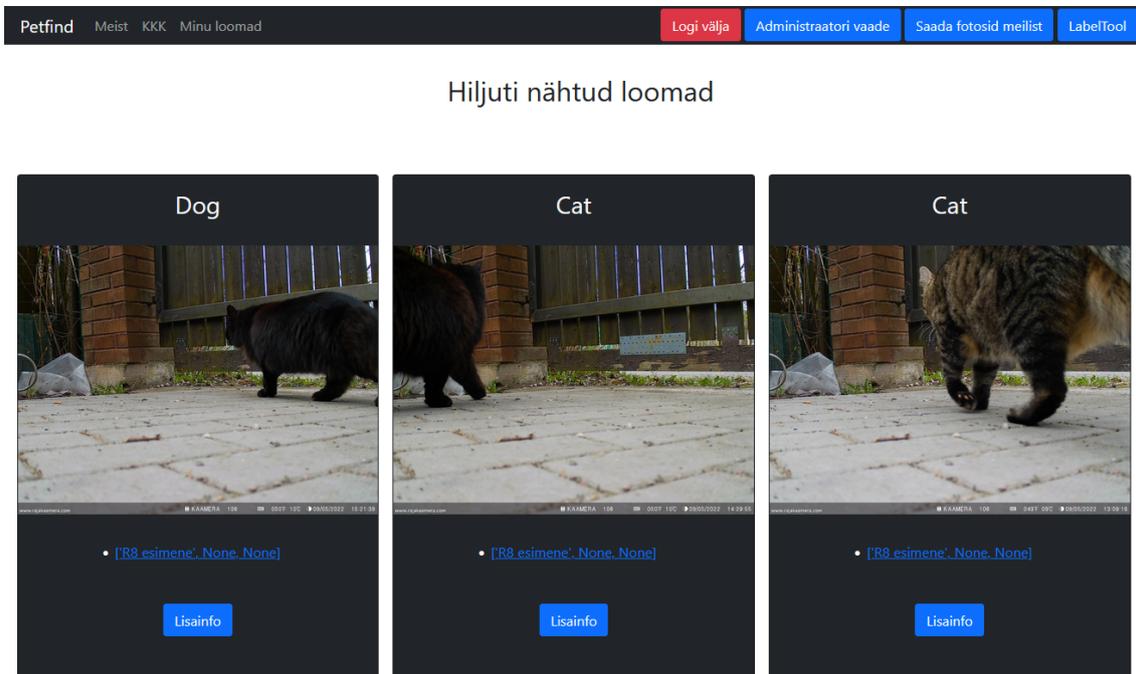


Figure 10. Example of a page "Recently seen animals"

At the moment there are no mechanisms that could offer a comparison of objects in images and, based on similarity, offer the best matches. Thus, if a dog is lost and the animal was predicted to be a dog in the camera trap images, it will be suggested to the user as a similar animal. Thus, the correctness of the proposed pair for comparison to the user depends only on the accuracy of the model's predictions.

Pildid, millel võib olla minu kadunud loom



Figure 11. Example of a page "Images possibly showing my lost animal"

In addition, the user can give feedback on each pair, which is recorded in the database for two parameters: the same animal, the same species. However, the results of this feedback have not yet been used anywhere. Recording of this data is however the key to future improvements that can provide new functionality to identify specific animals using a new machine learning system trained using this data.

5.2 Privacy and confidentiality requirements

Although our prototype was designed with these requirements in mind, we were unable to fully implement extensive protective measure to ensure the privacy of people who might accidentally be caught in camera traps. We assume that if there is an animal in the frame along with a person, then this image will end up on the main page.

The blurring of people's faces for example was not implemented, since this involves another neural network specially made for this task which fell outside our scope of development due to time constraints.

We make the assumption that humans will not often be seen by cameras (because they will be installed in remote locations) or recognized by the neural network (because the neural network is trained to recognize only animals, mainly cats and dogs), and therefore our solution will comply with privacy and confidentiality requirements. This was communicated clearly to the client who will take great care to install cameras only in areas where this risk is minimal.

5.3 Feedback after testing from the supervisor

Included in Appendix 8 - Feedback from the supervisor.

5.4 Key lessons

The key technical lesson that was strongly reinforced for us was that optimizing too early and over engineering software-technical solutions can lead to their downfall. By using off-the-shelf popular frameworks we avoided many pain points that are associated with cutting-edge, poorly documented and immature software. We provided the client with a working prototype as early as possible which allowed us to discover many flaws that we would never have noticed ourselves.

The utilization of Django in particular enabled us to incredibly rapidly add new functionality and deal with shifting demands and provides for room for future growth both in terms of complexity and number of users.

During our preliminary research we found articles that had tackled animal level similarities but we did not anticipate how difficult they would be to implement in a real world setting with actual live data. This was why we failed to provide animal level similarity and instead resorted to animal type level similarity identification.

The process improvement value provided by the short feedback loops of the Agile methodology was even more strongly reinforced to us then during previous projects. Frequent meetings and good relations with the client along with clear boundaries of responsibility and due dates enabled us to stay on schedule and address potential shortcoming early enough to take action to avoid roadblocks later on.

In future projects we will pay more attention to functionality instead of design. While the focus on a functional web application with a user interface was a requirement provided by the client and we should have put some effort towards it and its design we focused on it too much instead of ensuring the user flow and functionality was as good as it could possibly be.

6. Proposed future improvements

As part of the thesis, the Petfind was completed, but this is not the end of the platform development. During the course of the work, functionalities were mentioned that should definitely be added. The desired additions came from user feedback, testing and competitor analysis.

Here are the primary annexes that could be developed in the future:

- Identification of specific individual animals using the large dataset gathered by the current iteration of the application.
- Alternative data entry methods such as FTP and user submissions in addition to manual entry and importing from an email account that are currently implemented.
- Improved packaging of the application to enable more rapid setup
- On premise self-hosted setup option
- Federated collection of on-prem instances joined by a industry-standard standard to enable strict regulatory compliance without compromising on information sharing between jurisdictions with potentially differing data privacy regulations.
- Better visualization of geographic data with a maps integration
- Fully self-service account creation using a integration with a third-party identity provider such as RIA (Riigi Infosüsteemide Asutus) provided TARA (Riigi autentimisteenus) service.
- Notification integration with various different communication channels utilizing Ap-prise or a similar system to enable users and the operator to get instant notifications.
- Cloud-native redesign to enable scaling to a planet-scale user base
- Improved systems for managing the tasks that the operator needs to accomplish - checklists or a to-do board with open items up for review. This could be achieved as an integration with a service desk software such as Cherwell that is widely used by IT support personnel.

7. Summary

The aim of this work was to implement an application with which it would be possible to track animals captured from trap cameras with the integration of artificial intelligence.

During the course of the work pre-existing competitors were examined and compared with our planned application. As this application will be destined in the future for real world use our client took concerns of data privacy extremely seriously and these concerns were carefully considered in our process.

The solution was validated by the client by installing trap cameras in some area, reading the images sent from the cameras from e-mail and by testing user roles.

As a result of the development and testing, all functional requirements were met or partially met.

Given that this is a prototype, there are definitely plans for further development and improvement of this application along with plans to utilize it in real world use.

During the writing of the bachelor's thesis, we got acquainted with several new technologies. Bachelor's work turned out to be a larger learning opportunity than we could have imagined. We are satisfied with the finished application and that the development of the application has been very positive in terms of our personal development.

Bibliography

- [1] "*Kadunud ja leitud loomad ning koduotsivad hoiukodude/varjupaikade loomad group*". URL: <https://www.facebook.com/groups/kadunudjaleitudloomad> (visited on 04/26/2022).
- [2] "*Eesti Lemmikloomade Register website*". URL: <http://lemmikloomaregister.ee/kadunud-loomad/> (visited on 04/26/2022).
- [3] "*Loom24 website*". URL: <https://loom24.ee/beta/type/kaotatud-leitud> (visited on 04/26/2022).
- [4] "*FindMyPet website*". URL: <https://findmypet.help/> (visited on 04/30/2022).
- [5] Hao Li and Cor-Paul Bezemer. *Studying Popular Open Source Machine Learning Libraries and Their Cross-Ecosystem Bindings*. 2022. DOI: 10.48550/ARXIV.2201.07201. URL: <https://arxiv.org/abs/2201.07201>.
- [6] *The best AI Programming Languages - Java vs Python*. en-GB. July 2020. URL: <https://huddle.eurostarsoftwaretesting.com/the-best-ai-programming-languages-java-vs-python/> (visited on 05/30/2022).
- [7] Pirkko Mustamo. "Object detection in sports : TensorFlow Object Detection API case study". In: 2018. (Visited on 05/04/2022).
- [8] *TensorFlow 2 Detection Model Zoo*. Apr. 27, 2022. URL: https://github.com/tensorflow/models/blob/2a57771d2295ba10a0287b3360689ead80539399research/object_detection/g3doc/tf2_detection_zoo.md (visited on 04/27/2022).
- [9] darrenl. *LabelImg*. original-date: 2015-09-17T01:33:59Z. Apr. 27, 2022. URL: <https://github.com/tzutalin/labelImg> (visited on 02/26/2022).
- [10] *Dog and Cat Detection*. URL: <https://www.kaggle.com/andrewmvd/dog-and-cat-detection> (visited on 04/27/2022).
- [11] *Training Custom Object Detector — TensorFlow 2 Object Detection API tutorial documentation*. URL: <https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html> (visited on 05/02/2022).

- [12] Vijay Dubey. *Evaluation Metrics for Object detection algorithms*. Medium. Oct. 6, 2020. URL: <https://medium.com/@vijayshankerdubey550/evaluation-metrics-for-object-detection-algorithms-b0d6489879f3> (visited on 05/02/2022).
- [13] *Supervised learning*. In: *Wikipedia*. Page Version ID: 1073728937. Feb. 24, 2022. URL: https://en.wikipedia.org/w/index.php?title=Supervised_learning&oldid=1073728937 (visited on 03/24/2022).
- [14] Petfind. *[Petfind] Colab Detection Model Training*. Apr. 29, 2022. URL: <https://youtu.be/W2T9j1GIr0U> (visited on 05/02/2022).

Appendices

Appendix 1 - Non-exclusive license for reproduction and publication of a graduation thesis ¹

I, Kristjan Variksoo, Artjom Zarva, Veronika Tamm

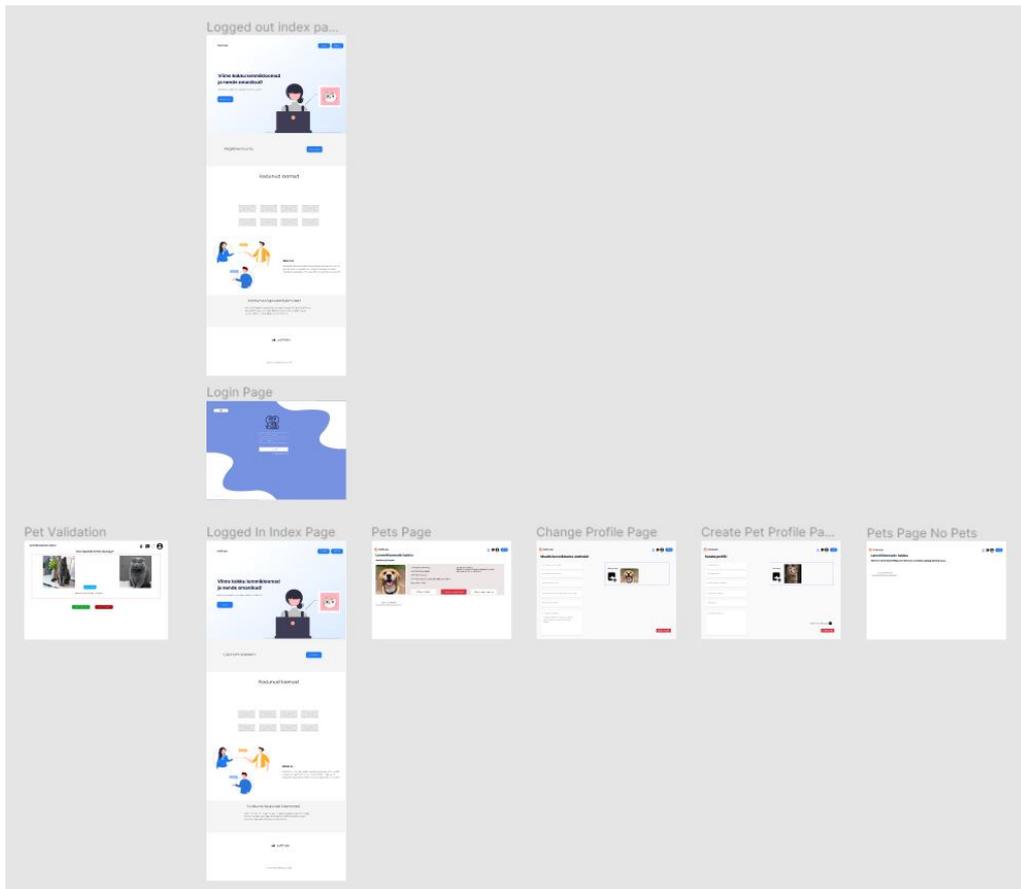
1. grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "Detection of Pets Wandering in a Residential Area", supervised by Raul Savimaa
 - 1.1 to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
 - 1.2 to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

30.05.2022

¹The non-exclusive license is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive license, the non-exclusive license shall not be valid for the period.

Appendix 2 - Website wireframe model

Pet Owner view of wireframe model designed with Figma



Appendix 3 - User Personas

First user persona

Desiree Watson



"Dogs are the most amazing creatures; they give unconditional love."

♂ Female
📅 26
📍 Tallinn
💍 Single
💼 Software developer

Bio

Desiree works at a tech company turning the day. Growing up her parents had a lot of pets, so after moving out Desiree wanted a dog of her own. Now she has been living together with Luna, her dog, for over two years.

She loves her dog and spends a lot of her free time by going on adventures with her after work. However, the amount of work Desiree gets can sometimes keep her away from Luna, who does not like to be inside the house a lot. She would love to keep Luna outside, but worries what might happen if she gets out of the yard. She would love a backup system that would be able to find lost animals if needed.

Most used apps



Behaviors

- Loves her dog and wants only the best for her.
- Uses an app for everything.
- Looking for fast and convenient services.

Goals & needs

- Wants her dog to have the best living conditions.
- Wants to give her dog freedom to go outside in the garden without worrying.
- Needs to know her dog will be safe in the neighborhood.

Frustrations

- Doesn't want to keep her dog on a leash at all times.
- She worries about her dog being alone inside the home all day.
- Can't find a proper platform obtaining an overview of pets moving in an area without supervision.

Second user persona

Zoe Stewart



"I love animals, sometimes more than people, because they don't judge. All they want is love."

- ♀ Female
- 📅 24
- 👤 Tartu
- 💍 Single
- 👔 Personal Trainer

Bio

Zoe is a personal trainer and a mindfulness coach who has a deep love for cats. She volunteers at the local animal shelter where she promotes animals up for adoption.

She loves hanging out with her outdoor cat Zendaja, However sometimes Zendaja roams out of sight for hours on end and makes her owner worried. It would give Zoe a peace of mind if she were able to upload information about her cat in case she goes missing and needs help locating her.

Behaviors

- Loves her cat and wants only the best for her.
- Uses an app for everything.
- Looking for fast and convenient services.

Goals & needs

- Wants her cat to have the best living conditions.
- Wants to give her cat freedom to go outside without worrying.
- Needs to know her cat will be safe in the neighborhood.

Frustrations

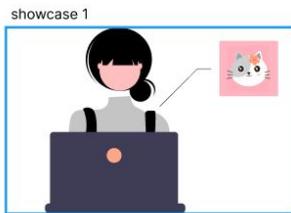
- Scared her cat will go missing.
- Can't find a way to track her cat without GPS.
- Can't find a proper platform obtaining an overview of pets moving in an area without supervision.

Most used apps



Appendix 4 - Used icons

Icons



Appendix 5 - Client views

Client login desktop view



Petfind

Jätkamiseks logi sisse

Kasutajanimi:

Parool:

Remember Me

Logi sisse

Ei ole veel kontot? [Võta meiega ühendust](#)

Copyright © 2021 - Petfind

"Minu loomad" view

Petfind Meist KKK Minu loomad [Logi välja](#)

Pildid, millel võib olla minu kadunud loom

Minu loomad

Xander



Minu kass

[Muuda Xander andmeid](#)

[Kustuta Xander süsteemist](#)

[Lisa uus loom](#)

Add new pet view

Petfind Meist KKK Minu loomad [Logi välja](#) [Administraatori vaade](#)

Lisa looma profiil

Owner:

Name:

Breed category:

Type category:

Address:

Chip nr:

Description:

Lost:

Notifications:

Animal image: no file selected

Recently seen animal card view

Petfind Meist KKK Minu loomad Logi välja Administraatori vaade Load from email LabelTool



www.rjakaamera.com KAMERA 108 050F 10C 09/05/2022 14:29:55

Original image



www.rjakaamera.com KAMERA 108 050F 10C 09/05/2022 14:29:55

Image with predictions

Süsteem tuvastas (liik:enesekindlus) - {'cat': '38%'}

Asukoht - R8 esimene, aeg - May 9, 2022, 2:30 p.m.

Tagasi Vale ennustus, saata märgistamisele

Appendix 6 - Pet identification system operator views

Admin page

The screenshot displays the Django administration interface. At the top, the header reads "Django administration" on the left and "WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT" on the right. Below the header, the page is titled "Site administration".

The main content area is divided into three sections:

- ANIMALS**: A list of management options for animals, each with "Add" and "Change" links.
 - Animals
 - Breed categorys
 - Locations
 - Pictures
 - Sightings
 - Type categorys
- AUTHENTICATION AND AUTHORIZATION**: A list of management options for users and groups.
 - Groups
 - Users
- DJANGO PLOTLY DASH**: A list of management options for dashboards.
 - Dash apps
 - Stateless apps

On the right side, there is a "Recent actions" panel. It contains a "My actions" section with a list of recent system events, each with a timestamp and a "Picture" link. The actions include successful operations (marked with a yellow checkmark) and failed operations (marked with a red X).

Recent actions:

- ✓ [Location: [R8 esimene', None, None]-, 'cat', 2, datetime.datetime(2022, 5, 9, 16, 41, 6, tzinfo=backports.zoneinfo.ZoneInf... Picture
- ✓ [Location: [R8 esimene', None, None]-, 'cat', 2, datetime.datetime(2022, 5, 9, 16, 41, 11, tzinfo=backports.zoneinfo.ZoneInf... Picture
- ✓ [Location: [None, None, None]-, 'bird', 2, datetime.datetime(2022, 5, 8, 17, 38, 3, tzinfo=backports.zoneinfo.ZoneInf... Picture
- ✗ [User: admin', 'A', 'MÄÄRAMATA', 'kass', 'Awe', '1', '123', False, True] Animal
- ✗ [User: admin', 'Admin', 'MÄÄRAMATA', 'kass', 'Add', '123', 'a', False, True] Animal

Appendix 8 - Feedback from the supervisor

Lemmikloomade tuvastamine

Kasutajatega testperioodi 1 ülevaade

Lemmikloomade tuvastamise projekti testimine viidi läbi kaamerate paigaldusega ja kasutajate rollide testimisega.

Testimine toimus 18. aprill – 8. mai 2022. Testimise korraldas tellija.

Testimiseks paigaldati esialgselt kolm kaamerat, pikemaajaliselt oli töös kaks kaamerat, mille asukohti muudeti. Üks kaamera asus Merivälja asumis ja teine Muuga aedlinnas. Testijad testisid administraatori rolli ning loodi ja kasutati kaks tavakasutaja rollis profiili.

Testimise eesmärk oli testida süsteemi riist- ja tarkvara toimimist realses keskkonnas, samas arvestades andmekaitselisi piiranguid.

Testimise tulemusena saadi järgmine teave:

1. Süsteemi põhifunktsioonid toimisid nõuetekohaselt.
2. Probleeme oli uute piltide laadimisega kaamera poolt saadud e-maili aadressilt. Pildid olid e-maili postkastis olemas, aga laadimisel tekkis mõnikord viga. Testperioodi lõpuks töötas piltide laadimine korrektselt.
3. Testijad märkasid ühte trükiviga ekraanil ja oli üks ettepanek kasutajavormi väljade järjekorra muutmiseks.
4. Testimine andis uut teavet ka kaamerate töövõime kohta: kuigi metsloomade rajakaamerana on piisav kaugus 5-15 meetrit, siis selleks, et saada piisavalt suur pilt üksi liikuvast lemmikloomast, peab kassi puhul kaugus olema 2-3 meetrit, koera puhul 2-5 meetrit. Samuti ei pruugi õnnestuda püüda loom kaadrisse, kui ta liigub kaameraga risti väga kiiresti. Seetõttu saadi arusaamine kaamerate paigaldamise täiendavatest nõuetest, et saada kujutisi, mis ei oleks arusaadavad mitte ainult inimesele vaid ka masinõppesüsteemile.
5. Süsteem võimaldab tuvastada looma liiki, aga mitte konkreetset isendit või tõugu. Samas on ka see toetav, kuna vajadusel saab sarnasuse täiendavalt tuvastada ka administraator/operaator.

Kokkuvõttes oli testimine edukas ja kinnitas vajalike funktsioonide toimimist.

08. mai 2022

Appendix 9 - References

Website	Address
Operator and user view	http://164.90.204.202/animals
Source code	https://gitlab.cs.ttu.ee/kvarik/lemmikloomade-tuvastamine