CMPE356 – Spring 2024 Term

Project Report Phase #3

Project No: 22

Kedi-net Istanbul –

Istanbul Cat Shelter Database

Submitted by:

Lena Heisel (CMPE)

Project Supervisor:

Dr. İlktan Ar

Project Reviewers:

Gülsüm Yiğit

Ahmet Kemal Cabbar

Yousef Hani Hassan Khalil

Faculty of Engineering and Natural Sciences
Kadir Has University

Spring 2024

TABLE OF CONTENTS

1 PHASE #3 – INTRODUCTION

In phase #3 of the software engineering project, the development of the backend was the focus. Different diagrams, thereof a class diagram, an ER model and five different sequence diagrams, were created, and the backend was built on top of them. For the backend development, I used Java Spring Boot framework. I used the key features that the Spring Framework offers: dependency injection, inversion of control, aspect-oriented programming and the model view controller architecture. Furthermore, I made use of the starter dependencies for web building and data JPA. I additionally used object-relational mapping using Hibernate to connect my logic with my relational MySQL database. The following report firstly shows phase #1 and phase #2. After that, we will continue with Phase #3, where I will present my fine-grained class diagram and my ER model. After that, I will explain my five sequence diagrams. Finally, there will be a conclusion and an outlook.

2 PHASE #1 – 3 SIMILAR WEB APPLICATIONS

2.1 Phase #1 Introduction

The number of stray cats in Istanbul is estimated to be around 160.000 [1]. While some of them are being taken care of and living a healthy life, others suffer from famine and illness. Luckily, different organizations and shelters are willing to take care of cats in need and look out for new homes for them to get healthy and feel safe. Due to the huge number of cats, shelters for cats are packed. It is not only my goal to help these shelters place cats into new homes where they are treated well, but also to reduce the trade of cats. I am convinced that with this high number of cats in Istanbul, everyone can find the exact cat they want, without having to support unnecessary animal trade.

Therefore, I aim to make it possible to have all cats from all shelters in Istanbul on just one website, which I will call “Kedi-net Istanbul”. This makes it as easy and as accessible as possible for a person wanting to adopt a cat, not having to search at every website of every shelter or even go to shelters, which can be time-consuming and deterrent. Additionally, I would like to provide the possibility of filtering as exactly as possible to find the exact cat a person is imagining. They can also specify the area where they would like to get the cat adopted. For now, my website is only thought of as a database for the shelter cats of Istanbul, but it could also be extended to a bigger region or the whole country. Other pets could be added on top of that, too.

In this report, I will share my research process. Since I am the only member of this group, I will take on all roles of the group. I found three websites similar to what I plan on doing, that I will analyze and compare to each other, to find out what kind of functional requirements my website should meet. Based on this, I will later discuss the SMART goals that I want my project to fulfil and end my report with a resulting conclusion. Before I go deeper into the research, I would first like to clear up the methodology fundamental to my preceding.

2.2 Methodology

Starting my software process, in my first phase I want to begin defining the functionalities of my software, which is part of the process state called ‘Software specification’ [2]. To gain insights and inspiration for my project, I had to find three websites that are similar to what I aim to create. For that, I used the Google Search Engine and clicked through several websites containing databases of animals in shelters. I narrowed my search a bit by only looking at websites in English since my website will also be in English. I collected all suitable websites and then chose the most fitting ones.

2.2.1 Research on Similar Websites

I then analysed the background technologies of the websites using a tool called ‘Wappalyzer’ to understand the underlying frameworks, the programming languages and other important components used in the development of that website. In my analysis, I will focus on the used frameworks. In addition to that, I looked closely at the structure and the functionalities of the websites from a user’s perspective. I wanted to observe not only how I could build a website most efficiently but also how the user would most probably have a good experience on that website and what exactly is needed for that. Finally, I will offer a list of all the advantages and disadvantages the different websites have concerning the topic of a database for cats in shelters.

2.2.2 SMART goals

Based on my findings, I will continue to find out what SMART goals my project should follow. ‘SMART’ in this case stands for Specific, Measurable, Achievable, Relevant, and Time-Bound goals [3]. This framework for defining my goals will help me to find the right goals that are possible for me to achieve in a relatively short time frame. Additionally, it can help me to keep on track and to make me focus on what is important during what time step.

2.3 Main Findings

I decided on three websites that share similarities with my chosen topics. I made my decision for these because they share some of the same features that were important for me like filtering for location and breed of cats but were different in the way they do it. They are as follows:

- Cats Protection: https://www.cats.org.uk/adopt-a-cat/find-a-cat
- Pet Finder: https://www.petfinder.com/search/cats-for-adoption/ca/qc/h0h/
- PetRescue: https://www.petrescue.com.au

2.3.1 Cats Protection

'Cats Protection' is a website that shows cats from shelters within the UK. It uses 'Microsoft ASP.net" as a web framework, which can be seen as an alternative to Spring Boot, which I will use in my project. It is open-source, and it is simple and fast to use [4]. It is based on the framework '.net', so that developers can use all libraries used on that platform, that offer different popular features [4]. Furthermore, for JavaScript, the library 'JQuery' is used. 'JQuery' makes creating a website with JavaScript easier and faster [5].

When you visit the website 'Cats Protection', you see a search field, where you can type in a postcode or a town within the UK. Additionally, there are five optional fields to set a tick, determining how the cat can live: with other cats or dogs, with children or older families or if it can live indoors only. Above the search field, you can click on a link to find out, how you can adopt a cat on this website. After you have typed in a postcode or town, you can see all the cats available for this location, as seen in Figure 1. There are no more specific options to filter or sort the results.

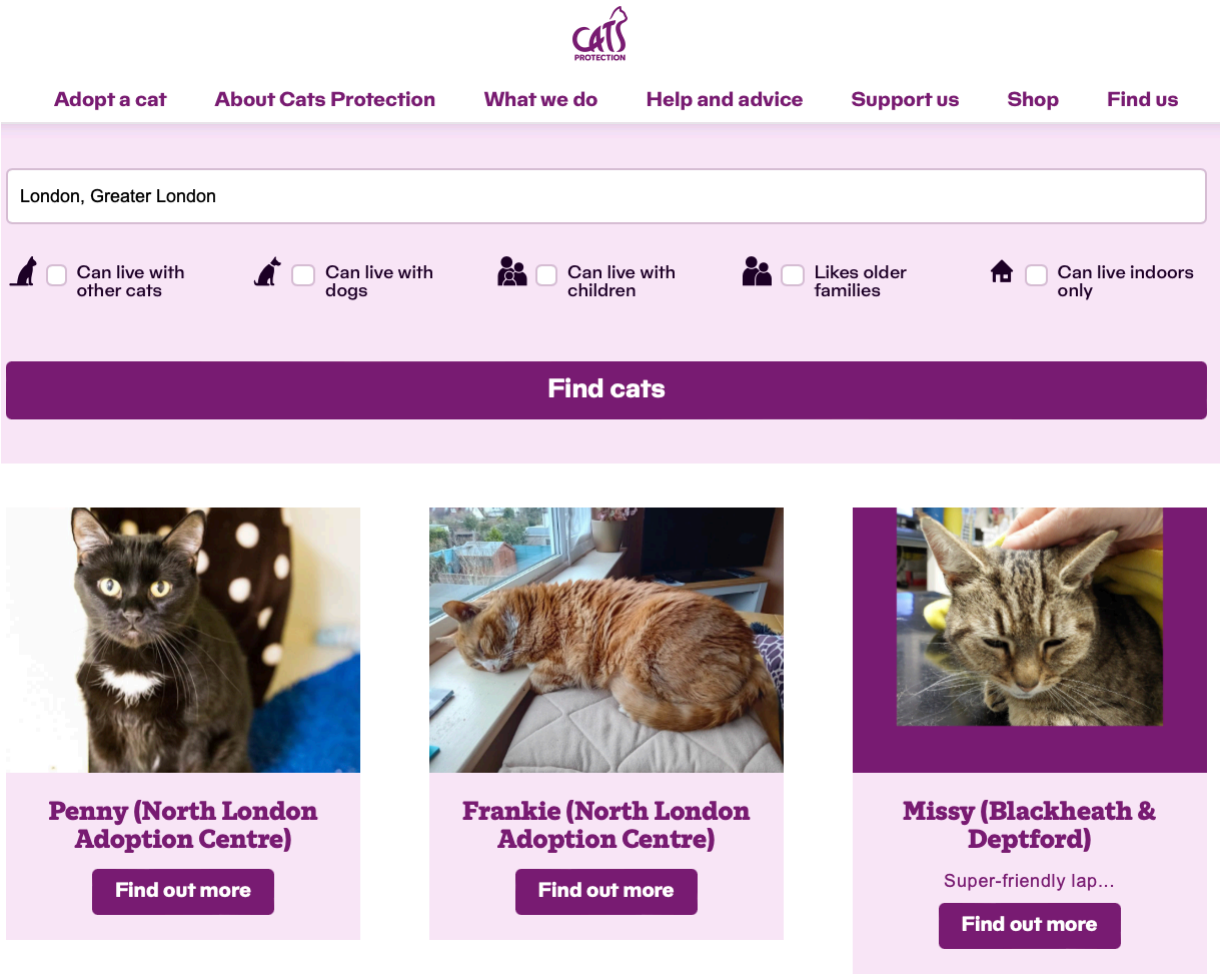

Figure 1 Searching for a cat on 'Cats protection' (Screenshot)

You can now click on the image of a cat to find more information. You will be led to a new site with descriptive text, more images and the same information that was also asked within the checkboxes before, as you can see in Figure 2. You also get the gender and the age, as well as a telephone number to contact the shelter.

Figure 2 Information on a cat on 'Cats protection' (Screenshot)

2.3.2 Pet Finder

On the website called 'Pet Finder', you can find different pets in shelters in Canada. For this website, I get a lot of information from Wappalyzer on their used frameworks and libraries. One of the used JavaScript frameworks is 'React', which could also be an option for me to use since it is made for interactive websites [6]. 'JQuery' is used on this website as well as a JavaScript library. As a web framework 'Next.JS' is used, which is the framework for the web when using 'React' [7]. As a UI framework the developers of 'Pet Finder' used 'Bootstrap', which is an important CSS framework for creating responsive websites and for the underlying database 'MySQL' was used.

When you visit 'Pet Finder', you see a search field, where you can first type in that you would like to adopt a cat and then you must also name a location in Canada. Then, a new site

is loaded, where you have different options to filter your search even more, as seen in Figure 2. You can choose between different breeds, ages, sizes, genders, ‘good with…’, coat length, colour, care and behaviour, days on ‘Pet Finder’ and shelters or rescues. You can also type in the name of the cat to look for a specific one. In addition, you tick a box, to include pets that must be transported from another location. It is also possible to be more precise on how near the cat should be to your chosen location and you can also sort your results accordingly. Besides, you can search by furthest, newest, and oldest addition, best match and randomized.

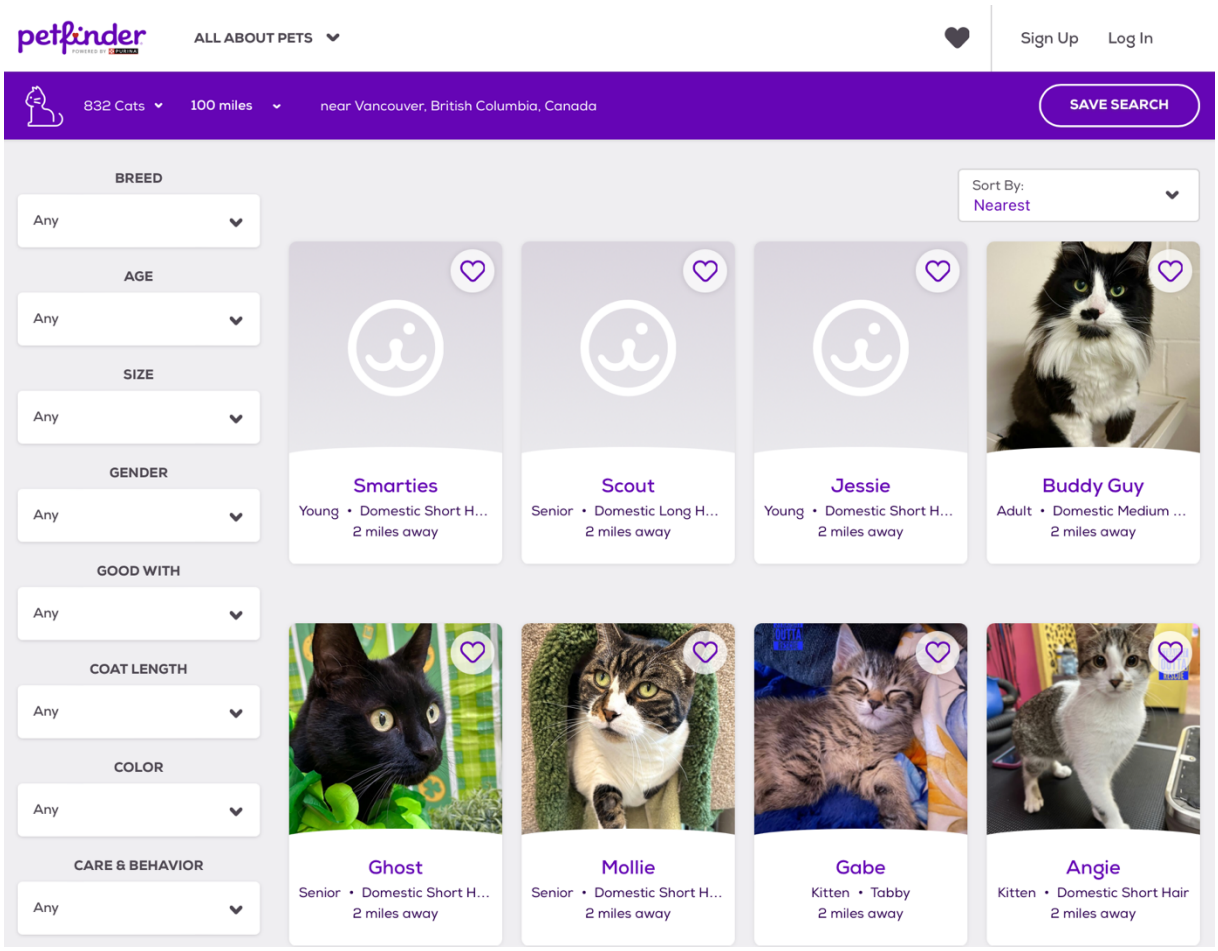

Figure 3 Searching for a cat on ‘Pet Finder’ (Screenshot)

Clicking on the picture of the cat will load a new site with additional information on the cat like the one you could choose in the filtering options, as seen in Figure 4. There are also more images, descriptive text, and information on how to contact the shelter. Furthermore, there is the possibility to read FAQs and to ‘heart’ the cat, adding it to a list of cats you might want to adopt. For this, you must create a user account.

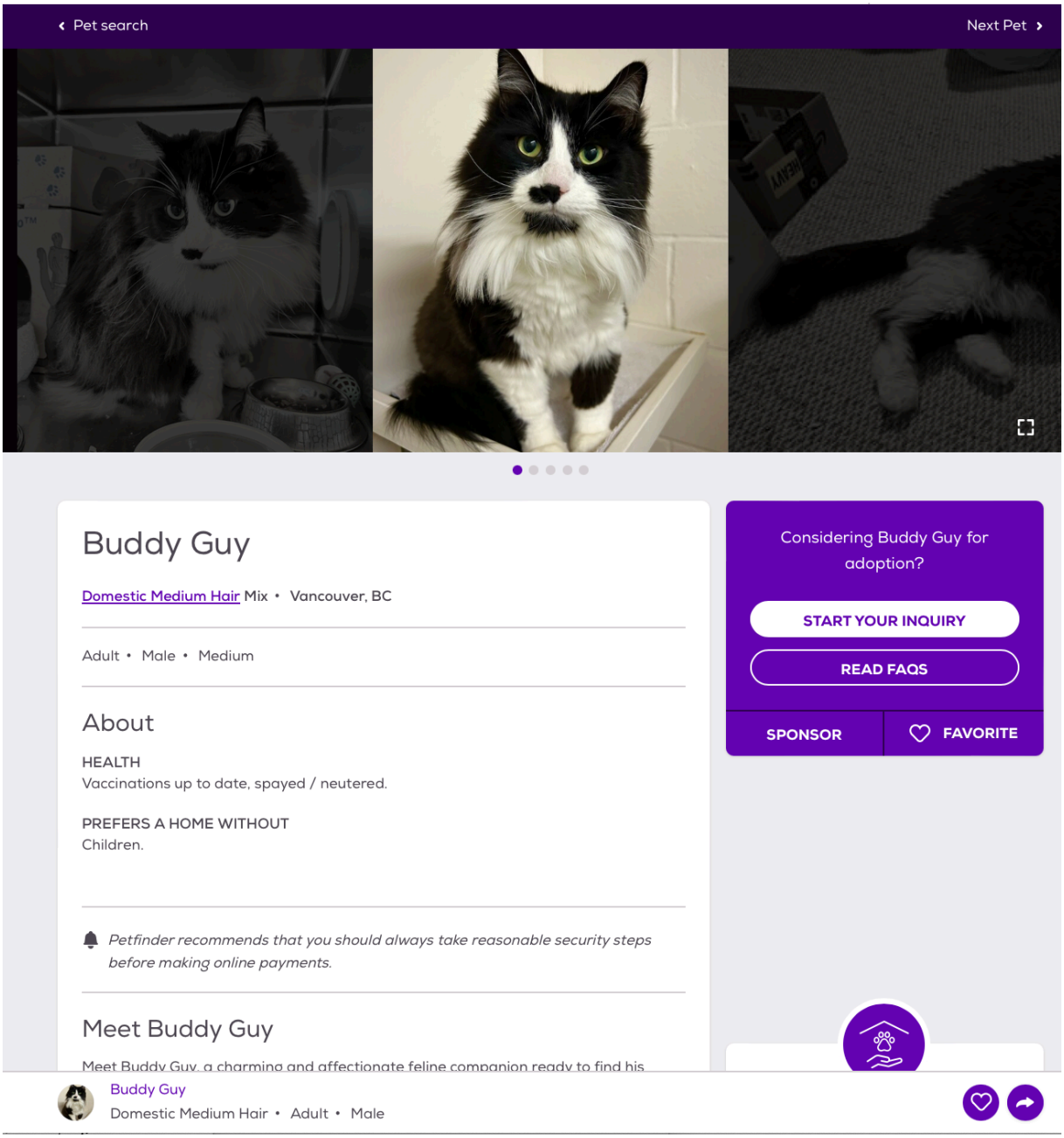

Figure 4 Searching for a cat on 'Pet Finder' (Screenshot)

2.3.3 PetRescue

'PetRescue' is an Australian website for finding all kinds of pets including cats from shelters all over Australia. It uses the JavaSrcipt Framework 'Alpine.js', which can be seen as an alternative to the framework 'React' [8]. 'Alpine.js' is more of a minimal framework made for smaller projects. Additionally, JavaScript libraries are used, mostly 'jQuery' and other 'jQuery'-connected libraries.

When you are on the page 'PetRescue', all you have to do to start looking for a cat is to click on 'cats', and then a menu will be shown underneath. On the right side, you can choose

the location by clicking on one of the eight possible states or territories. In the middle, there are no specifications to find a cat, but instead a ‘Cat adoption assistance’ with links to articles sharing tips or showing you how to set up an adopter profile. The left side shows some opportunities to view all, senior or bonded cats and cats that are only looking for foster care. Hovering over the fields will mark a field in a darker grey so that the user knows that they can click on it.

When clicking on ‘View all cats’, you can see cats from all over Australia or only in the specified area. Then, you get more options of filters to find the perfect cat as you can see in Figure 3. This is shown on the left side. You can now search by name and filter by state, cats near a certain postcode, gender, coat, age, more information about the home they need and if they are an indoor cat or if they prefer other cats at their home. Additionally, you can set a notification for pets that fall under the chosen parameters. Furthermore, you can clear all filters and decide how many cats are shown on one page. Also, you have a help button, where AI can answer some frequently asked questions.

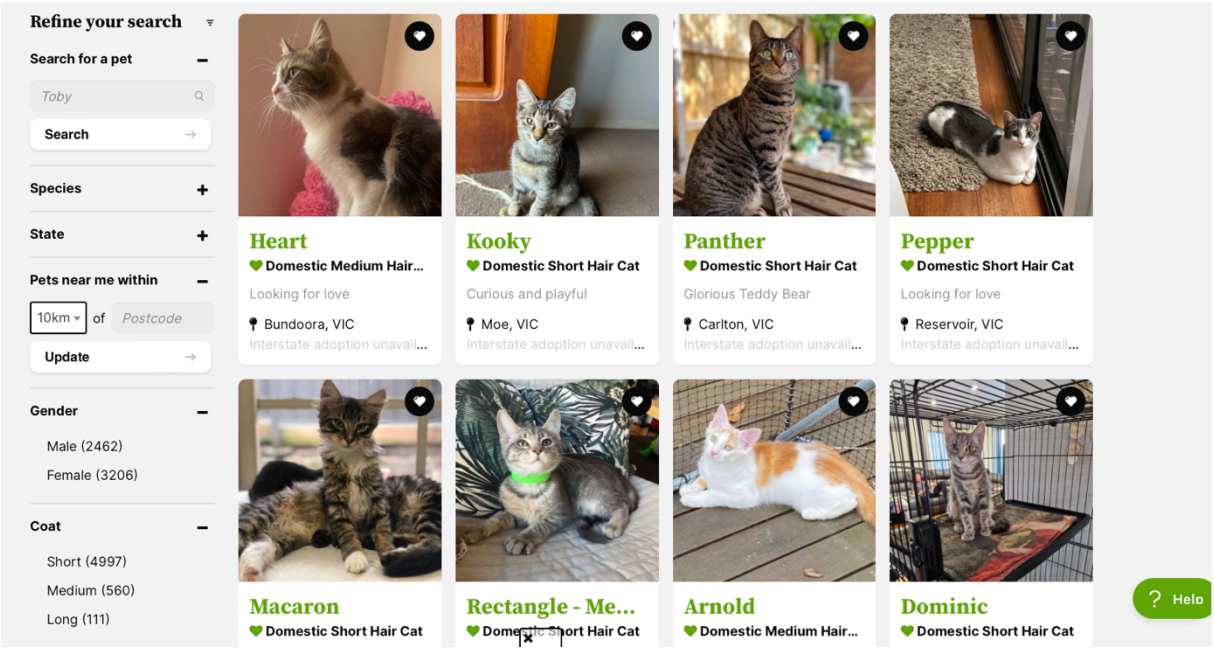

Figure 5 Searching for a cat on ‘PetRescue’ (Screenshot)

Clicking on the image of one cat, you go onto another page that shows more images and gives more information with descriptive text, important IDs and numbers, the adoption fee and pre-adoption checks like vaccination etc. This can be seen in Figure 6. You can ‘heart’ a pet to

add it to a list of cats you potentially would want to adopt, but it's only possible if you have a user profile for that page.

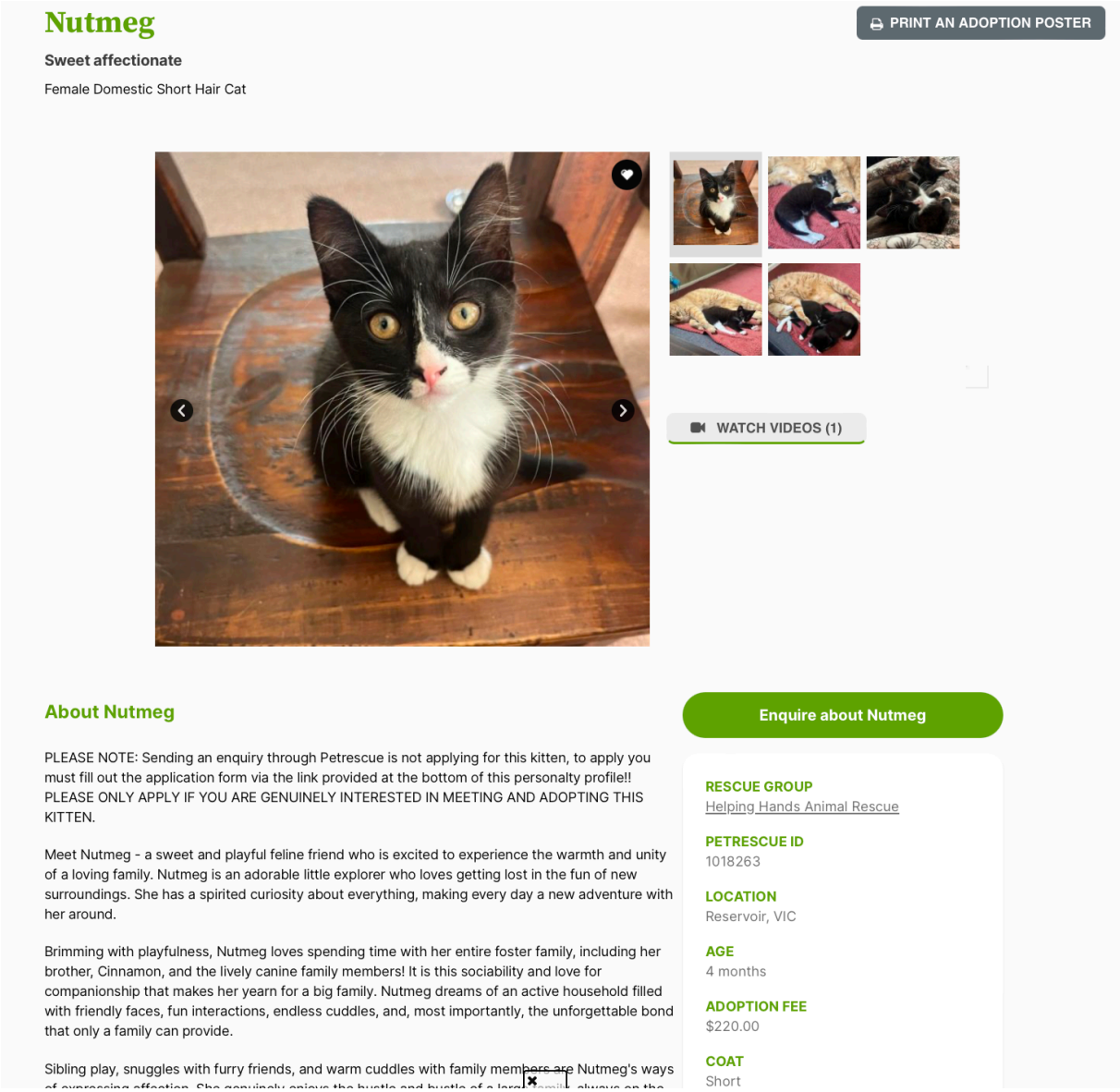

Figure 6 Searching for a cat on 'PetRescue' (Screenshot)

2.3.4 Advantages and Disadvantages of Websites

	Advantages +	Disadvantages -
Cats Protection	+ simple and straightforward design + search can be done fast + has additional information on how to adopt + you can search cats near ymy postcode or area + no advertisements + is responsive and can be used with different display sizes	- not many filters, so you cannot be very specific - not too much additional info about the individual cats beside the descriptive texts - you have no option to sort the results
Pet Finder	+ gives the option for a 'quick search' with only naming the region + includes a variety of options to filter by + you can search cats that are near ymy postcode + you can sort ymy results in different ways + you can add cats you might want to adopt to a list + there is additional information on the individual cats + you can see quickly how far away the cat is from the given location + no advertismentes + is responsive and can be used with different display sizes	- you need to create a user profile for some functionalities to work - you have to load a second site to be able to see all filtering options shown
PetRescue	+ gives the option for a 'quick search' with only clicking on the region and the kind of cat you want + has additional information on the functionalities of the website and tips for adoption + includes a variety of options to filter by + you can search cats that are near ymy postcode + you can get help by asking an AI chatbot + you can add cats you might want to adopt to a list + there is a lot of additional information on the individual cats + the user can get notifications for new cats + is responsive and can be used with different display sizes	- you need to create a user profile for some functionalities to work - you have to load a second site to be able to see all filtering options - you have no option to sort the results - inbetween the results, advertisements are shown

Table 1 Advantages and Disadvantages of Websites

2.4 Discussion

What I can see from my research of the three websites using Wappalyzer is that lots of different frameworks and libraries were used. But all the websites used jQuery as a JavaScript library, and one used React. In my project, I can choose between React and Angular for the frontend. While Angular is an actual framework for frontend, React is only a library for JavaScript [9]. React has the advantage of having reusable and more predictable code, a faster development time and the possibility for integration with third-party libraries [9]. Angular on the other side has the advantage of an intuitive application structure and is easy in building, maintaining and testing [9]. Not having used any of the two, it is hard to say, which one would be the better choice for building my software. But since in my observations of the websites React was used and Angular was not, I now tend towards React. Using React, libraries like JQuery that were used on all of the three websites, might not be needed then. Since it is already set for my project to use Spring Boot for my backend, I will not have to search for further alternatives. But, I looked up the combination of Spring Boot and React and there are many helpful sources on how to build software with the help of these two components.

Looking back at my findings from the three websites, I can summarise that the last two websites ‘Pet Finder’ and ‘Pet Rescue’ had similar approaches when it came to the filtering options, while the website ‘Cats Protection’ was structured simpler and more showed a more straightforward approach. For me, something in between these two kinds of approaches is what I aim for since I might not have so much time to develop a very complex website with many different options and helpful functions.

2.4.1 User Experience Considerations

Looking closely at my table of advantages and disadvantages (Table 1), I favoured a quick search but also considered the option of having a lot of choices for filtering as an advantage. Furthermore, I saw it as a disadvantage that a user had to go to another site to see all the filtering options. So, I concluded that it would be best, to see all the options possible for filtering from the beginning on without having to choose any of them. It should be possible to search for all cats in all Istanbul, or only in specific parts. But you can also be very specific with what your cat should be like from the start.

Sorting:

Another important feature for me would be the sorting. Only on one website, it was possible to sort the results. But this is a relatively important feature when someone tries to find something effectively and quickly. This is why I want my results to be able to be sorted. In what way will be specified as soon as I have decided on all the characteristics you can filter the results. Deciding on the characteristics of the cats will be done in the last phase of this project. This is an important decision since it will determine how my database will be built.

User Profiles and 'saved' lists:

I thought that it would be a good idea, that you could add a cat you might want to adopt to a list that can be saved. However, having to set up a user account is annoying, and it might keep one from using this option. So, the 'saved' list will only be an option for those who want to sign up. Another advantage. of that I do not think that it will make sense for my website, is that you can see how far away the cat is from you. Since Istanbul is not as big as a whole country, the feature is not as important. I would still like to show in what part of Istanbul the cat is located.

Notifications and Advertisement:

Two things I additionally deducted were the possibility of getting a notification when a new cat is added and the possibility of showing advertisements on the website. Both will not be integrated into my systems. Getting a notification is often connected to having a user profile, which I do not want to implement. In addition, it is more of an 'extra' function, that seems nice to have but is not an important functionality. Showing advertisements is something I do not want to implement since it is not my goal to make money with this website. From a user's perspective, advertisements are also mostly only annoying and distracting.

Help and FAQs:

Two of the observed websites gave additional information on how to adopt a cat there. I thought this was great to get a better understanding of the functionality of the website. Of course, I aim to make my website as easy to understand as possible, but it can always happen that some things remain unclear. That's why I would also like to implement a way of giving information on how the adoption works. One of the websites was supported by an AI Chatbot that could answer the most frequently asked questions. Though I think that this is an amazing

tool and very user-friendly to have, I, unfortunately, will not be able to develop or integrate such a Chatbot since I lack the knowledge.

Responsive Web Design:

All the observed websites had a responsive Web Design. I want to achieve as well, that the website is equally convenient to use on a smartphone, a tablet, or a computer. What I could even improve from what I have seen on the three websites is, that when you zoom out and drag the screen larger, you can see more containers for cats distributed across the width of the screen.

2.4.2 Functionalities and SMART Goals

Summarizing my previous findings, I came up with the following functional requirements, that I can view as my specific goals:

- Create a database with information on cats and shelters
- Some of these characteristics of cats can be used for a filtered search
- Some of these characteristics of cats can be used for sorting the results
- I need two types of pages
    - A „search page“ with filter and sorting options
    - A „cat page“ with information on the cat
- Optional is a page on how the adoption on this website works, but that could also just be a pop-up window
- My web design should be responsive, simple and as user-friendly as possible

It is my goal to have at least these requirements implemented in my software at the end of this project, leaving possible space for extra features that are not essential.

To make my goals more measurable and time-bound, I aim to start with the database at the end of March and be done with it within April. I also want to have a minimum of 500 cats from at least 10 shelters listed in that database. They can be created artificially. Starting the work on my backend at the end of March as well, I want it to be done by the end of May completely so that I still have enough time for user testing, updating, and documenting. By then, I want my search to be filtered by five characteristics of the cat. I want to start the frontend design at the beginning of April and and give me enough time until the end of April so that I can make sure to enable an excellent user experience.

Starting with the necessary and basic functionalities and giving me enough time for this makes my goals not only achievable since I do not have much time, but also concentrates on the most relevant functionalities. My priority is for the user to be able to search conveniently so that they come back using this website and want to adopt a cat, which in the end is my main goal.

2.5 Phase #1 Conclusion

All in all, I am satisfied with the selection of the three websites and the information I could gather from observing them closely. They gave me a good insight into how such a website is built and gave me inspiration for what I want to achieve and what I do not want for my project. I am optimistic about achieving the SMART goals that I have set for myself. However, it might be possible that I will have to change the time plan as I go since it is the first time for me building a complete software and it is hard to estimate what time I might need for what component.

In the end, it is my main goal to promote the adoption of cats in Istanbul by offering a simple and effective website that gives an overview of all the cats in the shelters of Istanbul. I believe that this website will not only benefit the shelters by increasing their visibility but also the potential adopters by simplifying their search process. Alongside this, I also hope to benefit from building this website since I can learn about how a software development process works, which will be an important skill for my further studies and my future jobs.

3 PHASE #2 - SYSTEM REQUIREMENT DOCUMENT

3.1 Phase #2 Introduction

Requirement Engineering is an essential part of the software engineering process. Requirements describe what a system should be able to do. In the following, I have divided the requirements needed for the "Kedi-net Istanbul" application into functional and non-functional requirements which can be seen in Tables 2 and 3. Functional requirements are those that provide a clearly defined service, while non-functional requirements describe constraints on the services and relate more to the system as a whole [10]. In both the table of non-functional and the table of functional requirements, there is also a subdivision into system and user requirements. User requirements describe which services are provided for the user, while system requirements are more detailed and describe what exactly the system must do, which functions it should have and which services it must offer [11]. In a requirements table, the author of every requirement is usually given, but since I am the only author of the requirements, I left it out in the following.

3.2 Functional Requirements

Req. Label	**User / System Req.**	**Details**
1	User	The user shall be able to search for a cat easily and efficiently.
1.1	User	The user shall have the opportunity to have filtering options to find a cat they like.
1.1.1	User	The filtering options shall be the area of the shelter, breed, ‘can live with…’, colour, gender, age, indoor cat, size, and coat length.
1.1.1.1	System	The filtering options shall be implemented with check boxes in the frontend.
1.1.2	User	None of the filtering options shall be mandatory. The user shall be able to use as many options they want.
1.1.4	User	The user shall be able to reset their filtering choices.
1.1.4.1	System	When a reset button is clicked, all filtering variables shall be set back to their initial value.

1.2	User	The search (including the optional usage of the filtering options) shall result in a listing of all cats fitting these options.
1.2.1	System	The fitting cats shall be found using SQL queries and shall be handed back as a list to be displayed on the website.
1.2.2	System	A ‘search’ button shall result in the showing of that listing.
1.2.3	User	After the first search, the user shall be able to further change their filtering choices.
1.2.4	User	The user shall be able to sort the search results.
1.2.4.1	User	The search results shall be able to be sorted using the following characteristics: newest/oldest addition, youngest/oldest cat.
1.2.4.2	System	The results shall be sorted in the ways mentioned using SQL queries.
1.2.5	System	The cats in the results shall be shown with the following information: picture, name, gender, and area of shelter.
1.2.6	User	The user shall be able to click on the box, where the image and the name etc. of a cat are lying, to go onto another site, with more information on that cat.
2	User	The user shall be able to visit a ‘profile page’ of a cat to get more information on the cat.
2.1	System	Every cat shall have its own ‘profile page’.
2.1.1	System	Information on the cat shall be saved and provided using a SQL database.
2.1.1.1	System	Every cat shall be identified by a unique ID.
2.1.1.2	System	Every cat is connected to a shelter, that shall also be identified with a unique ID.
2.1.2	System	On the cat’s profile page, all the characteristics shall be listed that we have on that one cat: name, gender, breed, age, indoor cat, size, coat length, colour, can live with, disease, descriptive text.
2.1.3	System	On the cat’s profile page, images of the cat shall be displayed.
2.1.4	User	On the cat's profile page, the user shall also get information on the shelter. That way the user shall be able to contact the shelter in case they would like to adopt the cat.
2.1.4.1	System	There shall be a link to the website of the shelter, the e-mail address and/or telephone number and their address.
3	User	A user shall be able to have a user account.

3.1	User	A user shall be able to sign up or log into their account using an e-mail address and a password.
3.1.1	System	The system shall make sure that the password of the user is save (longer than 8 characters). The user shall only be able to sign up if his email address is unique and the two passwords are equal.
3.1.2	System	The system shall save an e-mail address and a password so that the user can log in.
3.1.3	System	The user shall only be able to log in if his typed in password is equal to the saved one.
3.1.4	System	There shall always be an option of clicking on a "Sign Up" or "Log in"-button. When the user is logged in, the 'Log In' symbol on the page shall change to 'Log out'.
3.1.4.1	User	The user shall always be able to not only log in, but then also to log out.
3.1.5	System	The system shall be able to save the information if a user is logged in or not and if a user is logged in, which one it is.
3.2	System	Each user shall be uniquely identified by their email address.
3.3	User	The user shall be able to change their password.
4	User	The user shall be able to save cats they like to a list, if they are logged into their user profile.
4.1	System	The system shall be able to save a list of all the cats a user wants to save.
4.2	User	Users shall be able to not only add but also delete cats from the list.
4.2.1	User	The user shall be able to do this by clicking on a heart that shall be available on the listing on the 'search'-page as well as on the cat's profile-page.
4.2.2	System	This list of cats shall be saved in the database and can be edited by adding and deleting cats.
4.3	System	The system shall provide access to that list from any page of the website.
4.4	User	On the 'saved-cats'-page, the user shall be able to click on the box, where the image and the name etc. shall be lying, to go onto the profile page of the cat.
5	User	The user shall get help on how the website and the adoption process works.

5.1	System	The system shall provide help documentation accessible from any page of the website.
5.2	User	Whichever page the user is on, they shall have the option of clicking on a "Help" button and being redirected to a page with FAQs.
6	User	Admins (e.g. worker of a shelter) shall be able to edit what cats of their shelter are shown on the page.
6.1	User	Admins shall be able to use an ID and a password, they received before, to log in.
6.2	User	Admins shall have their own user interface to be able to edit their cats' lists.
6.3	User	The cats shown to the admin (also the ones the admin can edit) shall only be the one associated their shelter.
6.3.1	System	Every admin shall be connected to a shelter in the database.
6.3.2	System	The access the system gives to the admin shall only concern the cats of the connected shelter.
6.4	User	The admin shall be able to delete a cat by clicking a deltete button.
6.5	User	The admin shall be able to add a cat by filling out a form completely.
6.5.1	System	The system shall check if the form is filled out correctly and completely before adding the cat to the database.
6.5.1.1	User	The admin shall receive feedback on what must be filled out in what way.
6.6	User	The admin shall be able to log out wherever they are on the page.

Table 2 Functional Requirements

3.3 Non-Functional Requirements

Req. Label	**User / System Req.**	**Details**
7	User	A good user experience shall be ensured.
7.1	User	The website shall respond quickly, and the results shall load fast.
7.2	User	The website shall be intuitive and easy to navigate, ensuring that users can find desired information easily and fast.
7.2.1	System	All buttons and other elements shall be clearly labelled.

7.2.2	System	Textual content shall have a colour contrast for readability, and images shall include alt text for screen readers, to ensure accessibility.
7.3	System	The Web Design shall be responsive.
7.4	User	The website should always be available for users.
7.5	User	The user data should be stored safely and according to common Saftey Standards.

Table 3 Non-Functional Requirements

3.4 Use-Cases

Use cases (UC) are used to describe interactions between the system and its users [2]. This can be done with the help of text as well as with graphical models, so-called UC diagrams. The following Figure 7 shows an overview of my interactions and use cases. This figure is not a standardised diagram, it is only intended to provide an overview. The use cases contained in it are described in more detail below.

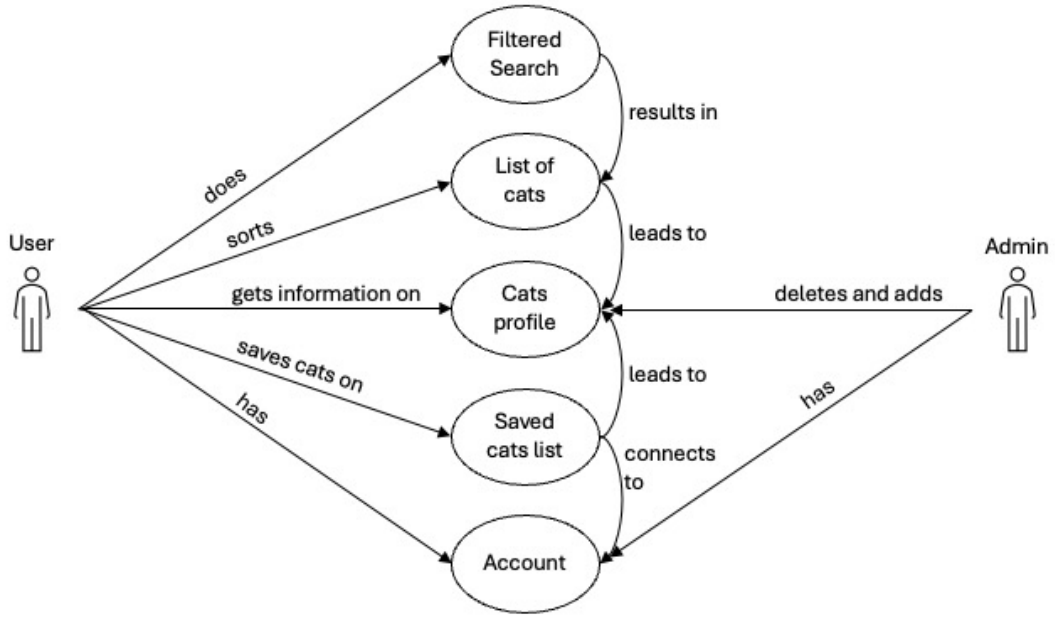

Figure 7 Overview of Use Cases

3.4.1 Use Case Descriptions

When describing the use cases, I divided them into five different use cases based on the specifications given by the requirements. Usually, the author of the use case is stated in such information, but since I am the only author, I have left this out.

UC #1: Search for cats:

Date: 11.04.2024

Purpose: The user can search for a cat efficiently.

Overview: A user visits the website to look for a potential cat to adopt. They start on the search page, where they have different filtering options. The filtering options are the area of the shelter, breed, 'can live with…', colour, gender, age, 'indoor cat', size, and coat length. Important to note is that none of the filtering options are mandatory and as many options as wanted can be chosen. When the user is done with choosing their filtering options, they can click on 'search'. The chosen filtering options are saved in the system. Then they are applied to the cat's table using SQL queries. A list of all the cats that fit the chosen characteristics will be given back. This list is now displayed on the website in the following way: Each cat is represented by a box with an image and details like name, gender, breed, and shelter location. The user can further change their filtering choices. The change of the choices is handled by the system and a new list of cats that fits the options is displayed. Additionally, the user can reset all filtering options by clicking a button. Then all filtering variables are set to to their initial value again by the system. The user can sort the search results using the following characteristics: newest/oldest addition and youngest/oldest cat. The system uses an SQL query to get the sorted list. The list will then be displayed again on the website.

Cross Reference: all requirements of label 1, UC diagram for UC 1 (Figure 8)

UC #2: Get information on cat:

Date: 11.04.2024

Purpose: The user can get information on a cat and the shelter it is in.

Overview: After the user has executed a search request and got the list of cats fitting the chosen filtering options, the user can click on any box representing a cat, to get more information on that cat. The user will be led to a new page: the cat's profile page. The system knows which cat should be displayed by handing over the ID of the cat belonging to the box the user clicked on. Every cat has its profile page with one to five images of that cat, that the user can click through.

All the characteristics that are saved in the database are displayed here: name, gender, breed, age, ‘indoor cat’, size, coat length, colour, ‘can live with…’ and disease. An additional information text is displayed as well. All this information, including the images, is saved in the SQL database in a cat-table. It is reached using a unique ID that every cat has. On the cat’s profile page is also information on the shelter of the cat, so that the user can contact the shelter. This includes a link to the website of the shelter, if it exists, their address, their e-mail address, and their telephone number. In the SQL database, every shelter has its table and its unique ID, and every cat is connected to a shelter by including the ID of the shelter in their table. That way the information on the shelter can be easily displayed by using an SQL query.
Cross Reference: all requirements of label 2, UC diagram for UC 2 (Figure 9)

UC #3: Sign up and log in to a user’s account:
Date: 11.04.2024
Purpose: A user can sign up for a user account to store information.
Overview: To sign up for an account, the user must first type in an e-mail address. The system verifies if it is an e-mail address. The user also must type in a password. They must repeat the password once, to make sure they did not misspell it. The system will check if the two are equal, if not the user is requested to check again. The system will also check if the password is at least eight characters long, which is due to security reasons. Additionally, the system will check using an SQL query if the e-mail address is new in the system, so if it is unique. If it is not unique the user will be informed that they already have an account and could log into it instead. If the e-mail address is unique and the password is approved as well, the user is now signed up and logged in and will be led back to the search page. The log-in data is stored in the database in a user-table. Furthermore, the system stores the information that this specific user is now logged in. When the user logs out using a log-out button, this information will also be stored. When a user wants to log in again, they must type in their e-mail address and their password. The system checks first if the typed-in e-mail address is an actual e-mail address. If not, the user will be informed about it. The system then checks if the password matches the unique e-mail address saved in the user-table in the database. If it doesn’t match, the user is informed about it. If it matches, the user is logged in and will be led back to the search page. The user can change their password again by going to the account and typing in a new password and repeating it once. The procedure is similar to the signup process.
Cross Reference: all requirements of label 3, UC diagram for UC 3 (Figure 10)

UC #4: Save cats to a favorite-cats-list:

Date: 11.04.2024

Purpose: The user can save cats they might like to adopt in a list for later.

Overview: To be able to save cats into a list, the user must be logged in. After the user has done a search request and has gotten back a list of cats, they are now able to click on a little heart that is in the box of each cat displayed on the site. The colour of the heart shows if the user has already added the cat to its list. If it is white, they haven't added it, if it is black they have added it. The system provides this information by checking if the ID of a cat can be found in a table of favourite cats that is linked to every user. If the user now clicks on a white heart, it is saved to this list in the database linked to this user. The heart symbol has now changed its colour to black. If the user clicks on it again, the colour will change back to white and the system will delete the cat from the list. Moreover, the user can see all the cats saved on their list on a page designed for this purpose. There the cats are displayed in the same way they are displayed on the search page but without the heart button. Instead, a button showing an X is shown on every cat signalising that this cat can be deleted from the list. When the user clicks this button, the system will delete the cat in the same way it was described before. After that, the shown list is updated, and the cat is not displayed on the page any longer. On this page, it is also possible for the user to click on a cat, and they will then be led to the profile page of that cat.

Cross Reference: all requirements of label 4

UC #5: Deleting and adding cats as an admin:

Date: 25.04.2024

Purpose: Admins can delete or add cats to their shelter.

Overview: To be able to delete or add a cat, an admin must first log in to their account. This is done in the same way as described for the user's account. The only difference is that the admin's information is stored in an admin-table and instead of an e-mail address an ID is used to log in. After having logged in, the admin is led to a page that looks the same as the page of favourite cats used by the users. The admin sees all the cats here that belong to the shelter that is associated with the admin. The system provides this information using an SQL query. Every admin has a table in the database where a corresponding shelter ID is stored. The admin can delete a cat by clicking on an X-button. The deletion works in the same way as for the favourite cat's list of users. The only difference is that a pop-up window is opened to ask the admin if

they are sure about deleting the cat. If the admin confirms, the system will safely remove the cat, so the corresponding cat-table, from the database. An admin has also the option to add a cat to their shelter by filling in a form. Every information that is given in a cat-table must be filled out here and will be transformed by the system into SQL code to add a cat to the database. The form consists of text fields, where the number of characters that can be typed is limited matching the specifications made for the SQL database. For those cases where pre-defined options are given, like the breed of the cat, the admin must simply put a check into a check box to decide on an option. The admin must make an entry for all properties of the cat, including the upload of at least one image of the cat. To finally add a cat to the database, the admin must click a submit button. Now the system checks if all inputs are given in a way that matches the specifications of the properties of the cat's table. If they don't match, the admin is informed about it. If any information is missing, the admin will be informed what he still must fill out in the form. If every information is given in the right way, the cat is added, and the admin will be led back to the page showing all the cats in the shelter the admin is responsible for. When the admin is finished with deleting or adding cats, they can log out again.
Cross Reference: all requirements of label 6

3.4.2 Use Case Diagrams

Use Case diagrams serve as a model of the interaction between an actor and the application, whereby the user scenarios can be visualised in a simplified way. I decided in favour of visualising use cases 1, 2 and 3. UC 1 shows many user interactions and UC 3 offers the possibility of simplifying the interaction between different processes in the system. UC 2 was ultimately very easy to visualise. I decided against visualising the last two use cases, as these are far more detailed and the interactions are interlinked, which makes them more difficult to present and could become too confusing. That is why a textual description is preferable. The use case diagrams should be viewed together with their related use case descriptions from the previous section.

UC Diagram 1: Search for cats:

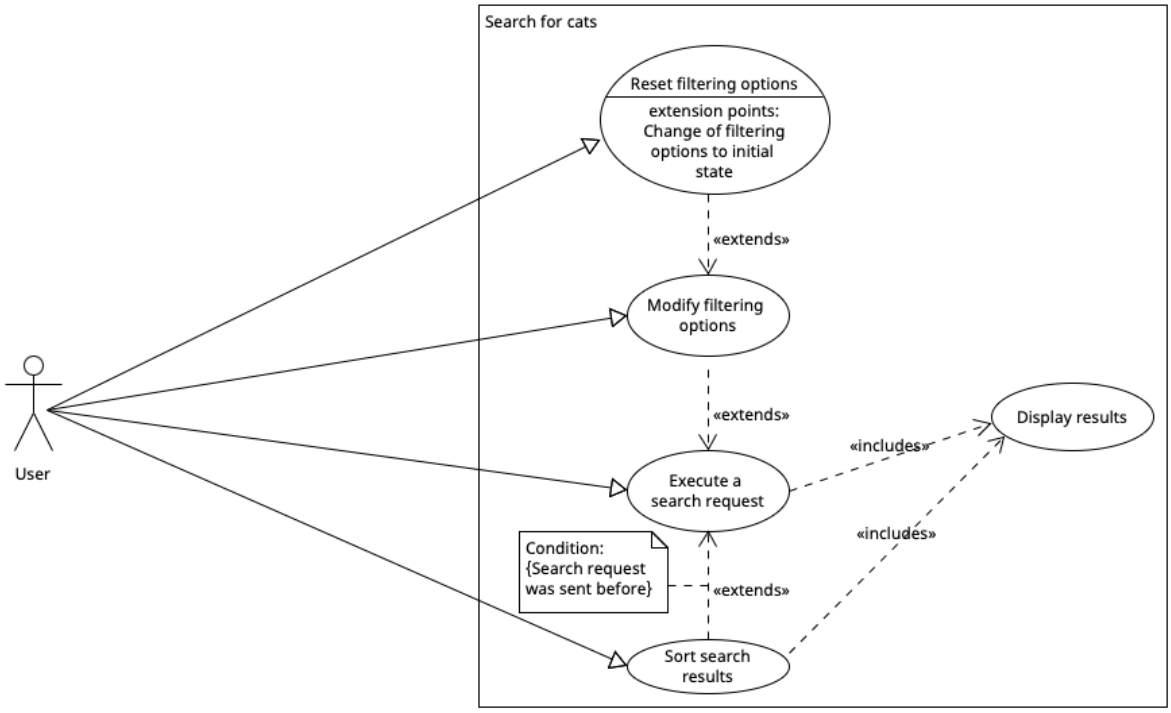

Figure 8 UC Diagram 1 – Search for cats

UC Diagram 2: Get information on cat:

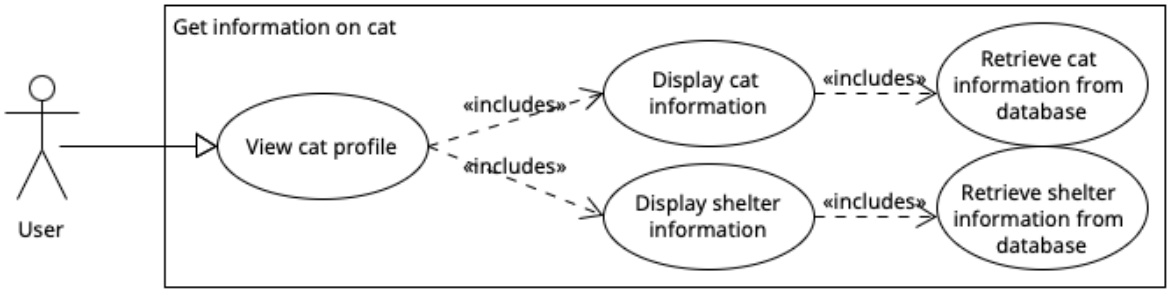

Figure 9 UC Diagram 2 – Get information on cat

UC Diagram 3: Sign up and log in to a user's account:

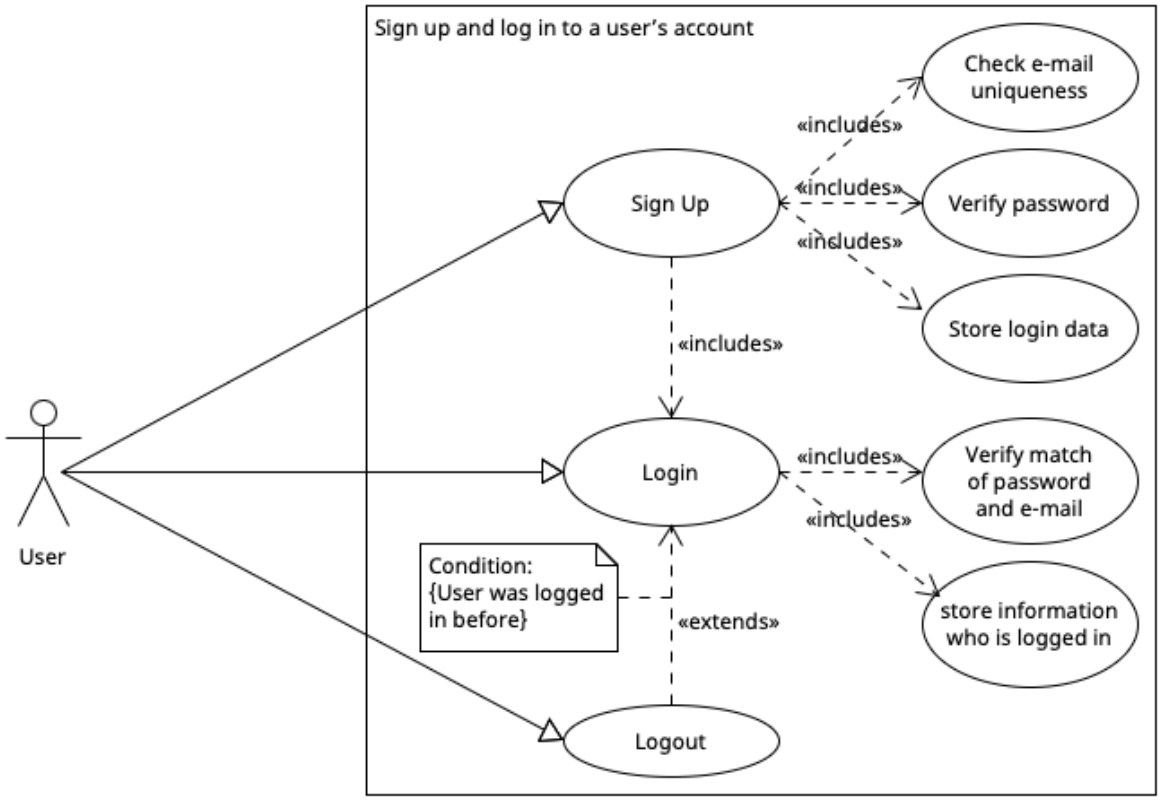

Figure 10 UC Diagram 3 – Sign up and log in to a user's account

4 PHASE #2 – FRONTEND DESIGN

4.1 Development Process

Now we come to the frontend design, which is crucial for a good user experience. To ensure a uniform design, I first decided on a colour scheme, which can be seen throughout the entire design. A uniform and attractive design is important for the user experience, as it can help to convey the ideas of the site, provide clarity and be more memorable for the user. I also created a logo using AI, which is shown in Figure 11. I then started designing the frontend by creating wireframes using Figma. I then converted the wireframes into code, using Bootstrap for the layout, a CSS-frontend framework [12]. Additionally, I used React to build my Javascript code. I decided in favour of React because it is the most popular frontend library and can also be implemented well with Bootstrap.

Figure 11 Logo of Kedi-net Istanbul

4.2 Wireframes and Final Frontend

As this is my first experience with frontend development, some of my ideas for the wireframes weren't converted one-to-one into JSX code. These changes in the process are presented in the following illustrations. Since the developed pages are closely linked to the use

cases, which were already described in a lot of detail, the descriptions in this part will mainly concentrate on the changes made in the design.

4.2.1 Search Page

On the search page, both the wireframe (Figure 12) and the implementation (Figure 13) initially show the navigation bar on the top, which can also be seen on all other screenshots since it was adopted for the final frontend. Below the navigation bar is the filter section. There, only the choice of buttons has been changed and the design has been made a little clearer. The display of the cats is also similar but was simplified a little more in the final implementation.

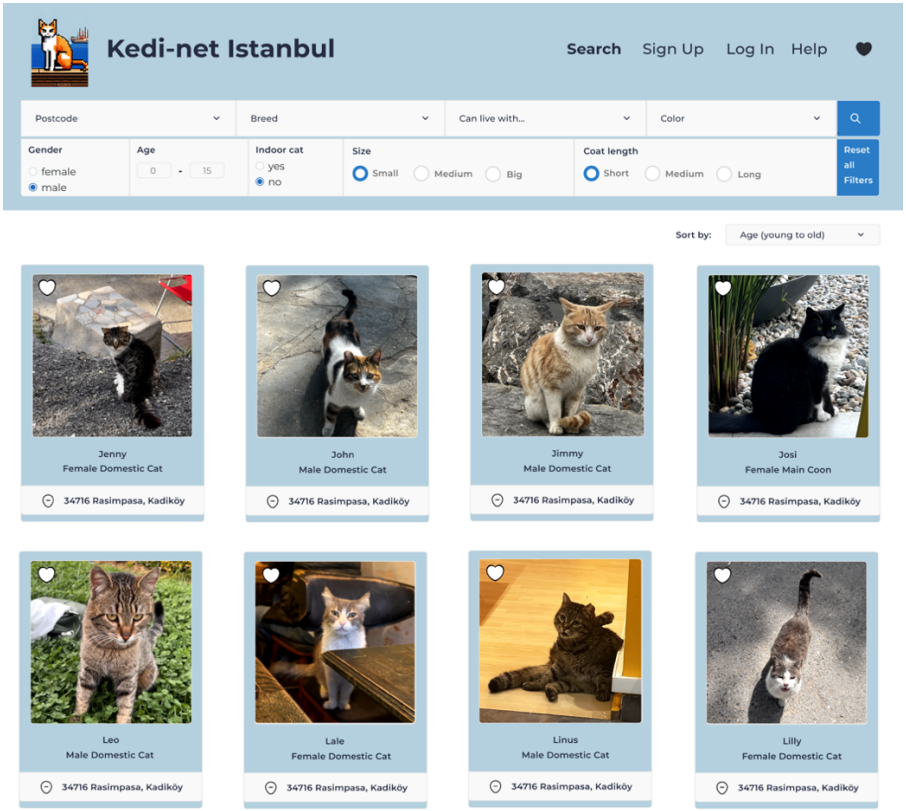

Figure 12 Wireframe ‘Search Page’

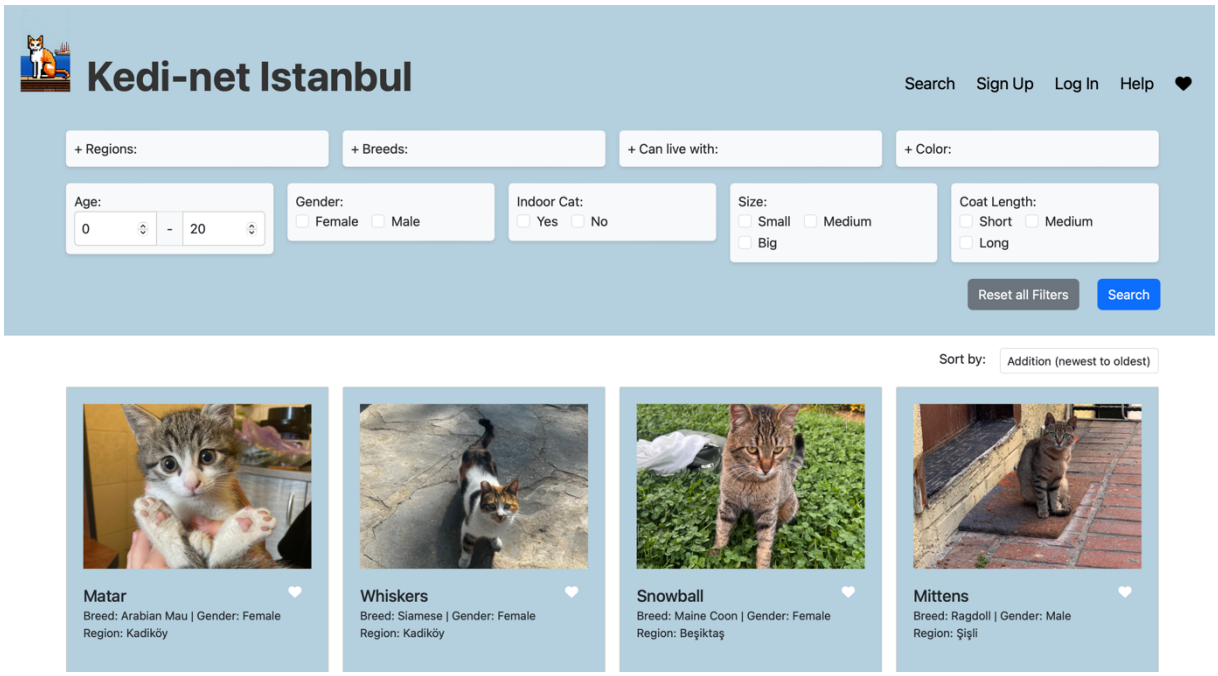

Figure 13 Final Frontend 'Search Page'

4.2.2 Cat's Profile Page

The cat's profile page has not changed much from the wireframe (Figure 14) to the implementation (Figure 15). I decided to use a little more colour for a friendlier design, but the structure is the same. You can click through several pictures and get information about the cat and the shelter.

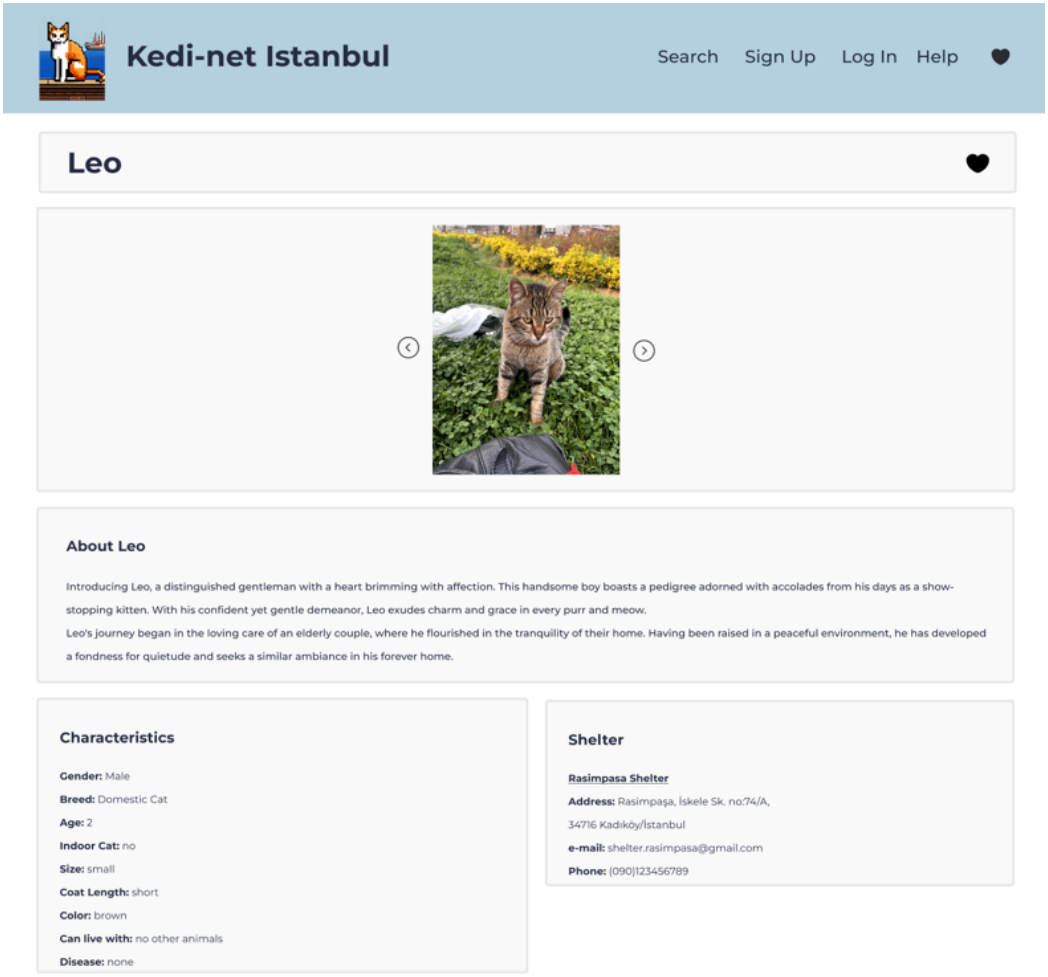

Figure 14 Wireframe 'Cat's Profile Page'

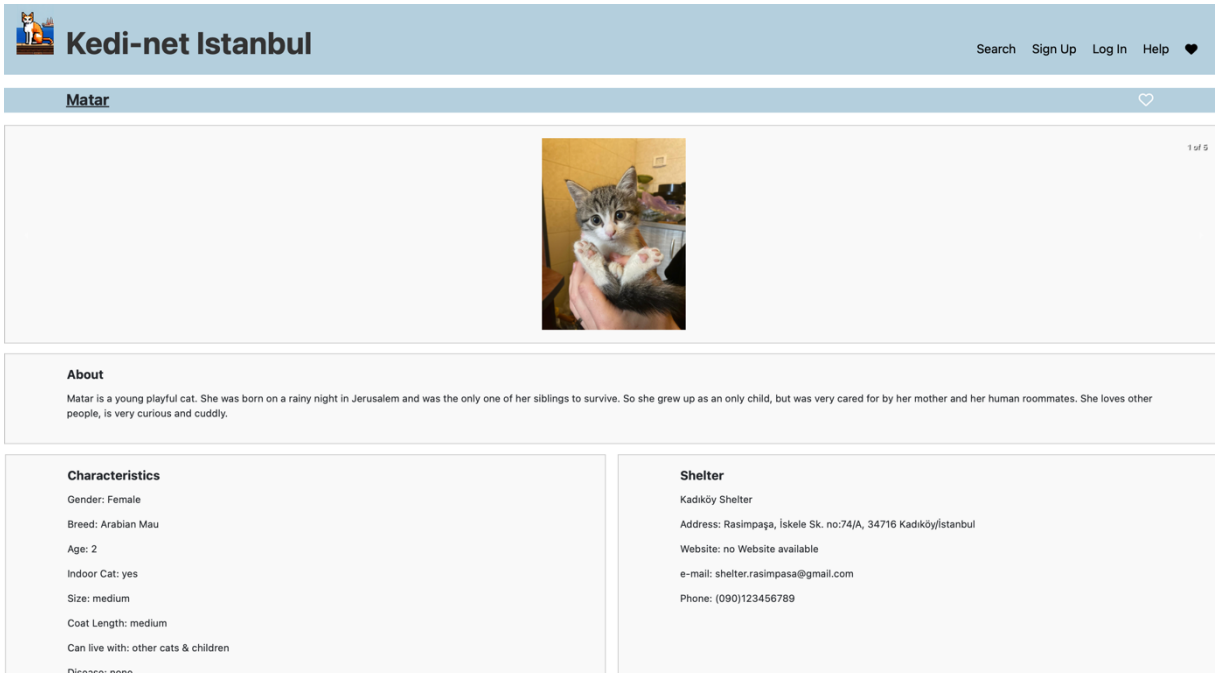

Figure 15 Final Frontend 'Cat's Profile Page'

4.2.3 Signup and Login Pages

As the signup and login pages look almost identical, it is sufficient to only show the login page here. There is also a user's account page, where the user can change their password, which looks also very similar and is left out here for the same reason. If you compare Figure 16 and Figure 17, you can see that the wireframe design has been implemented very similarly. However, in Figure 17 something new can be seen: the footer. This is retained in all pages accessible to the user and contains the contact, a link to the 'about' page and the link to the admin login. The admin login is somewhat hidden, as it should not be of interest to the user.

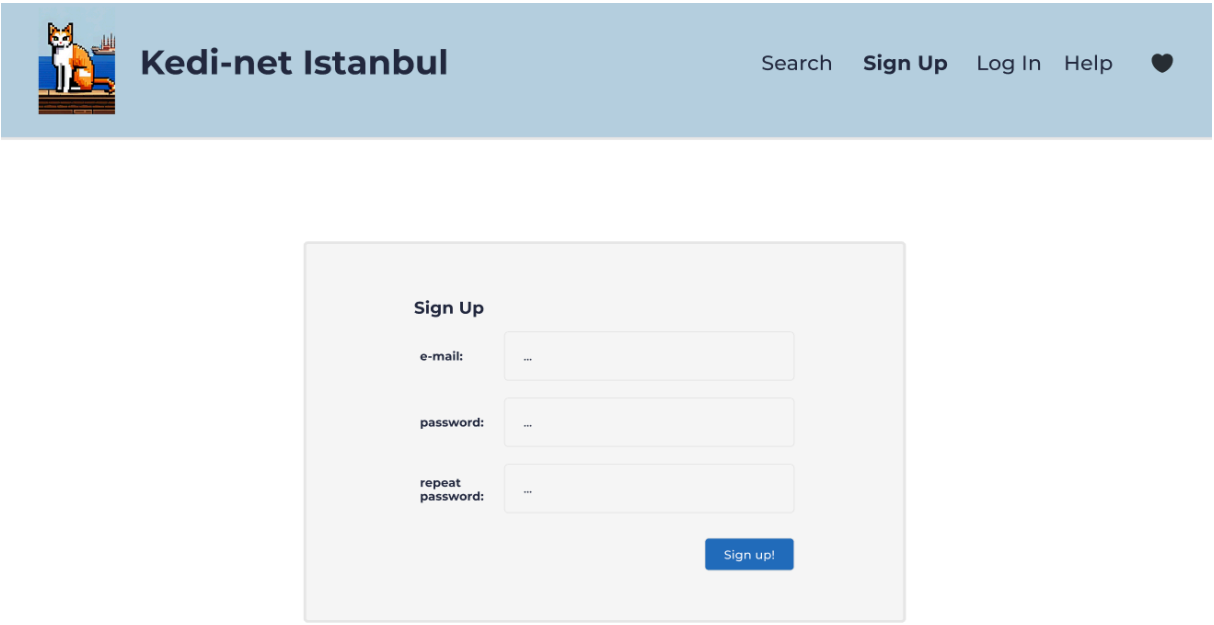

Figure 16 Wireframe 'Signup Page'

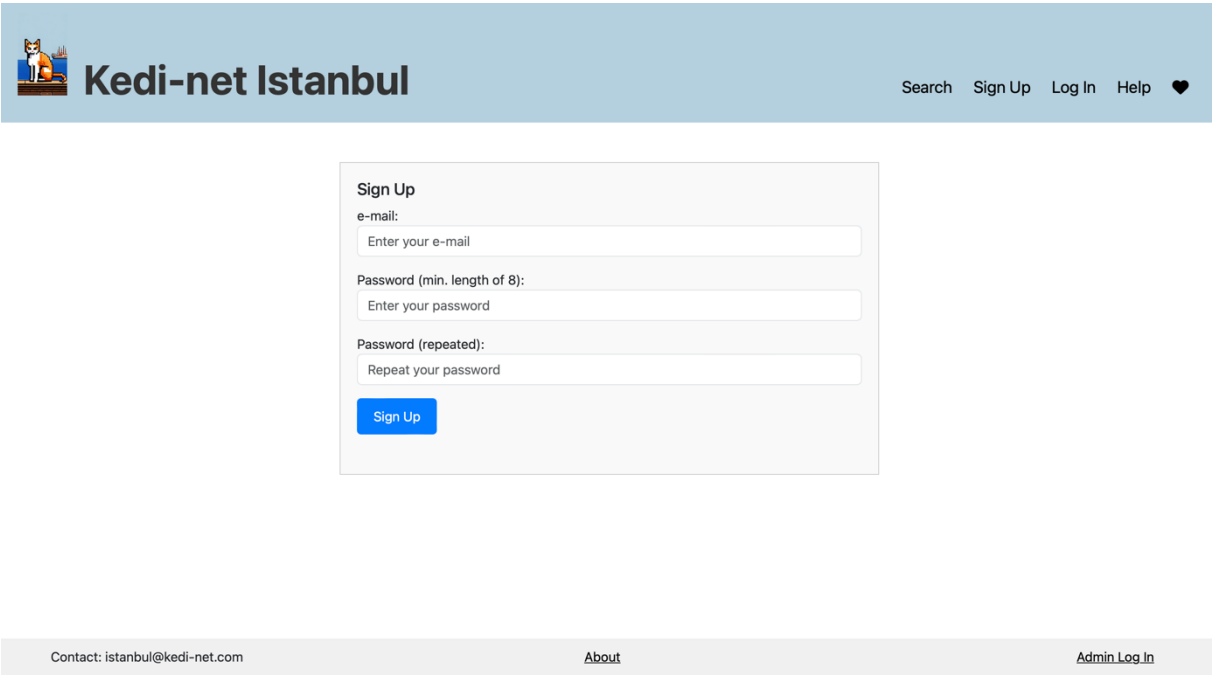

Figure 17 Final Frontend 'Signup Page'

4.2.4 Favourite Cats Page

On the favourite cat's page, you can see all the cats that a user has favoured. They can be deleted here using the X button, as shown in Figure 18 and Figure 19. The design was largely adopted from the wireframes. Only a container for the title of the page was added.

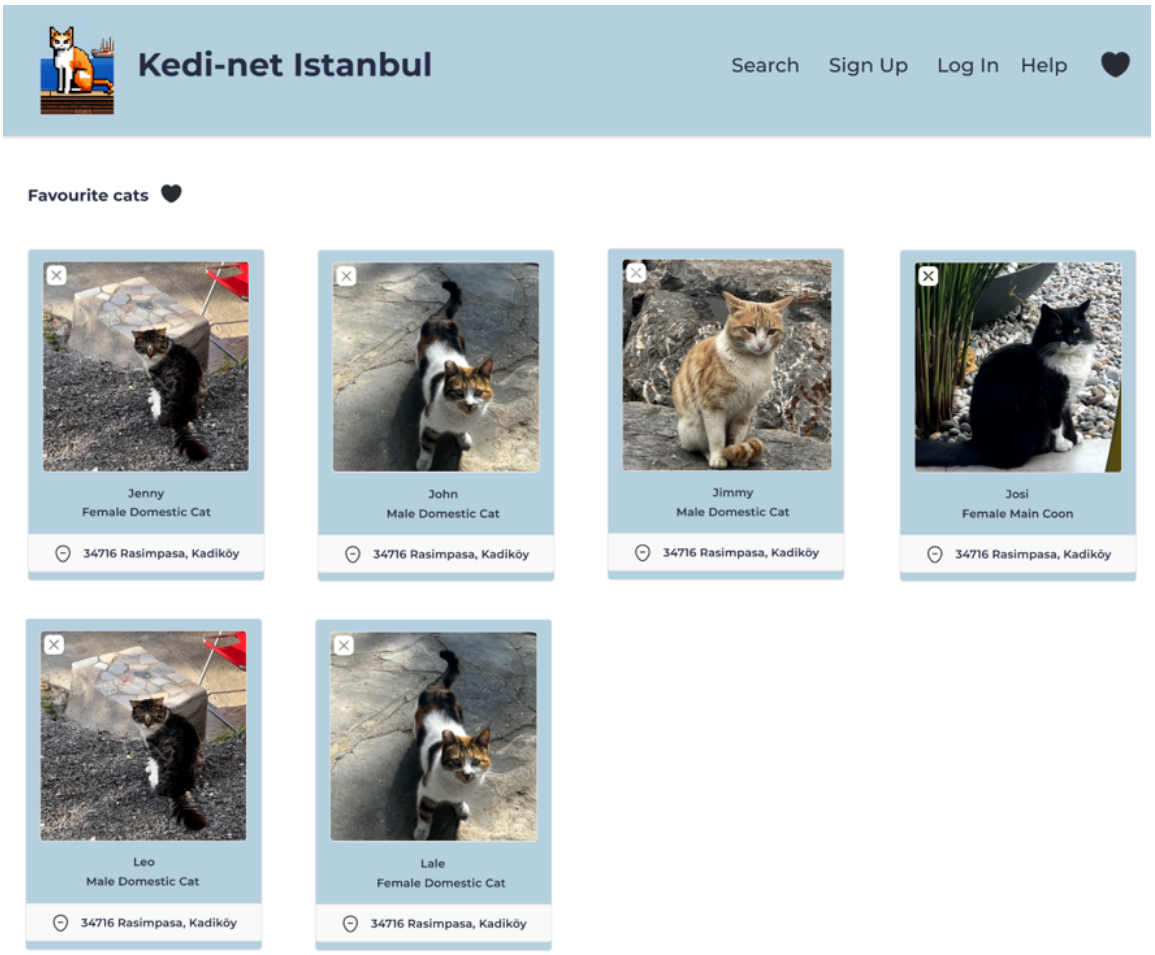

Figure 18 Wireframe 'Favourite Cats Page'

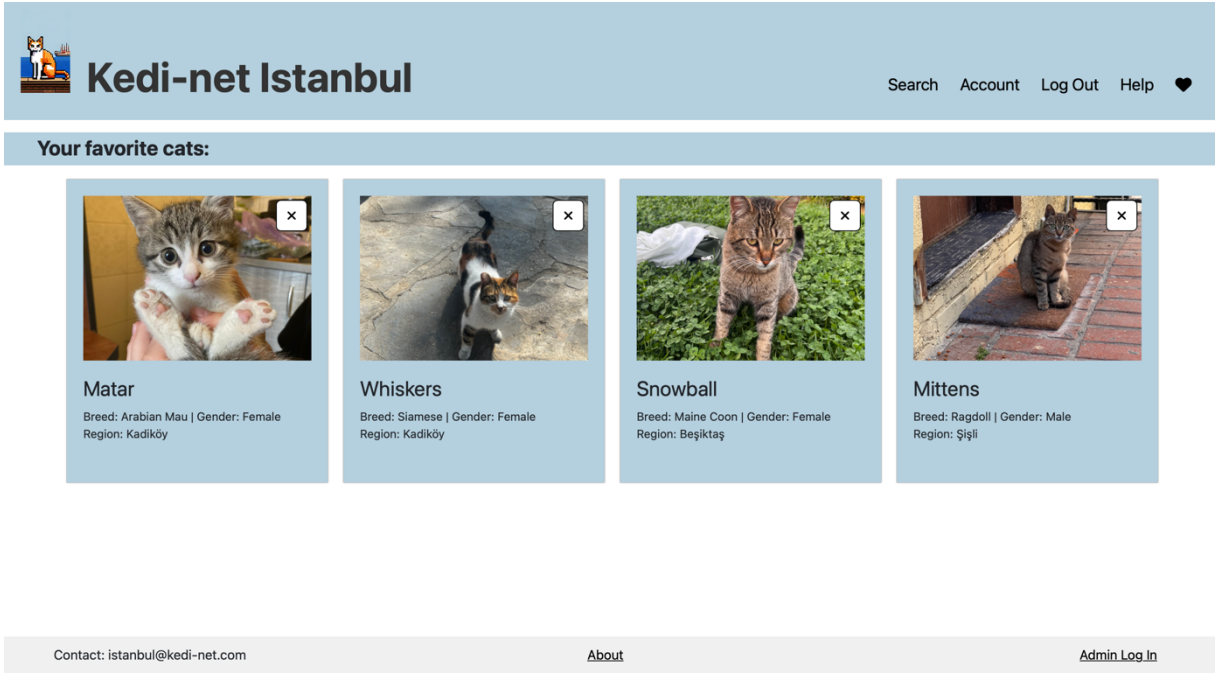

Figure 19 Final Frontend 'Favourite Cats Page'

4.2.5 Help Page

I divided the help page into two parts in the wireframe in Figure 20 and continued doing so for the implementation of the final frontend, seen in Figure 21. The first part is intended to answer questions about the website itself and the second part answers questions about adoption in general. What has changed for the final implementation is primarily the content.

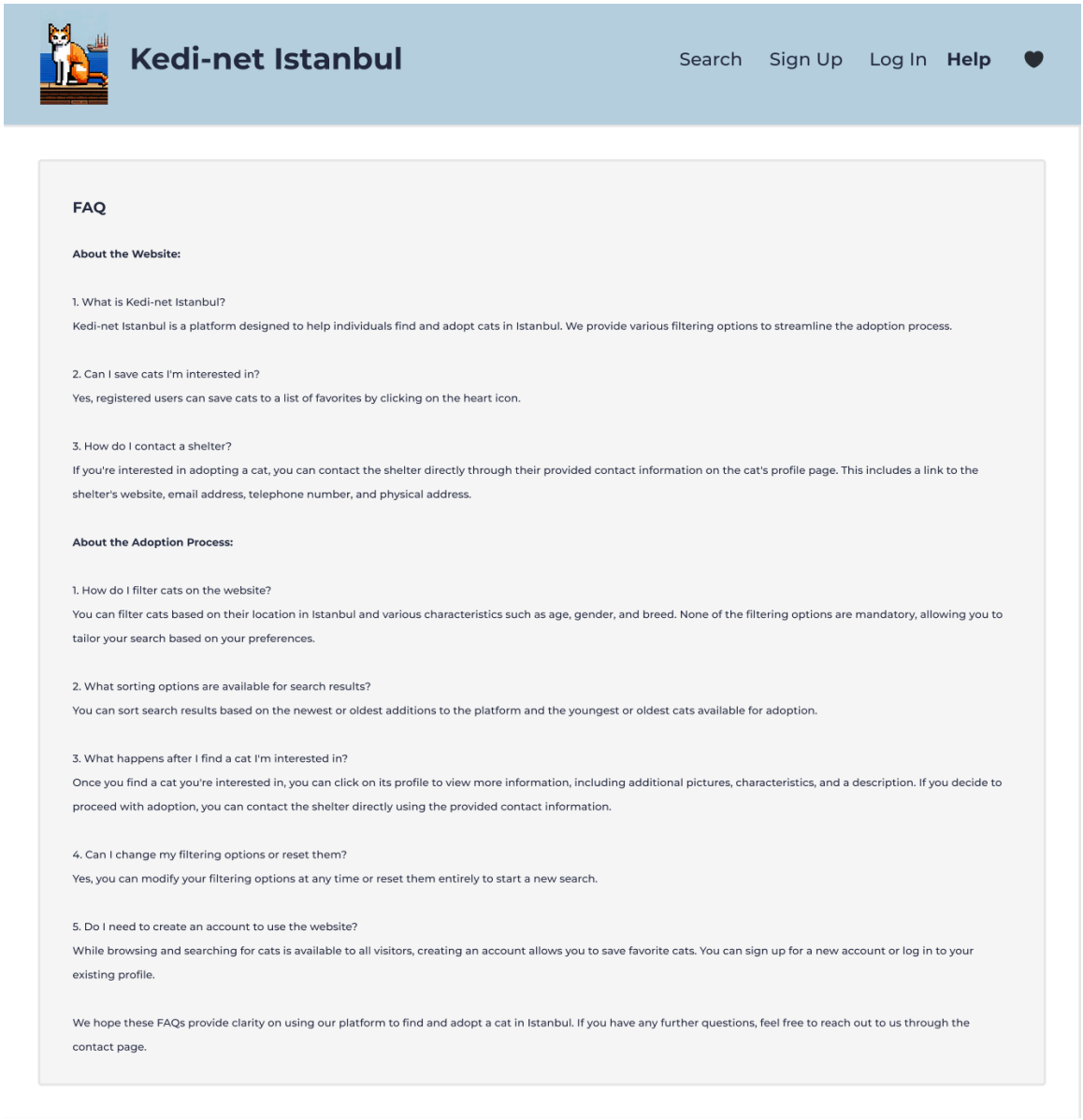

Figure 20 Wireframe ‘Help Page’

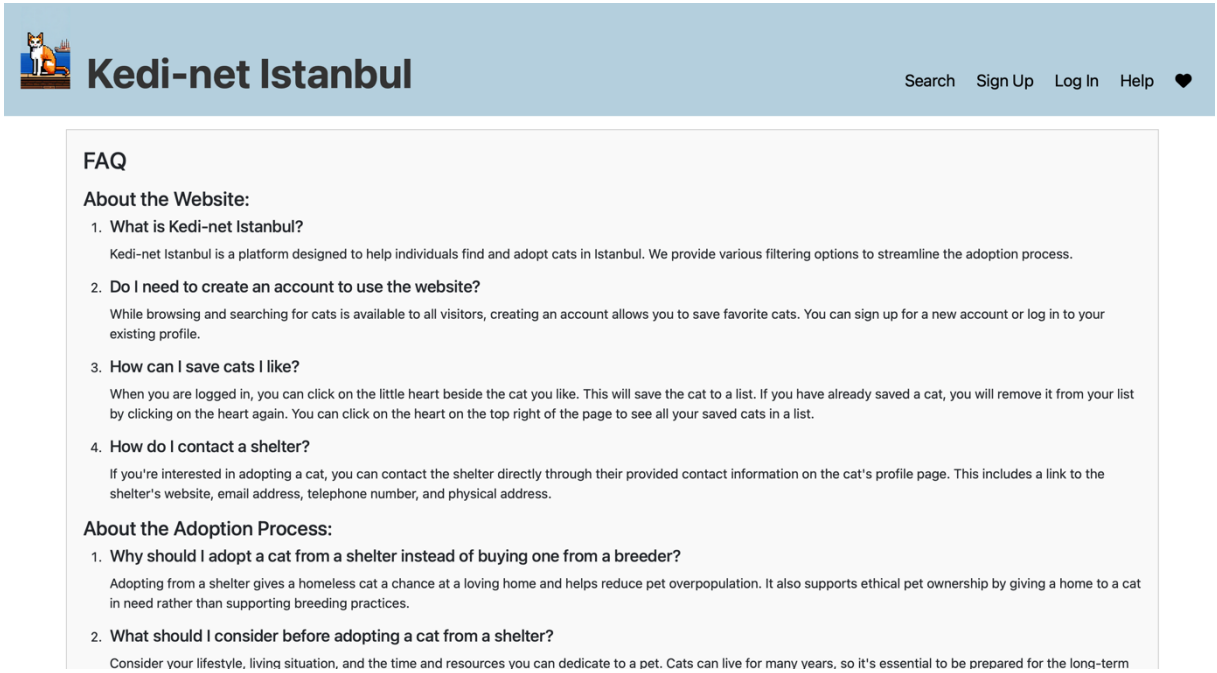

Figure 21 Final Frontend 'Help Page'

4.2.6 About Page

A new addition that was not implemented in the wireframes is the about page, which can be seen in Figure 22. I noticed that every website has an about page, which is why I added one. It is kept in the usual standardised style.

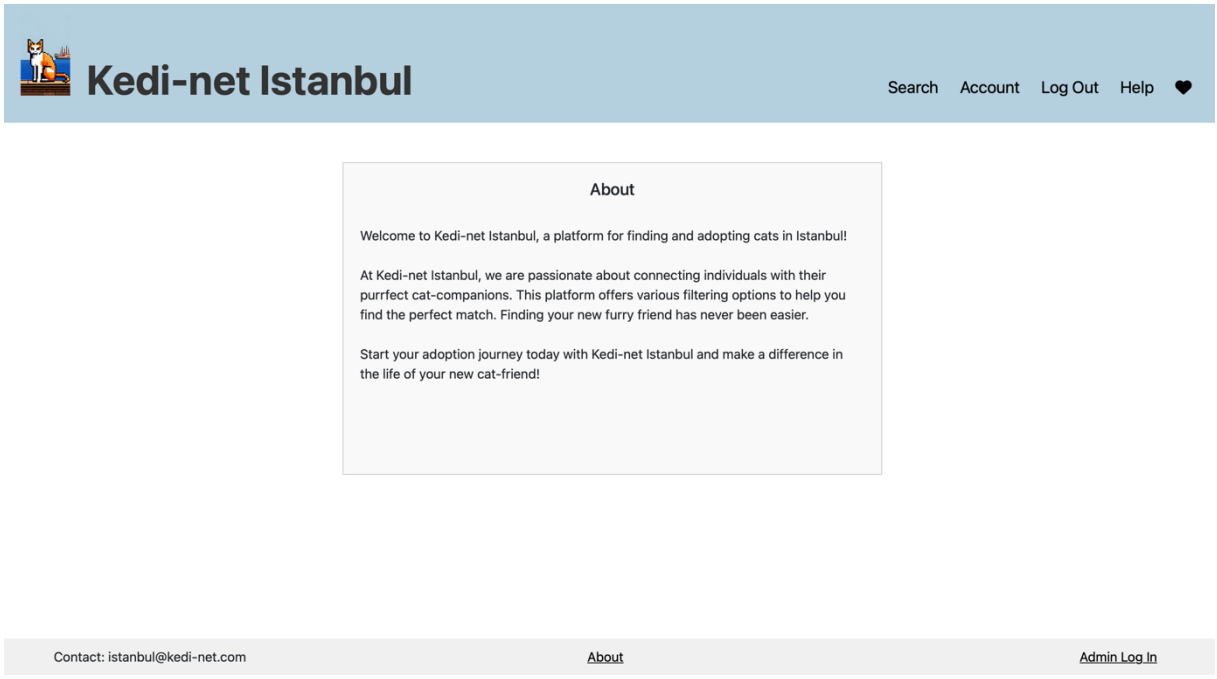

Figure 22 Final Frontend 'About Page'

4.2.7 Admin's Pages

The following three pages were also not implemented in the wireframes because the idea for admin functions was added later. The admin can log in, as shown in Figure 23. The design is the same as for the user login. They then get to the edit page (Figure 24), where they see an overview of the cats that belong to their animal shelter, and they can delete them here from the list. What's particularly interesting here is that the navigation bar has changed and now shows admin-specific navigations. The title also states that these are now pages of 'Kedi-net Istanbul' that are only accessible to admins. Lastly, in Figure 25, we see the add cats page, where the admin can fill out a form to add a cat.

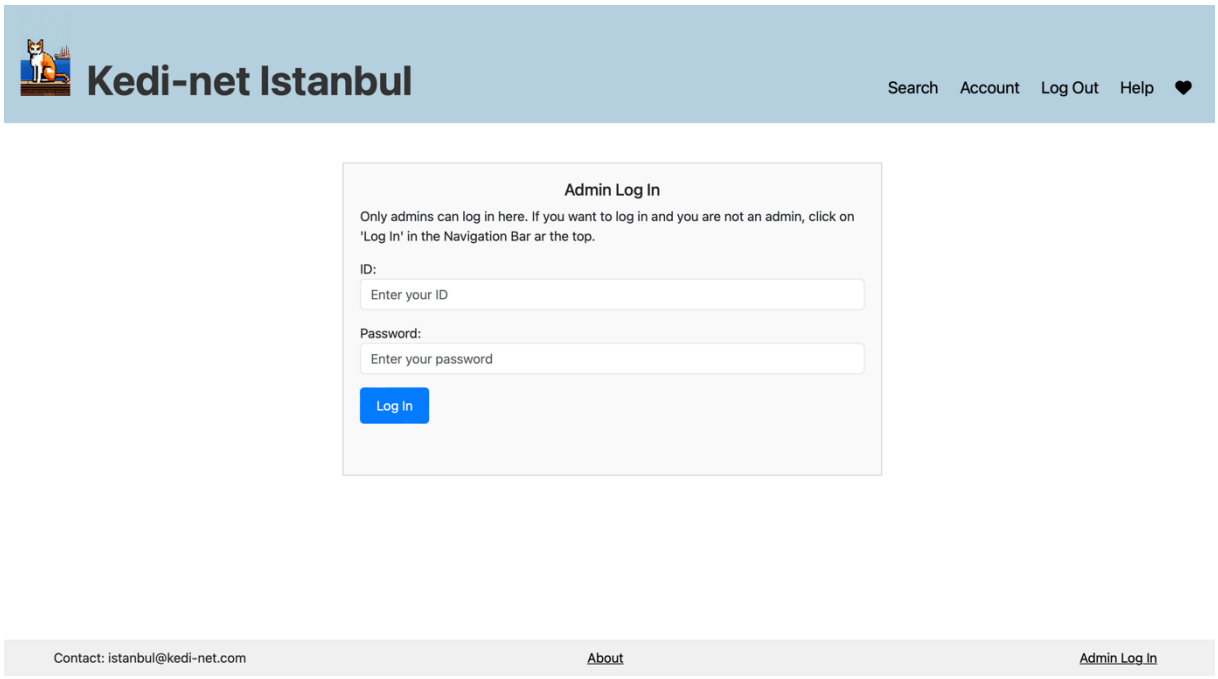

Figure 23 Final Frontend 'Admin Login Page'

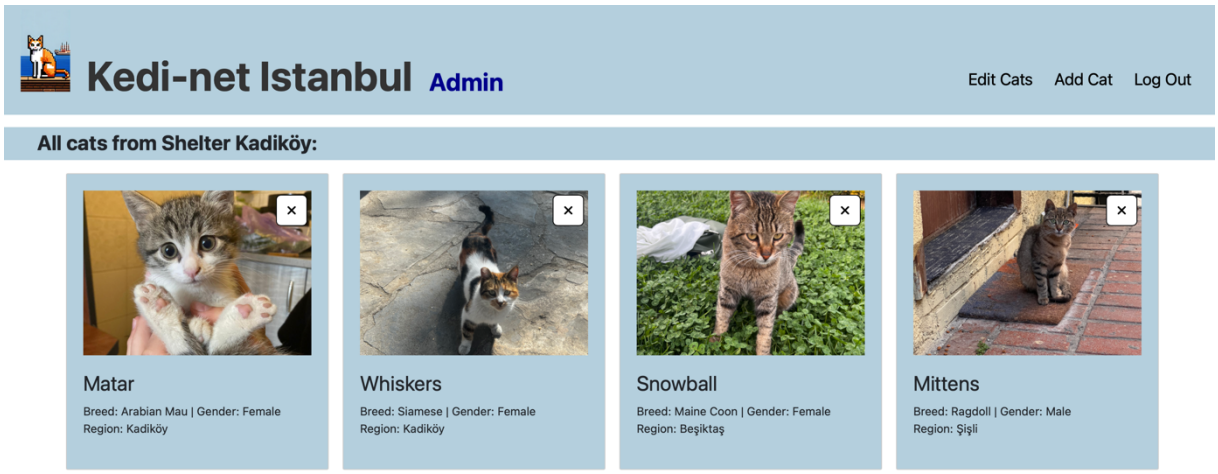

Figure 24 Final Frontend 'Edit Cats Page'

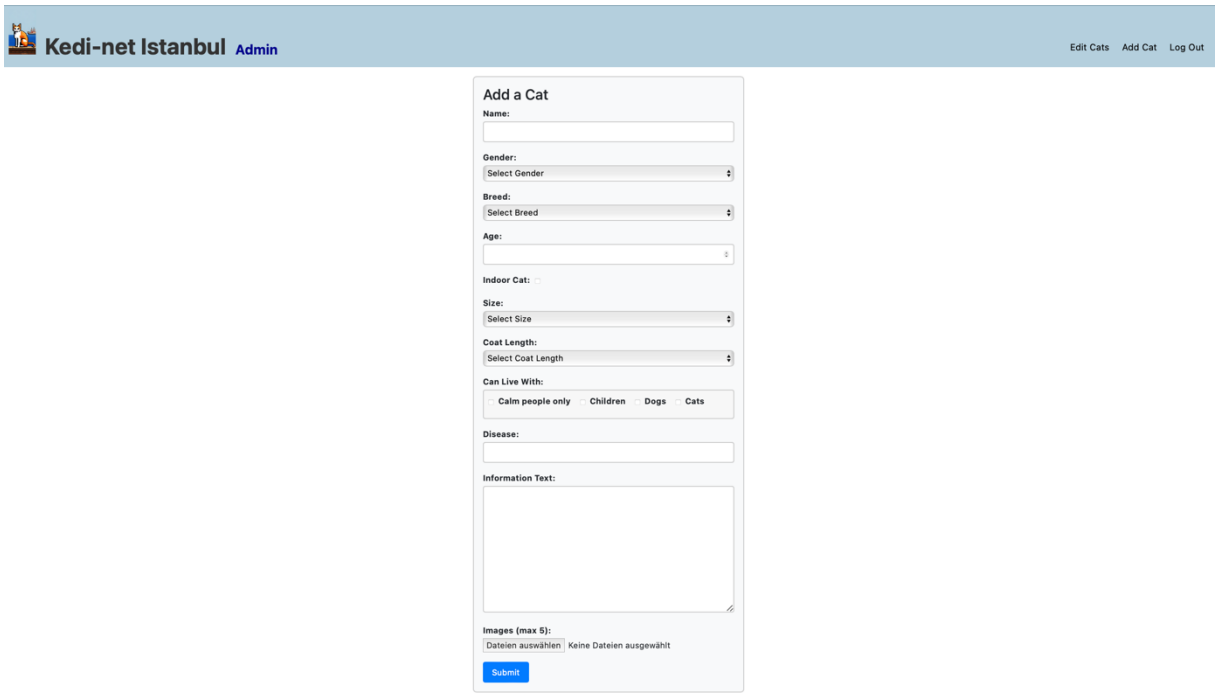

Figure 25 Final Frontend 'Add Cats Page'

4.3 Phase #2 Conclusion

In phase #2, I managed to identify both functional and non-functional requirements for my project. Additionally, I have written down the use cases of my application in detail and created

appropriate use case diagrams. I have also completed the frontend design so that I can now concentrate fully on the backend. All of this will help me in the subsequent phase #3 to be able to work on my backend development in a targeted and organized manner.

5 PHASE #3 - CLASS DIAGRAM AND ER MODEL

System modelling is an important part of the software engineering process by representing the system clearly and abstractly [2]. In this way, we can gain an overview of the organisation of a system. We can use structural models to represent the organisation of a system by showing the components that make up the system and the relationships between them [2]. I will present two structural models in the following, that I generated using PlantUML code.

5.1 Fine-grained Class Diagram

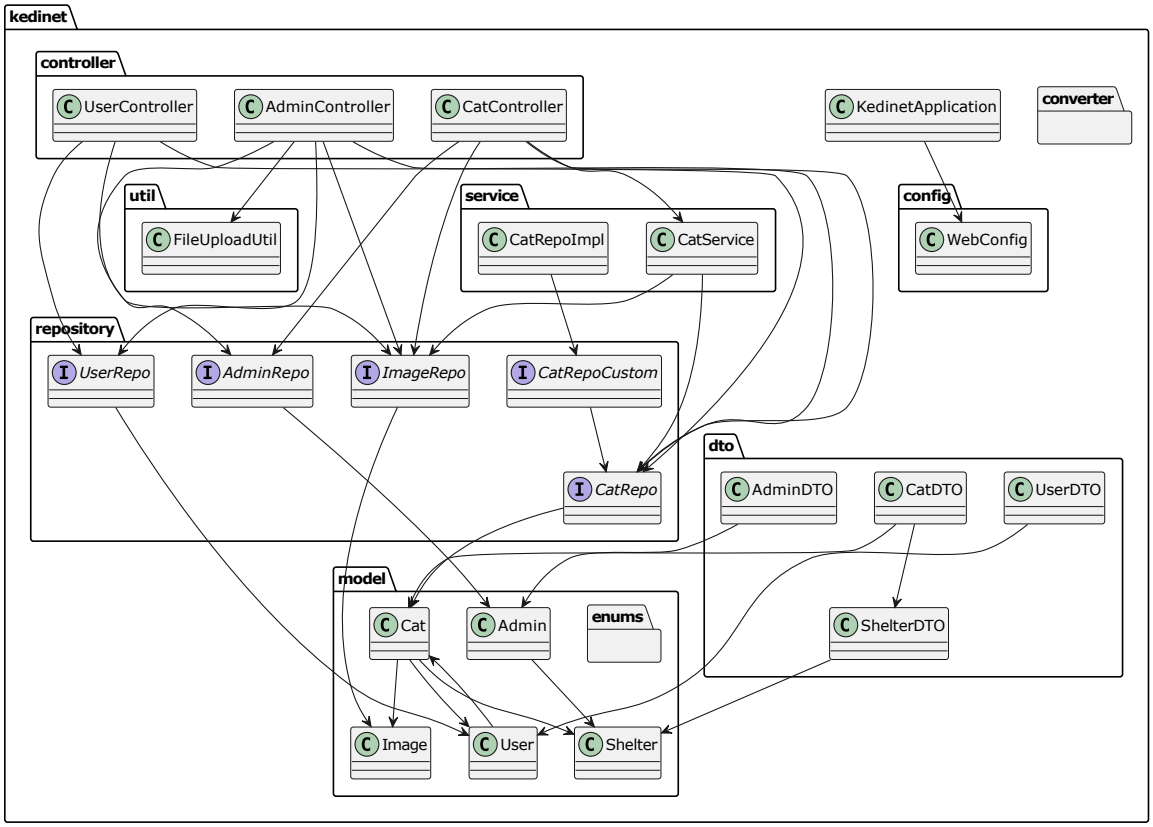

Figure 26 Coarse-Grained Class Diagram

Class diagrams are the most useful type of structural model for representing object-orientated systems [2]. They show the classes of a system and the associations between these classes. In Figure 26, you can see an overview of all the classes in my coarse-grained class diagram. My fine-grained class diagram includes not only the class names but also the attributes and the operations for every class of my backend. It can be found as an appendix to this

document since it is too big to be showcased here in one piece. Furthermore, I have left out a few classes so that the diagram is not too full. These classes are all enums that are used to store attributes for the models, as well as a utility class that controls the storage of images, but otherwise has no relationship to the other classes. Furthermore, the actual programme root 'KedinetApplication', which contains the main function, does not appear here as well as a WebConfiguration class that deals with formatting. I have also ignored seven converter classes in my class diagram, which perform such formatting operations from strings to the respective enums.

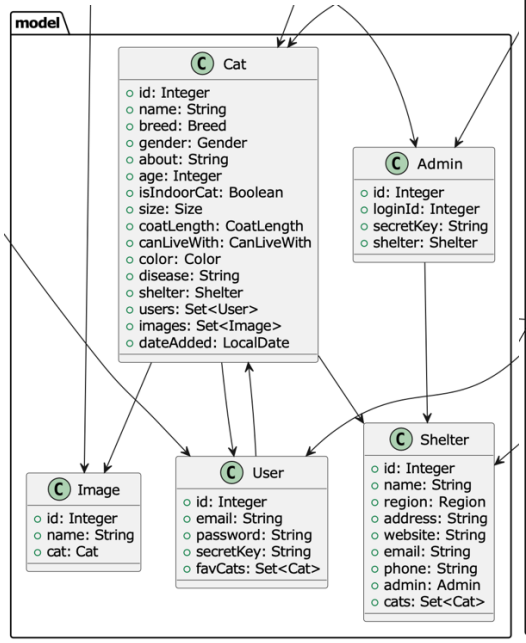

Figure 27 Model Package

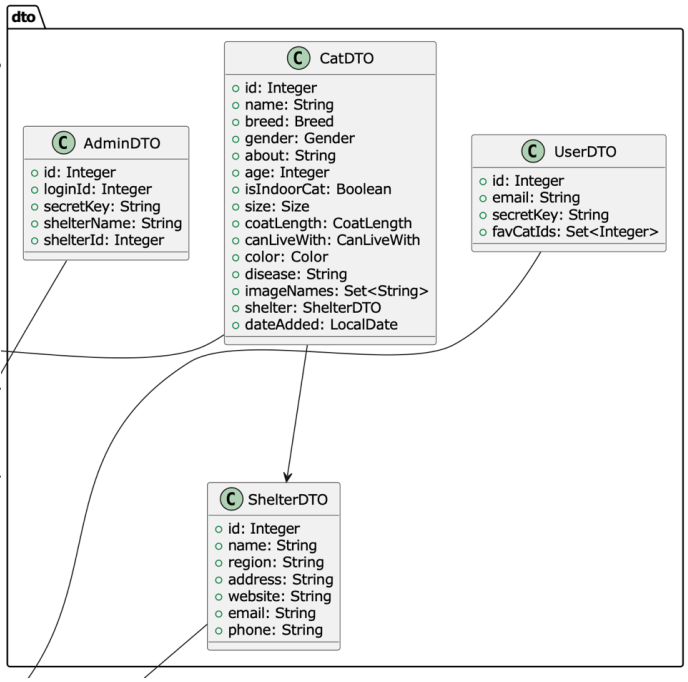

Figure 28 DTO Package

Let us now turn to the classes that are contained in the fine-grained class diagram. At the heart of the system, we have the model package, seen in Figure 27, with our model classes: this includes the classes for Cat, Admin, Shelter, User and Image. The classes are interrelated. I will go into this in more detail in the next part, as these classes are an exact representation of the entities in the relational database. All model classes have different attributes, but no associated getters and setters, as Spring Boot generates these and thus boilerplate code can be avoided. Next, we have four Data Transfer Objects (DTO) for Cat, User, Admin and Shelter as presented in Figure 28. They are created using a constructor that gets an associated class object and sets the attributes belonging to the DTO using this object. I use these DTOs to send only the relevant

data to the frontend, which is why a DTO does not contain all the attributes that the associated class has.

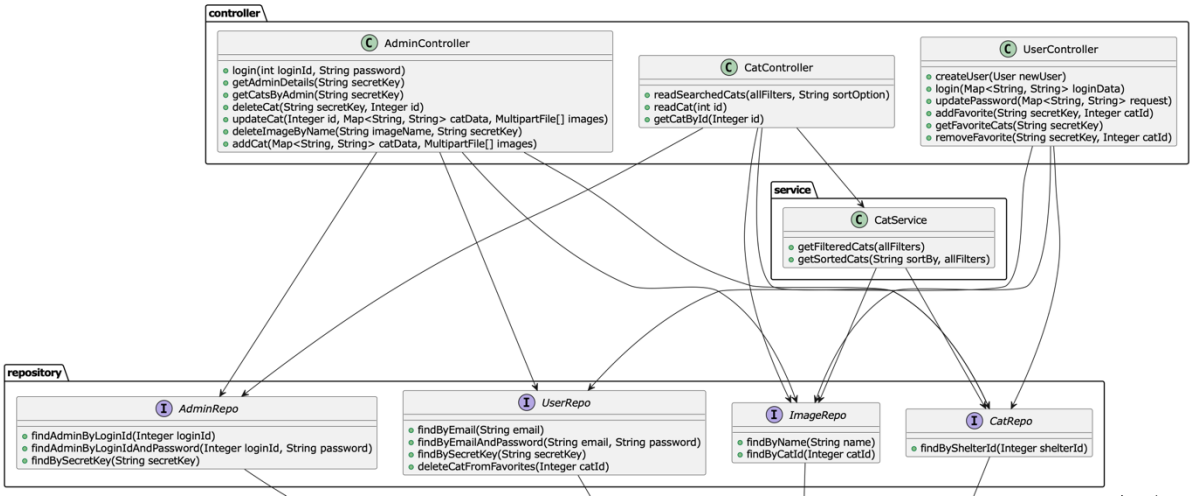

Figure 29 Controller, Service & Repository Packages

The next packages can be seen in Figure 29, starting with the Repository package. This contains the repositories: an admin repository, a user repository, an image repository and a cat repository. The repositories contain methods for interacting with the database. By using a Spring Boot annotation, a lot of boilerplate code can be avoided here again. Additionally, basic CRUD operations can be carried out on the database. Another important component of Java Spring Boot is the controller. I have a Controller package with one controller for Admin operations, one for Cat operations, and one for User operations. The controller acts as an intermediary between the view, so my frontend, and the model, so my data and the logic including the related repositories. The controller has the task of handling specific HTTP requests for Admin, User and Cat. Finally, the Service package contains the business logic of my application that goes beyond HTTP requests and responses. It acts as an intermediary between the controller layer and the repository layer. I use the Cat Service class to find cats that match certain criteria using the “getFilteredCats” method and to sort these cats using the “getSortedCats” method. The Service class has access to the Cat repository and to the Image repository to get the requested data from the database. The Cat service is called when the search or filtering of cats is requested via an HTTP request via the Cat Controller.

Last but not least, it must be noted here, that the handed-over parameter “allFilters” stands actually for “Integer ageFrom, Integer ageTo, List<Breed> breeds, List<CanLiveWith> canLiveWithList, List<CoatLength> coatLengths, List<Color> colors, List<Gender> genders,

Boolean isIndoorCat, List<Region> regions, List<Size> sizes". This simplification was used here to save space and generate more clarity. The correct representation can be viewed in the attached fine-grained class diagram.

5.2 ER Model

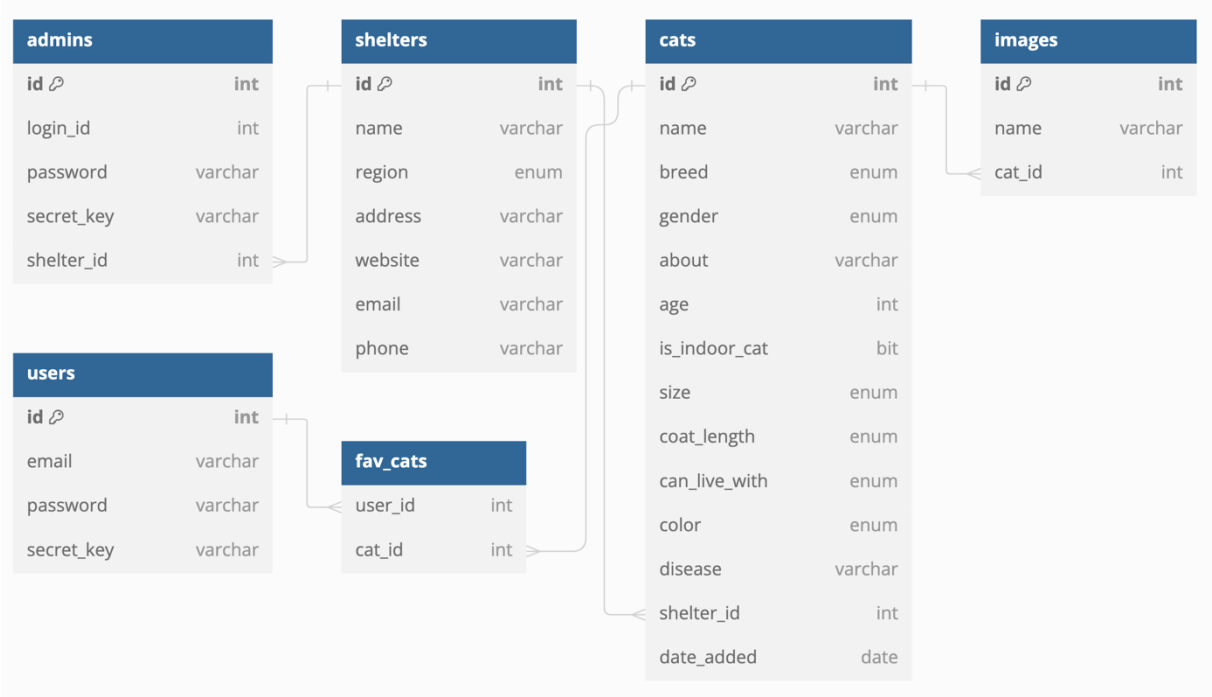

Figure 30 ER Model

Entity-relationship (ER) models can be used to represent databases. In my case, my database corresponds to the model part of the class diagram since I am using ORM. As can be seen in Figure 30, my database consists of six tables. The cats' table is in the middle. A cat entity has various attributes representing all the important information about this cat. It also has a unique ID as the primary key. This is used as a foreign key in the images table. A cat can have several images that are used on the kedinet-website to represent this cat. Each image entity also has a unique ID as the primary key; the only other attribute is the name of the image. The image itself is not stored here but is later retrieved using the image's name. Let's go back to the cats' table, which also contains a foreign key, namely the shelter ID. There can be several cats in a shelter. The shelter table also has an ID as the primary key and other attributes that could be relevant for a user of the kedinet-website. A shelter can have multiple admins assigned to it, who are then responsible for the cats in the database in the corresponding shelter. The admins'

table contains the shelter ID as a foreign key, an ID as a primary key and attributes that are relevant for logging in and for User authentication, like a unique e-mail address and a unique secret key. There is also a users table with a unique ID as the primary key and, similar to the admins table, attributes for login and user authentication. Furthermore, a user can add cats to a favourites list. We can see this connection in the fav_cats table. This table exists because otherwise, this would be a many-to-many relationship since a user can add several cats to their favourites list and at the same time a cat can be added to lists by several users. The fav_cats table contains the user_id and the cat_id as a solution and can therefore suitably map the many-to-many relationship.

6 PHASE #3 – SEQUENCE DIAGRAMS

Interaction diagrams can be used to specify interactions in a system. The sequence diagram is a type of interaction diagram that focuses on the sequences of interaction for a specific task [2]. That way we can model the dynamic behaviour of a system. In the following, I will present five different sequence diagrams that represent essential processes in my system.

6.1 Admin Login Sequence Diagram

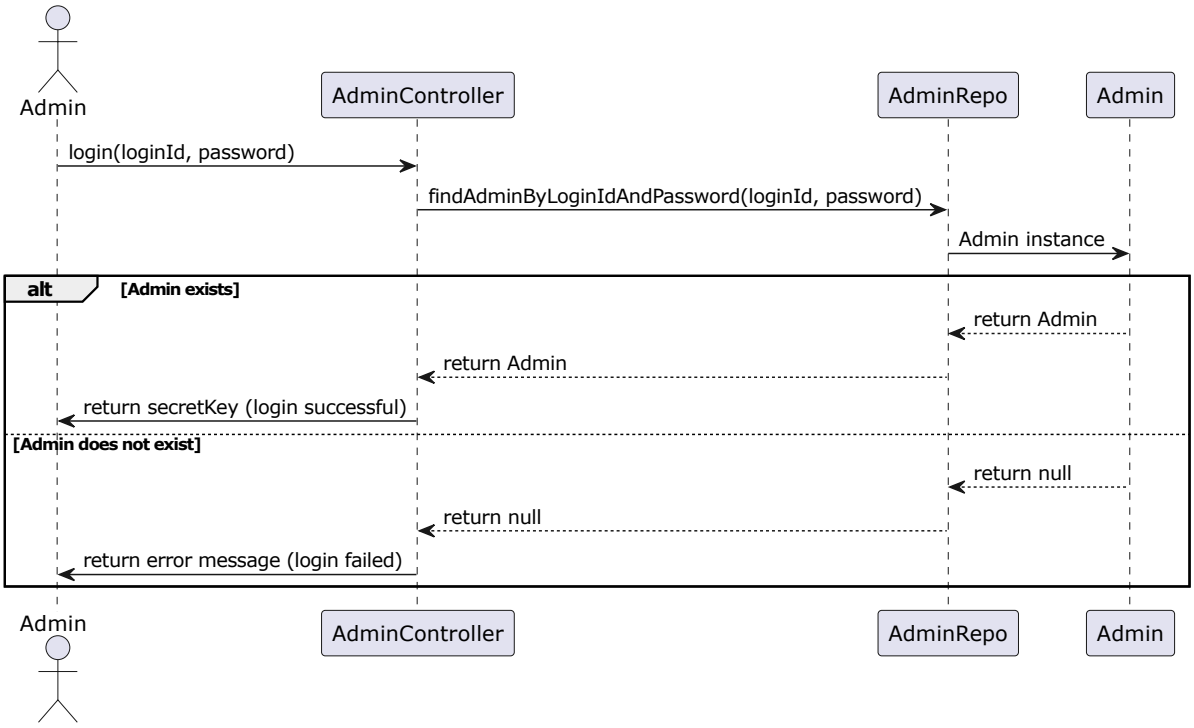

Figure 31 Admin Login Sequence Diagram

In the sequence diagram presented in Figure 31, I show the process of an admin logging into the system. The process begins with the admin typing in their login ID and password, which is sent as an HTTP Get request to the Admin Controller in the backend. The next step is verifying that the login ID and password exist and match so that the user can log in. This is done by the Admin Repository, which handles tasks concerning the database. If the login ID and the password exist and match, the admin exists in the database and is returned to the Admin Controller. After that, the admin gets an HTTP response containing the secret key of that admin which is needed to log in and for the admin to be further authenticated on the website. The

admin has successfully logged in now. Alternatively, in the case that either the login ID and the password don't match or that they don't exist, null is returned. That means the admin could not be found in the database and hence doesn't exist. The login fails and only an error message is returned to the frontend.

6.2 Get Cats by Admin Sequence Diagram

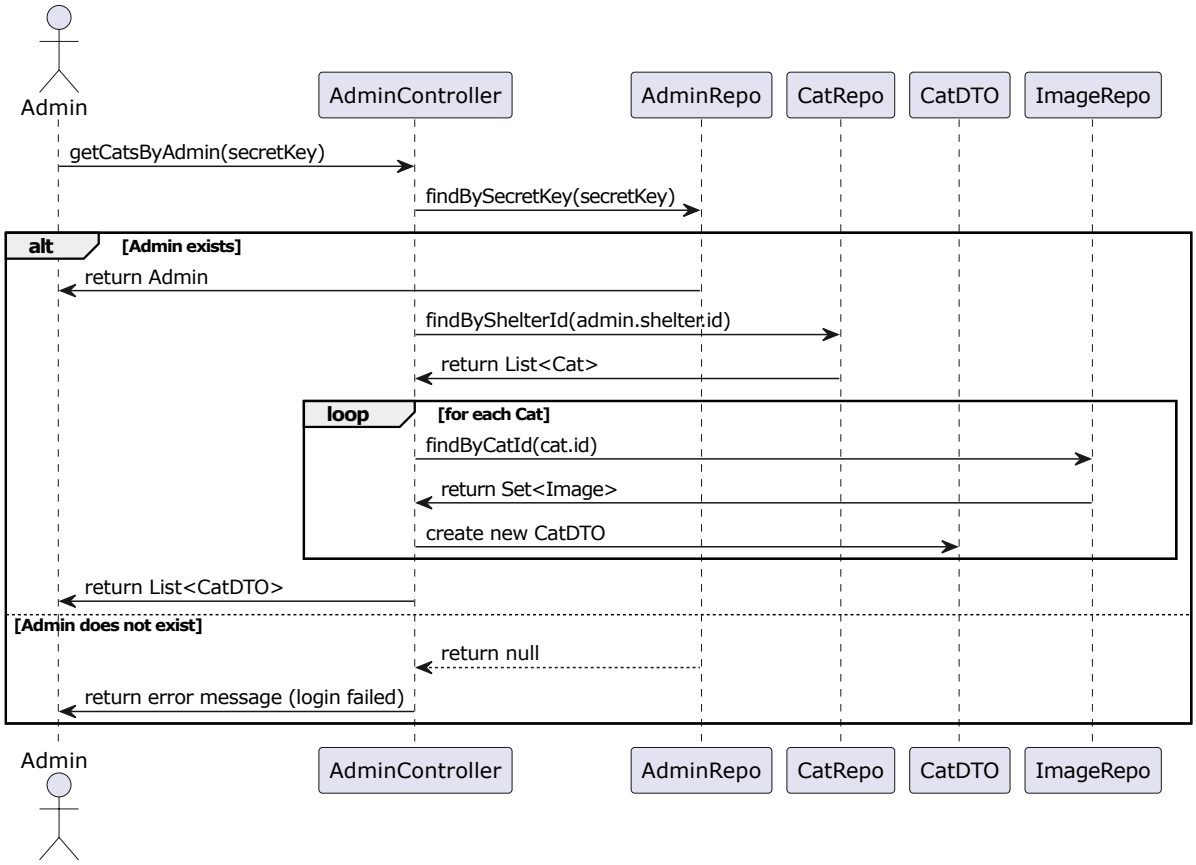

Figure 32 Get Cats by Admin Sequence Diagram

Next, we look at Figure 32, which shows the process of getting all cats in a shelter for which the admin is responsible and finally sending them to the frontend. This process begins with an HTTP Get request, handing over the secret key of the admin to the Admin Controller in the backend. The Admin Repository then helps to find the admin matching this secret key. This is done for every interaction of the admin with the database to ensure admin authentication. This authentication process always sends back the admin as a response, if the secret key is valid. If it is not valid, an error message is returned. Otherwise, the shelter ID of the shelter associated with the admin is handed over to the Cat Repository. The Cat Repository returns a list of all the

cats in this shelter to the Admin Controller. For each cat, a set of all associated images is now requested, which is returned by the Image Repository. Then, a Cat DTO is created for every cat that can finally be sent back in a list of all Cat DTOs to the frontend to be displayed.

6.3 Add Cat Sequence Diagram

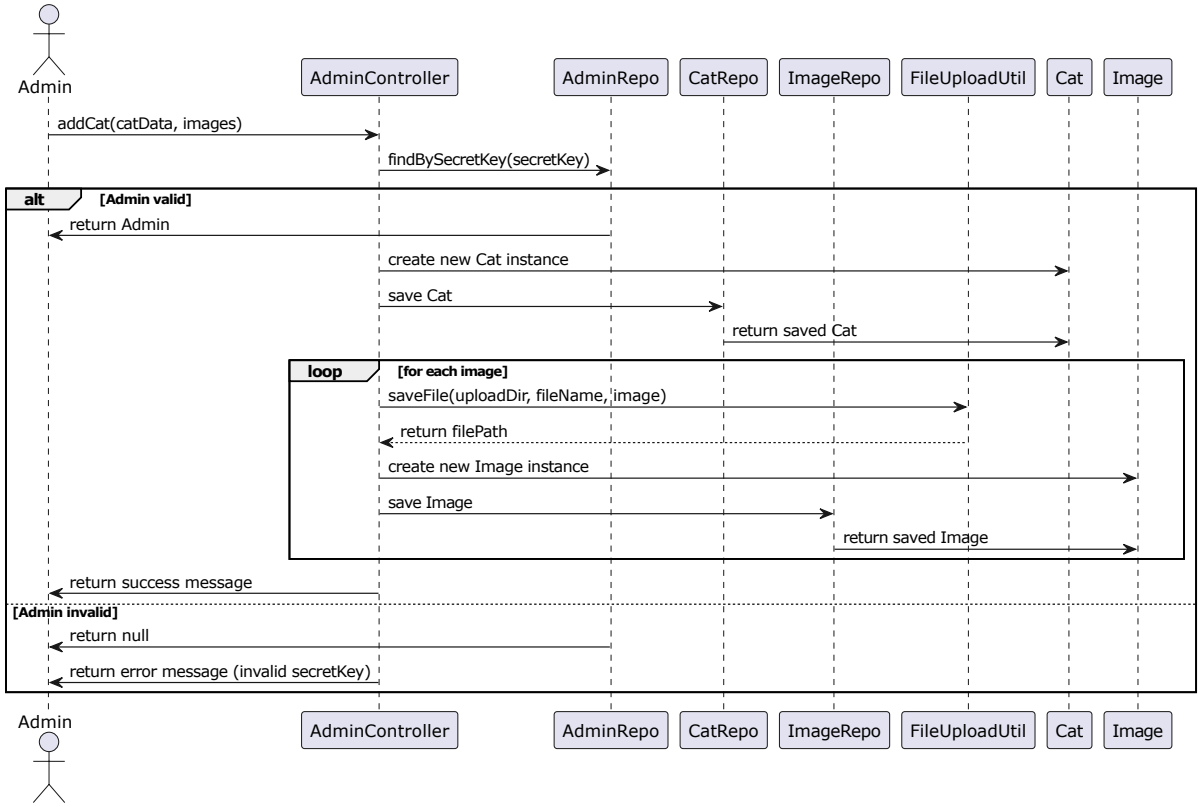

Figure 33 Add Cat Sequence Diagram

In Figure 33, we can see a sequence diagram showing how an admin can add a new cat in their shelter to the database. The admin can add a cat using a form in the frontend, and then the filled-in cat data and the images, that the admin added, are sent as an HTTP Post request to the Admin Controller in the backend. We see the same authentification process as in Figure 32 to check if the admin's secret key is valid. In the case that it is valid, the process of adding a cat can continue. The Admin Controller is responsible for creating a new Cat instance that is saved in the database, which is then done by the Cat Repository. Additionally, the images must be saved, which is done for each image separately in a loop. The images themselves are saved with the help of the File Upload Utility Class and then every representation of the image is saved in the database, only using the image name and not the actual image. That way the image can later

be accessed by putting together the path of the image folder with the title of the image. The process ends with the Admin Controller sending back a response to the frontend that the process was successful.

6.4 Search Cats Sequence Diagram

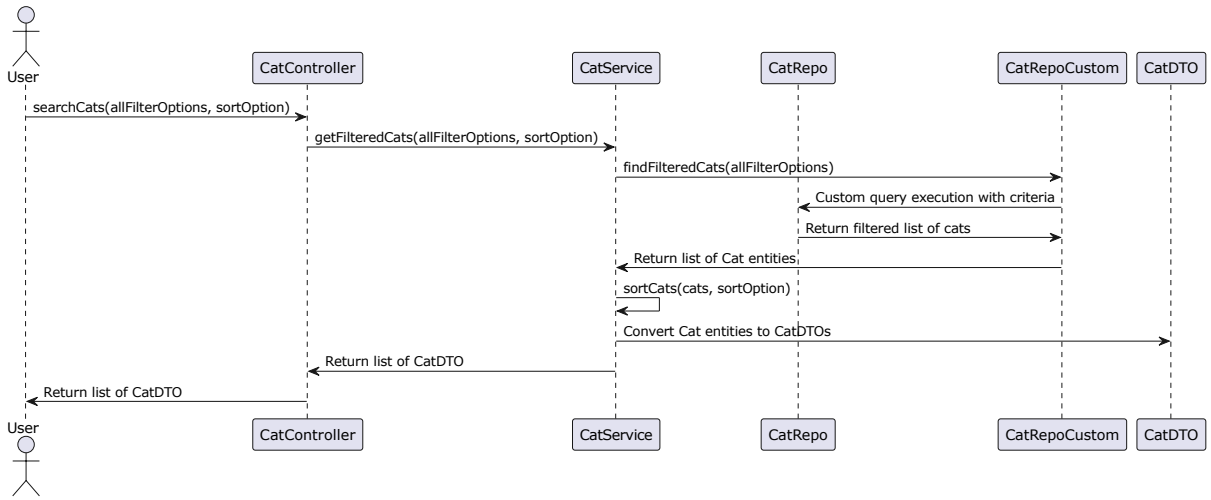

Figure 34 Search Cats Sequence Diagram

Now that we have covered the most essential admin functions, we can move on to another very basic function without an admin involved. Figure 34 shows how to search for cats that match certain filter options. Neither an admin nor a user needs to be logged in for this, so no authentication process is needed this time. First, all filter options are passed to the Cat Controller. This is shown here in a simplified form as “allFilterOptions”, but actually “ageFrom, ageTo, breeds, canLiveWithList, coatLengths, colors, genders, isIndoorCat, regions, sizes” are passed. It is also handed over how the search results should be sorted. Since getting only the cats matching the filtering options is a bit more complicated task, I used an extra Cat Service, providing business logic for this request. This again passes forward the filter options to an extra Cat Repository that can execute a query to receive only the matching cats. The customized Cat Repository “CatRepoCustom” functions as an extension of the Cat Repository implementing the “findFilteredCats” function. The Cat Service finally gets a list of all the matching cat entities and now sorts the cats according to the handed-over sort option. After that, the cat entities are converted to Cat DTOs to be returned to the Cat Controller and finally sent back as an HTTP response to the frontend.

6.5 User Adds Favourite Cat Sequence Diagram

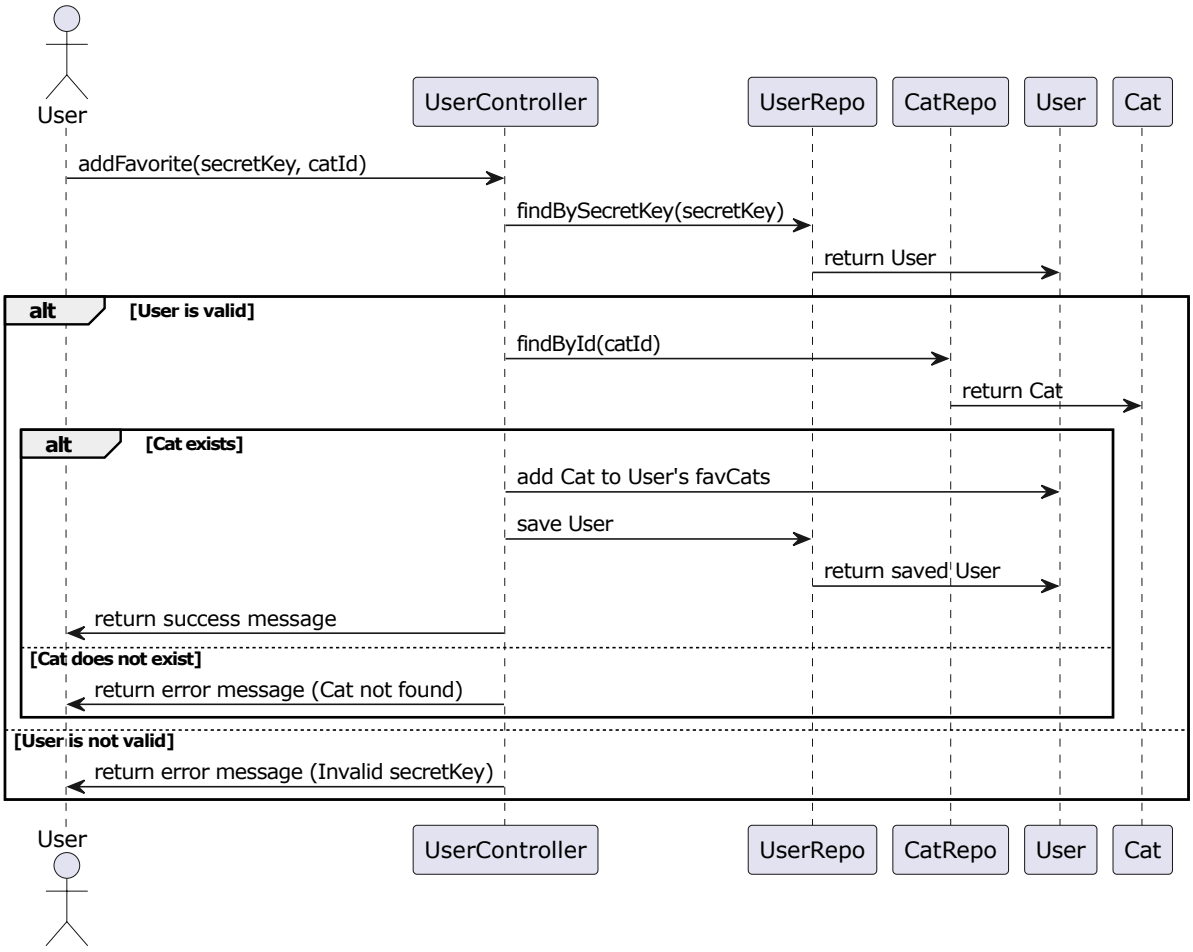

Figure 35 User Adds Favourite Cat Sequence Diagram

Now we come to the last sequence diagram in Figure 35, which shows the process of how a user adds a cat to their favourite cats list. First, as we saw previously with the admin, the user is authenticated. Here, too, an error message is sent back if the secret key of the user is not valid. If the secret key is valid, the User Controller checks whether the cat that the user wants to add to their favourites list exists. The Cat ID from the Cat Repository is used for this. Here, too, an error message is sent back to the frontend if the cat does not exist. Otherwise, the cat is added to the Favorite Cats list of that user. Then the user with the favourite cats list is saved. Finally, the User Controller sends a success message back to the frontend.

7 PHASE #3 – CONCLUSION

In phase #3, I managed to create diagrams for my backend and to finally develop the backend accordingly. I now have a frontend that can communicate with the backend via HTTP requests and a backend that can perform CRUD operations on the database. I have thus created a functional website that corresponds to what I had set out to achieve before. I tried not to repeat the disadvantages that we saw looking at three different similar websites in phase #1. Instead, I tried to recreate the advantages that we found there. I also implemented the requirements that I developed in phase #2. In phase #3, the Spring Boot framework allowed me to build an application in a very efficient way, without having to worry about dependencies and without having to write boilerplate code.

Of course, there are still things that can be improved. For example, I only realized late that it is not particularly sustainable to store the age of a cat in the database, as this changes every year. It would have been better to store the date of birth or even just the year of birth instead so that the age can then be generated and is always up to date. Security also needs to be worked on. Currently, both the user and admin passwords are stored in the database without any additional security measures, as is the secret key. This secret key is also not encrypted when passed back and forth between the frontend and backend. I am aware that it should never be programmed like this, but unfortunately, I did not have the time. This security concept could be revised in the future. Other functions could also be added to the website, such as showing how far a shelter is from the user's location. A map could also be integrated. However, for now, I am content with the result and with the fact that I have achieved what I set out to do. The final project can be found on GitHub at the following link: https://github.com/lenalarissa/kedinet.

REFERENCES

[1] *Why cats rule the streets of Istanbul?* Made in Turkey Tours, 2023. https://madeinturkeytours.com/why-cats-rule-the-streets-of-istanbul/

[2] Sommerville, I. *Software Engineering*. 10th Edition, Pearson Education Limited, Boston, 2016.

[3] Boogaard, K. *How to write SMART goals (with examples).* Work Life by Atlassian, 2024. https://www.atlassian.com/blog/productivity/how-to-write-smart-goals#:~:text=What%20are%20SMART%20goals%3F,within%20a%20certain%20time%20frame.

[4] Clark, J. *Top 10 Asp.Net Alternatives - Which is the best?* Back4App Blog, 2022. https://blog.back4app.com/asp-net-alternatives/#Spring_Boot_Framework

[5] *alpinejs vs react: Detailed NPM Packages Comparison | Performance, Security & Trends.*(n.d.). https://moiva.io/?npm=alpinejs+react#:~:text=React%20is%20a%20powerful%20library,need%20for%20complex%20state%20management.

[6] *Top 5 React.js frameworks for building modern web applications in 2023*. (n.d.). https://www.nan-labs.com/blog/reactjs-frameworks/

[7] *Next.Js by Vercel - the React framework.* (n.d.). https://nextjs.org/

[8] *JQuery Introduction.* (n.d.). https://www.w3schools.com/jquery/jquery_intro.

[9] Dhaduk, H. *Angular vs React: Which to Choose for Ymy Front End in 2024?* Simform - Product Engineering Company, 2024. https://www.simform.com/blog/angular-vs-react/#:~:text=React%20is%20a%20JavaScript%20library%2C%20whereas%20Angular%20is%20a%20TypeScript,is%20Angular%20different%20than%20React%3F

[10] *Software Engineering | Classification of Software Requirements*, 2023, https://www.geeksforgeeks.org/software-engineering-classification-of-software-requirements/

[11] *Writing User Requirements and System Specification Documents*, 2008, https://wiki.eecs.yorku.ca/course_archive/2008-09/F/4312/_media/requirementsspecsexample.pdf

[12] *Build fast, responsive sites with Bootstrap. https://getbootstrap.com*