PAW (Pet Adoption Website)

Bachelor's Thesis

Selin Karol – Taylan Tanyeri

20SOFT1023 – 19SOFT1010

Supervised by

Berke Özenç

January 2024

ABSTRACT

In our project, we aimed at highlighting the importance of disabled pets and how people find it uneasy about their adoption. They are often overlooked by people, due to their physical differences or sicknesses. This creates a hard process when it comes to placing them in their homes and forming a bond between people. That's why we dedicated this project to finding them a home, a friend and a safe place that they can be taken care of.

The solution of this case is to create a webpage for reaching as many people as possible through internet. We created an application in which these pets are put up for adoption for people. By including their medical documentation and habits, we are looking for the best matches, responsible people ready for their care and people who wish to put these pets up for adoption due to various reasons. This platform creates a place for not only pets and adopters, but also those who wish to extend their knowledge on the matter & donate to these pets in need. With the help of this platform, the disabled pets who are at a disadvantage when it comes to adoption, will have a chance to connect with people and find a home with full transparency and elaborative information when it comes to their disabilities.

ACKNOWLEDGEMENTS

We would like to thank our project supervisor Berke Özenç, who guided us during the process, supported us and pointed out the deficiencies of our project by giving constructive feedback, invaluable insights and continuous encouragement.

We are also grateful for Mertali Köprülü's advice and interest in our project. It has been a wonderful process with lots of learning and improvement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Web-based applications are preferred mostly for reaching as many people as possible, regarding a problem that needs to be announced, an advertisement for certain items or raising public awareness. In our inclusive pet adoption platform, we provide a network specifically for the disabled pets who get significantly less attention from people while having a harder time to exist and adapt to the conditions of life. In order to reach people, educate and highlight the importance of these lives, we have adapted the mission of connecting individuals with disabled pets, fostering a community build on empathy and understanding. It not only supports the disabled pets in need of care and happiness, but also it creates an opportunity for those who wish to provide a better life for their pets, by accessing people that are eligible for taking care of them. Through this platform, we aim at spotting trustworthy individuals and developing a safe community for all who wish to find people that will give the love the pet needs and find a life-long friend. By acknowledging the beauty of diversity, we decided to develop a website for the people and the animals to connect, develop friendship, and most importantly understand that every living being deserves love and care, no matter how hard the conditions are, and that disabilities and sicknesses are just a part of our lives, not differentiating us from others.

1.1. Problem Definition

Our project revolves around disabled pets and contributing them to the society by raising awareness and taking action via the platform we chose to develop. We know there are many street animals that do not have a quality life due to being born with illnesses or injuries. Moreover, there are people who cannot provide for them due to various reasons such as work, time complexities, money or transportation issues if a regular vet check is required for the animal. As a result of that, we decided to develop a website to find forever homes for these pets and create a bond between the owner and the pet which will be no different from a pet that has no disability.

1.1.1. Responsibilities

Our project is aimed at providing a safe space for disabled or sick animals to connect with people. To reach this goal, there are some responsibilities we need to consider:

- The platform may establish confidence and trustworthiness by providing a verification procedure for every participant, whether they are caretakers, or future adopters. This is necessary for the accuracy of the pets, their information and for users that will get into

the adoption process. The verification process requires the social security number of the users, managed by admin(s).

- User information must be secured, the passwords are encrypted. The caretaker of the interested pet and the admins may see the user profile and information, which include email, first and last name, added pets, address, phone number and some additional questions regarding the eligibility of hosting a pet.
- A limitation when it comes to adoption and adding pets is required. This is for preservation of pet information and accuracy of the process. Other than that, visitors are eligible for displaying pet information except the medical documents and location. This is for information security purposes.
- The platform should include a section defining the problem and possible solutions for it. We decided to create a page regarding the issue, to reach people and raise awareness.
- To monitor the adoption process, admins need to regularly check the eligibility of users. There needs to be multiple steps of verification when it comes to adoption. The first step needs to be admin's approval, while the next one needs to be the actual caretakers.

1.1.2. Challenges

- Processing the verification of the users was a challenge. Defining access areas of each user type, executing the implementation and handling control mechanism. We had to check these steps each time we made any changes in the project.
- We wanted to handle some pages using email verification such as Admin's verification & request page, notifications page. But it took too much time and we had to move this part to the backlog.
- General knowledge of Flask was not our profession, so we had some general syntax errors, logic errors, and wrong library implementations. This was the first project that both of us used Flask.
- Database implementation also had some challenges. We wanted to use SSMS instead of SQLite. We were not able to pull it off, so we continued with SQLite.

1.2. Project Scope

This project aims to raise awareness on animals or pets with special conditions. There can be many different situations such as disabilities, illness or some other special cases. Animals with those conditions don't have much chance of surviving on the streets or in the wild. To address those problems, we decided to design a website with special pets.

Biçimlendirdi: Yazı tipi:

Biçimlendirdi: Yazı tipi:

1.2.1. Functional Scope

Our pet adoption website's functional scope includes a comprehensive set of features designed to provide users with a smooth and satisfying experience, while also providing for the distinct requirements of potential adopters, shelters, and impaired animals. We have included some features for the users to perform actions easily and efficiently. Such features can be listed as:

- Registration for expanding realm of authorities of users,
- Search and filtering options for better search,
- Pet profiles for detailed view of the pet,
- Adoption application for starting the journey of adoption,
- Verifications for adoptions and safety of the platform,
- Donation page for helping shelters,
- Admin panel for tracking user status and activities to ensure security,
- Questionnaire for finding a suitable pet,
- Favoring pets to not lose sight of them.

1.2.2. Technical Boundaries

When defining the technical bounds of our pet adoption website, it is critical to acknowledge and handle some technological constraints that may restrict the project's scope. We have developed a platform compatible with various browsers for reaching wider audiences for a better user experience. We prioritized compatibility with major browsers.

The responsive design of the website is adaptable for screens with different viewports. Although it was challenging to optimize the design to different screen sizes, visual representation of the page is adaptable for various devices.

Moreover, security is an important concern that we took note of. In order to protect user data, we have used encryption for user passwords when saving to database. We have also restricted access to it, but absolute immunity from unforeseen security risks cannot be guaranteed.

Our website incorporates third party services and APIs. To enhance some functionalities, we used location services of OpenLayers, Flask framework with Bcrypt for securely hashing

passwords, Login and SQLAlchemy via SQLite extension of VS Code for the database. Our terminal commands are below:

```
pip install Flask
pip install Flask-SQLAlchemy
pip install Flask-Login
pip install Flask-Bcrypt
npm install ol
```

Figure 1.1 Terminal Commands

1.2.3. User Groups and Stakeholders

Formations like municipal shelters might not be able to meet the needs of those animals with special conditions or there might be too many animals for one shelter. Those situations will hinder the services of animals. So authorized workers of those shelters will come to us to decrease their workload by entrusting their special animals to us.

1.2.4. Resource Constraints

We had some budget constraints on implementing some functionalities. We wanted to use Google Maps API, but this service required billing from us. So instead, we continued with Open Layers API. Different document requirements also constrained us from time to time. Both developers were working on other jobs during the project's development. Those were our time constraints. Also, the lack of knowledge on Flask and python slowed us a bit during development.

1.2.5. Dependencies

We need to use some tools and technologies for running and developing this project. Firstly, we need to download Python and Flask framework. Our version of these tools can be seen below:

```
PS C:\Users\Selin\Desktop\project_backup 2.0> python --version
>>
Python 3.10.6
```

Figure 1.2 Python Version

```
PS C:\Users\Selin\Desktop\project_backup 2.0> pip show flask
>>
Name: Flask
Version: 3.0.0
Summary: A simple framework for building complex web applications.
```

Figure *1.3 Flask Version*

Jinja2 and Werkzeug are also demanded by Flask-Bcrypt, Flask-Login, Flask-Mail, Flask-SQLAlchemy. Jinja2 is a template engine for Python for rendering dynamic HTML pages. Werkzeug is a library for python, handling low-level details of processing HTTP requests and responses.

```
Name: Jinja2
Version: 3.1.2
Summary: A very fast and expressive template engine.
Home-page: https://palletsprojects.com/p/jinja/
Author: Armin Ronacher
Author-email: armin.ronacher@active-4.com
License: BSD-3-Clause
```

Figure 1.4 Jinja2 Version

```
Name: Werkzeug
Version: 3.0.1
Summary: The comprehensive WSGI web application library.
```

Figure 1.5 Werkzeug Version

We have installed SQLite extension for VS Code to work with databases and use SQLAlchemy.

Furthermore, our maps API structure from OpenLayers is also another dependency for frontend mapping to manage interactive maps.

```
PS C:\Users\Selin\Desktop\project_backup 2.0> npm show ol version
>>
8.2.0
```

Figure 1.6 OpenLayers Version

In order to manage the JavaScript packages, we used Node.js with the version of:

```
PS C:\Users\Selin\Desktop\project_backup 2.0> node -v
>>
v18.17.0
```

Figure 1.7 Node.js Version

1.3. Objectives

Main objective is just like mentioned above to raise awareness. In development, our main objective is to create a functioning adoption process with different types of users. There are also login and signup functionalities, pet information view etc. Those objectives also bring the security and verification systems. Those two objectives were also implemented for our project, and both are crucial.

2. LITERATURE REVIEW

2.1. Similar Applications

2.1.1. Petfinder

Petfinder is the most similar project when we compare it with our project. It just like our website it has adoption process, pet pages, donations, shelter page and readings. The main difference here is that we just focused on animals with special conditions. In this website they focus on all types of pets.

2.1.2. PetRescue

PetRescue is also in a similar position with Petfinder. But there is some difference. PetRescue also focused on reading. There are a lot of readings on their website. This adds a new functionality to the website. Users can search view pets, adopt pets or list pets. But also, they can just read the "news" or writings that are posted on this website.

2.1.3. ASPCA

This application has lots of articles and blogs on tips of having a pet. Also, in this website only shelters are providing pets for adoption. Users are not able to add pets, due to the aim of the platform being only for shelters. The main goal is to save pets from shelters. Users can view pets and search pets, read articles and donate to shelters. They can search and view rescued pets, and also volunteer for it.

2.1.4. Comparison of Similar Applications

Below, Table 2.1, you can find a summary and comparison of existing applications.

Table 2.1. Comparison of Pet Adoption Applications

	Petfinder	PetRescue	ASPCA
Pet Adoption/List	✓	✓	✓
Donation	✓	✓	✓
Readings	✓	✓	✓
Volunteering			✓
Rescued Pets	✓	✓	✓

2.2. Current Software Architecture

The existing architecture of the previous system is modular monolith. This means there is only one database that is tied with multiple modules. These modules work independently; therefore, the work is carried out in parallel. Separate modules enable the system to work without getting affected by the changes of other modules.

It is composed of three layers: a database layer that houses the data, an application layer that uses the programming languages Java, Python, or C#, and a presentation layer that displays the user interface made with HTML and CSS.

The client-side frontend element of the system is the web browser, which communicates with users, accepts input, and controls their interactions. The business logic is handled on the server side by the web server, also known as the backend component. It handles the requests that people make. The database server, which is the last layer, manages the data integrity and storage. As a result, this architecture allows another layer to access the data, preventing consumers from doing so directly.

As observed, the developers of the previous project used monoliths as an architecture, dividing the tasks into modules and using a single database, in which they stored pet information, registration and user profiles. To tie these tables with each other but also pursue them individually, this architecture type is used.

2.2.1. System Components

The system is divided into components, as mentioned, for less coupling and high cohesion. Such components can be listed as authentication module, pet module, search and filter subsystem, adoption process subsystem, administrative panel, data storage subsystem, user interface subsystem and so on. These modules and subsystems interact with each other, resulting in communication between the modules. The project is designed to be scalable, meaning that whenever a module is changed, the effect of this change should be minimized upon other modules. However, this is not always 100% achievable, as the modules are tied to each other. For example, registration module is tied with administrator module, as one manages the other. Furthermore, the administrator module inspects user activities and pet information, meaning they are all connected.

2.2.2. Frontend Technologies

For the existing project, the frontend technologies are Vue.js for core framework for building user interfaces, Axios for HTTP client for making asynchronous requests and ensuring

a connection with the backend server, CSS and HTML for design and Vue Material Design Icon Components for material design and icons.

2.2.3. Backend Technologies

Backend uses Express, providing a minimal Node.js web application framework. It also uses cookie-session for handling user sessions securely and Nodemon for reloading the server upon file changes.

2.2.4. Database Design

Pg is present in the technologies used for interfacing with PostgreSQL database, enabling communication between Express and the database. Therefore, PostgreSQL is used within the application, including the values of DB_HOST, DB_USER, DB_PASS and DB_NAME.

3. PROPOSED SYSTEM

This chapter is dedicated to our problems and solutions.

3.1. Introduction

Our software architecture is MVC design pattern. We included a model.py for database models, a view.py for functions and auth.py for controllers. The reason we chose this pattern is that it is used for web applications widely. It was easier for us to import necessary functions and models for these files and manage them. Updating the data and the changes, processing and accessing data were effective due to different layer of the architecture.

3.2. Graphical User Interface

We have used Bootstrap for front-end as a framework. It provides a collection of code for various templates, enables responsive design with respect to viewport height of the user screens and simplifies the development process due to having a pre-defined grid system. It is used with HTML, CSS and JavaScript for our work.

3.3. Business Logic

There needs to be some features that needs to be stored in the database as a business logic. A pet eligibility check is required by validating medical records in the database, the pet image needs to be taken from users to store. In the adoption process, the application information of the applicants needs to be delivered to admins and then caretakers via fetching from the storage. When it comes to backend implementation, these features are crucial. Moreover, user information is placed in the database and validation check is done according to that information plus some additional information required for identification. All these consist of data that needs to be processed within the application. The inputs have some constraints for users to ensure accurate and correct information, such as requiring some characters in passwords or limiting the characters in credit card information. After getting the input, the flow of data is transferred to the database, hashing and encrypting sensitive data. When it comes to access control, it is limited to admins to view user inputs. However, even admins are prevented from directly accessing the database for security reasons. In the case of exceptions or successful requests, users are informed through the system.

3.3.1. Login Services

We constructed an individual User class using flask_login. During the login process, this class is used to load users and represents user instances. The auth.py file safely stores the credentials for the admin account, including the username and password.

3.3.2. View Profile Services

User can reach to any pet's profile by clicking on their card. This section provides information about pet's name, age specie, breed, medical description, emotional details, location, PDF documents, Image etc.… Also, if the user that clicked on a pet is the owner of that pet, they can reach to edit pet, delete pet from this section. Other users can also favor this pet from pet view. If a user accepted another user's adoption request, then they can reach to each other profile for communication information and contacting with each other.

3.3.3. Questionnaire Services

The user-provided answers to our questionnaire, which is implemented in views.py, are used to calculate and produce the top two matches. To give an example, each pet receives points based on the user's selection of "angry" as their pet's emotion. For example, the pet gets three points if it is angry; otherwise, it gets zero points. Each pet goes through this process once more, after which their total points are determined. Ultimately, the user receives back the two pets that received the highest scores. Using this method, we can provide the user the pet matches that are best suited for them based on the answers to the questionnaire.

3.3.4. Pet Addition Services

Our website's "addpet" feature aims to make sure the accuracy and reliability of the pet-related data that users submit. On our project only verified users are allowed to add pets or adopt pets. Users submit a verification request to our admin and admins decide their status. Users are granted access to the "add pet" feature after approval. A verified user is required to fill out a form with information about their pet when they select the "add pet" option. Important details like the pet's name, age, breed are recorded on this form. Also, users must upload photos to address their pets.

3.3.5. Pet Modification Services

Users can change any part of their pet's information in the "Edit Pet" section. With this feature owners can maintain their pet's most recent status and contribute updated information to other users.

3.3.6. User Profile Services

View profile services include different functionalities like viewing pet profiles, user's favorite pets, user information, etc. There is also editing options like Edit Profile for editing name, surname, update password. Users can delete the profile of their own in this section. And also if the user is verified there will be indicator which shows that they are verified by admin. Adding pet is also done by clicking on "+" sign near pets from user profile section.

3.3.7. Adoption Services

In our platform, we offer adoption services for those verified by admin. A verified user is able to start adoption with a pet he/she wishes to adopt. For that, we need more information regarding the status of the potential adopter, including contact details and e-mail addresses. By collecting such information, admin accepts or declines the user request on adoption. Since it is a 2-step verification process for providing the best match for the pet, the pet caretaker also needs to approve of the request after the admin's approval. This way, the platform eliminates unnecessary and arbitrary requests before they reach to caretaker, improving user experience for them by also providing information pollution and congestion.

Users are able to see their adoption process from the notification dashboard, and caretakers can also view the requests from the same page. This provides easy access for all types of users.

3.3.8. Admin Panel Services

3.3.8.1. User Verification Services

For trustworthiness of the platform, only verified users (by admin) can add and adopt pets. This is tracked via social security numbers of the users. If they wish, they can be verified by sending their social security numbers to admins and proving who they are. After checking the information of such users, they get approved and officially verified.

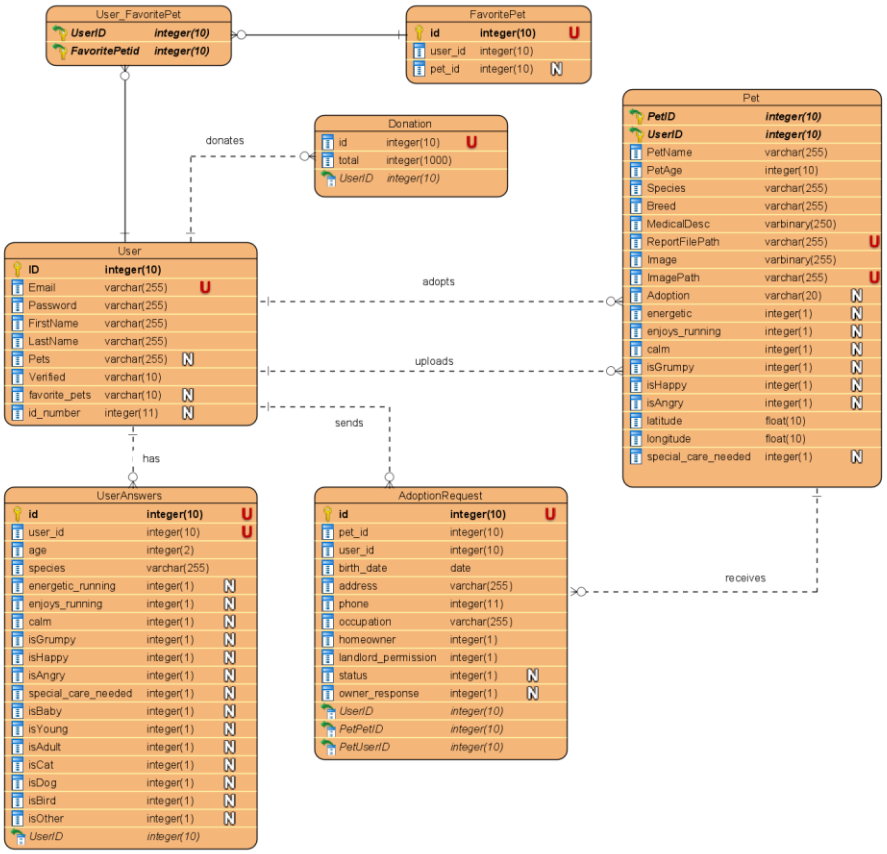
3.3.8.2. Adoption Decision Services

As it is explained in 3.3.7, admin firstly evaluates the user's social security numbers by checking the e-devlet. When the user is identified, admin approves the user, meaning the user is verified and identified by the platform.

3.3.8.3. Admin Pet & User Profile Services

Admin has the authority to check pet profiles and user profiles. Also, he/she can delete pets when needed. Moreover, the admin panel includes user types. Verified users, users, pets are seen independently in the panel for continuous monitoring of the platform and increase safety of information.

3.4. Data Management



Here, entity relationships of different tables can be observed.

4. IMPLEMENTATION, TESTS, EXPERIMENTS

This chapter covers implementation details and tests.

4.1. Implementation

In our implementation, we used some libraries and frameworks. As mentioned, we used bootstrap for frontend and flask for backend. We also used necessary imports for writing our codes and utilizing third-party libraries.

```
#IMPORTS-----------------------------------------------------------------------
from flask import Blueprint, render_template, request, flash, redirect, url_for
from flask_login import login_required, current_user, logout_user
from .models import UserAnswers
from .auth import save_pet_image
from .models import Pet
from .models import User
from .models import AdoptionRequest
from .models import Donation
from .models import FavoritePet
from . import db
import os
from sqlalchemy import desc
from flask import current_app
from flask import send_file
from werkzeug.utils import secure_filename
from .auth import save_pet_document
from flask_bcrypt import generate_password_hash,check_password_hash
```

Figure 4.1 Imports in View

Blueprint is used for defining views in Flask applications. For rendering and requesting access to data, we use render_template & redirect and request. In order to give feedback to users, we have flash. We generate urls for edpoints. Login and logout functions are for user authentication, keeping track of the logged in user is managed by current_user. All functions are maintained according to the current user.

In our models, we reach database tables for defining functions. Our models are named “UserAnswers” for questionnaire, “Pet” for adding pets and saving pet information, “AdoptionRequest” for collecting necessary data to check eligibility of adoption of the user, “Donation” for keeping track of total amount of donations and saving credit card info if user allows, “FavoritePet” for saving the pet in favorites. These models represent data entities in the application. We also reach database by importing it.

We used os to import operating system specific functionality for Python and desc from SQLAlchemy to sort query results. Current_app represents the application context, which is

necessary for implementing the functions. For sending files saving pet images and saving pet medical documents, we imported functions and saved the inputs to files in the directory for future use and responses.

Lastly, we needed to ensure security for user profiles and pets. For this reason, we generated password hash and made them encrypted while saving to the system. Sinc they are encrypted, we cannot access them directly. In the case of changing passwords, we used check_password_hash to reach the encrypted passwords and could change them without directly reaching the password information. The newly generated passwords are also hashed and encrypted, prohibited from unauthorized access.

In our init.py, we imported flask, SQLAlchemy and login manager to create the database and the application.

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from os import path
from flask_login import LoginManager
```

Figure 4.2 Imports in Init

4.2. Tests

In our application, we used User Acceptance Testing method, allowing real users to interact with the application. We prepared the test environment in our local storage and used the test scenarios in our Requirements Analysis Document. After this involvement, we have discussed the results of the tests with our supervisor and addressed our issues. Below, we can observe the results of our tests in RAD, and the reasons:

RAD 3.2.2 Scenario 1 FAIL → No condition filter

RAD 3.2.2 Scenario 2 FAIL → No option for adoption request cancel

RAD 3.2.2 Scenario 3 PASS

RAD 3.2.2 Scenario 4 PASS

RAD 3.2.2 Scenario 5 PASS

RAD 3.2.2 Scenario 6 PASS

RAD 3.2.2 Scenario 7 PASS

RAD 3.2.2 Scenario 8 PASS

RAD 3.2.2 Scenario 9 FAIL → Different way of approach

RAD 3.2.2 Scenario 10 PASS

RAD 3.3 Usability FAIL → No different languages other than that ok.

RAD 3.3 Reliability FAIL → Did not published for public use.

RAD 3.3 Performance PASS

RAD 3.3 Supportability FAIL → No support system

RAD 3.3 Implementation PASS

RAD 3.3 External Interface PASS

RAD 3.3 Packaging PASS

RAD 3.3 Legal PASS

RAD 3.3 Other constraints PASS

5. CONCLUSIONS AND FUTURE WORK

In our project, the client's goals have been effectively met. The development process followed a low-cost methodology, adhering to the pre-established financial limitations. The implementation of traceability of requirements has been carried out with great care, creating a clear link between system functions and customer needs. Because of the platform's rapid development design, it can be quickly adjusted to changing requirements and integrated continuously. Prioritizing completeness, all functions requested by the customer have been seamlessly integrated.

The project completed objectives for end users. To guard against unauthorized access, safety and security measures have been put in place. The platform places a high priority on user-friendliness, offering a simple to use interface that is easy to navigate. One strong point is usability, which makes it easy for users to interact with the system. Functionality visibility guarantees that users can quickly understand the system's current state.

The system's robust and stable performance was ensured by minimizing errors during operation during design. A notable feature of the code is how easily its components can be modified for use in different sections of the system. Regardless of those achievements, maintaining consistency across interfaces is still a problem that needs to be fixed in the future.

To advance our project, we plan to adopt more professional and dynamic solutions, moving beyond basic and static approaches. We also aim to sum up all features that have been discussed in a way that is consistent with our objectives. Another goal is to make the website public in order to increase accessibility for a larger group of people and promote community involvement.

6. REFERENCES

- "Adopt a Pet." *ASPCA*, 2015, www.aspca.org/adopt-pet. Accessed 21 Jan. 2024.
- Arköse, Tuğberk. "Software Architecture SOFT3205." Week 2 – 12 October 2022. SOFT3205 Lecture, 12 Oct. 2022, Istanbul, Turkey.
- Arköse, Tuğberk. "Software Architecture SOFT3205." Week 9 – 14 December 2022. SOFT3205 Lecture, 14 Dec. 2022, Istanbul, Turkey.
- Arköse, Tuğberk. "Software Architecture SOFT3205." Week 10 - 21 December 2022. SOFT3205 Lecture, 21 Dec. 2022, Istanbul, Turkey.
- "Is Bootstrap a Framework or a Library." www.lambdatest.com, www.lambdatest.com/software-testing-questions/is-bootstrap-a-framework-or-a-library. Accessed 21 Jan. 2024.
- "PetRescue - Create Happiness. Save Lives. - PetRescue." www.petrescue.com.au, www.petrescue.com.au/. Accessed 21 Jan. 2024.
- "Petfinder." Petfinder, 2017, www.petfinder.com/. Accessed 21 Jan. 2024.
- Saraswathi, Ravi. "Four Architecture Choices for Application Development in the Digital Age." IBM Blog, 6 Jan. 2020, www.ibm.com/blog/four-architecture-choices-for-application-development/. Accessed 7 Nov. 2023
- "What Is Three-Tier Architecture." IBM, www.ibm.com/topics/three-tier-architecture#:~:text=A%20'layer'%20refers%20to%20a,separate%20from%20the%20other%2 0divisions. Accessed 9 Nov. 2023