Date: May 2, 2025 Document Version: 1.0

CS492: Senior Design Project II

Final Design Report

Group: T2443

yodai

Yasemin AKIN 22101782	Deniz Can ÖZDEMİR 22003854	Oğulcan KARAKOLLUKÇU 21802642	Alphan TULUKCU 22003500	İlhami ULUĞTÜRKKAN 22102546

Table of Contents

1. Introduction

After the recent legislation in Türkiye that mandates the removal of stray animals from public places and their placement in shelters, it is still urgent to enhance the adoption process to avoid overcrowding and euthanasia risks for these animals. In July 2024, this law was passed to address public safety concerns, but it has proved to be a source of tremendous difficulty for animal welfare organizations [1]. Our goal was to build a new platform to simplify adoption by strengthening the bond between potential adopters and the animals in need by providing comprehensive animal profiles, including health records and behavioral traits, personalized care recommendations, and most importantly, a sophisticated personalized stray animal recommendation system. The platform also provides adoption guidance and post-adoption support. Beyond providing a solution to the urgent problems created by the new law, this initiative also serves to support the long-term health of stray animals in Türkiye through the promotion of sustainable adoptions and strained shelter resources. This report aims to provide more detailed information about the culmination of the project, the final architecture and design of our system as well as the final status of the project.

1.1 Purpose of the system

The core purpose of Köpük is to simplify and optimize the process of matching potential adopters with animals in need of a loving home. The platform achieves this by creating in-depth, comprehensive animal profiles that include health records, vaccination histories, behavioral assessments, and personalized care recommendations. These detailed profiles provide adopters with a clear understanding of each animal's needs, enabling them to make well-informed decisions and be better prepared for pet ownership. A sophisticated recommendation-based matching algorithm further refines the process by aligning adopter lifestyles, preferences, and capabilities with the specific requirements of each animal, thereby fostering more compatible and successful adoptions. Beyond the initial match, the platform offers robust support features through an artificial intelligence-driven chatbot that assists users at every stage—from pre-adoption inquiries and the application process to post-adoption support for training and veterinary care. In doing so, Köpük not only reduces the risk of returned adoptions and alleviates the burden on overcrowded shelters but also advances the broader mission of animal welfare by ensuring that animals find permanent, loving homes while equipping adopters with the necessary tools and knowledge for responsible pet care.

1.2 Overview

This report details the final architecture and design of Köpük as well as the final status of the project, providing a comprehensive look at its refined requirements, final architecture, and system design. It describes the development and implementation processes, provides extensive testing results, and outlines the maintenance plan to ensure the platform's long-term success. Additionally, the report discusses ethical and professional responsibilities observed during the project, highlights the teamwork dynamics including contributions and leadership roles of each member, and evaluates how initial objectives were met. It also reflects on the new knowledge acquired and the strategies used for learning throughout the development cycle. Accompanied by the deliverable software system and an integrated User Manual, this report showcases the project's full lifecycle from inception to delivery, supporting its aim to transform the animal adoption process through innovative, AI-driven solutions.

2. Requirements Details

2.1 Functional Requirements

2.1.1 User Registration and Authentication

In *Köpük*, only adoption candidates are allowed to register themselves into the system. During registration, users are required to provide personal information such as their name, surname, age, address, and other relevant details, some of which are crucial for determining their eligibility for adoption. To ensure the integrity and reliability of the data provided, users must verify their information. Furthermore, an authentication mechanism is implemented to secure user accounts, adding an additional layer of protection against unauthorized access.

2.1.2 Deep Learning-Based Visual Breed Classification

A core feature of the system is the ability to classify the breed of dogs and cats from user-uploaded images using deep learning. To achieve this, three different models are trained and evaluated: a simple Convolutional Neural Network (CNN) designed from scratch, a MobileNet-based CNN trained from scratch, and a transfer learning model using a pre-trained network such as ResNet50 or VGG16. Comparative evaluations based on standard metrics guide the selection of the final model, ensuring the most accurate and efficient classifier is deployed within the application.

2.1.3 Recommendation System for Sustainable and Accurate Adoptions

The recommendation system is the centerpiece of *Köpük*, aiming to facilitate sustainable and appropriate pet adoptions. Market research revealed a gap in the use of recommendation algorithms in existing pet adoption platforms. To address this, a hybrid recommendation approach combining collaborative filtering and content-based filtering is adopted. User data such as preferences, behavioral patterns, and lifestyle information collected through forms and platform activities, along with animal data including breed, age, size, temperament, and medical needs, form the foundation for feature extraction and similarity measurements. The system dynamically adapts to user interactions, improving the quality of recommendations over time to support successful adoptions.

2.1.4 Quiz System

The quiz system is a system we created to inform users and help them better understand the responsibilities and requirements of adopting an animal friend. As a result of comprehensive meetings with our partners and mentors, we decided that having a quiz section in addition to the question and answer sections of the registration and animal adoption form could be tiring and overwhelming for the user. For this reason, we decided to remove the quiz section from the application. However, while removing the quiz section, we mixed the questions we wanted to ask as quizzes into the other two sections. This way, users do not have to answer questions in 3 different places and they can still understand their responsibilities and requirements.

2.1.5 Chatbot Support for Q&A

To support potential adopters and streamline communication, an intelligent chatbot has been developed as part of *Köpük*. Inspired by the real-world communication challenges observed by shelters such as Kurtaran Ev, the chatbot addresses frequent queries regarding animal care, health, and adoption procedures. The system is powered by a Large Language Model (LLM) trained on curated datasets, including shelter FAQs and verified information sources. Selected conversations may be publicly available as a community information resource. Long-term plans include developing a personalized veterinary assistant that tailors advice and guidance to individual user needs.

2.1.6 Notifications and Alerts

The application integrates a notification and alert system to keep users informed about the status of their adoption applications and other important events. Since some stages of the adoption process require manual intervention and approval by shelter administrators, timely updates are crucial. Furthermore, during the post-adoption phase, new adopters are required to submit evidence demonstrating ongoing animal welfare. The notification system helps manage these requirements through automated reminders and checkpoints, supporting compliance and communication between shelters and adopters.

2.2 Non-Functional Requirements

2.2.1 Usability

Given the wide demographic range of users, *Köpük* prioritizes ease of use and intuitive navigation. The application design ensures that users with minimal technical expertise can complete essential tasks such as registration, browsing pets, and submitting applications. Visual cues such as buttons, tooltips, and clear labels are consistently applied to enhance the user experience, guiding users step-by-step through the adoption process.

2.2.2 Reliability

Reliability is critical, particularly for core features such as the chatbot and the recommendation system. The chatbot is required to achieve at least a 90% correct response rate based on a curated set of common questions, ensuring user confidence in automated support. Similarly, the recommendation system is designed to achieve an approximately 80% precision rate in retrieving relevant pet matches, based on historical adoption archive data from Kurtaran Ev. Emphasis is placed on both the robustness and dynamic updating capabilities of these systems to adapt to changing conditions and user needs.

2.2.3 Performance

The platform's performance requirements emphasize low-latency access to media assets such as images and videos. All media content must load in under three seconds under normal network conditions. Techniques such as lazy loading, content compression, and optimized database queries are employed to enhance loading speeds and ensure smooth browsing experiences.

2.2.4 Supportability

Köpük is designed to be a cross-platform application to accommodate the varied device usage patterns of the target audience. Comprehensive documentation is maintained throughout the project, detailing both the design and the implementation processes. Furthermore, the platform incorporates an internal feedback system, allowing users to report bugs and issues, which can then be prioritized and addressed efficiently by the development team.

2.2.5 Scalability

Scalability is a major concern given the potential for the application to serve thousands of concurrent users. The system architecture is designed to handle increasing user loads gracefully, ensuring that core functionalities such as search, chat, and adoption application processes remain responsive even under high demand. The use of scalable cloud infrastructure and containerized microservices supports future growth without significant architectural changes.

2.2.6 Sustainability

To ensure financial sustainability, especially during the early stages of deployment, the project favors free or low-cost infrastructure solutions where possible. As user demand grows, migration plans are in place to transition to higher-tier paid services capable of supporting larger networks and more intensive computational needs, ensuring that the platform can continue to grow while maintaining cost-efficiency.

2.3 Pseudo Requirements

Several internal development practices and standards are established to enhance the project's success, even though they are not mandatory end-user requirements. Text within the application is crafted to be concise, clear, and user-friendly, minimizing confusion. Internal testing datasets for recommendation systems are designed to simulate real-world conditions as closely as possible, ensuring realistic validation of algorithms.

Error messages are carefully worded to be informative without resorting to technical jargon, enabling users to address common issues independently. The system architecture emphasizes modularity and maintainability, with clear documentation to facilitate debugging, future modifications, and onboarding of new developers. During the prototype phase, navigation flows are designed logically to accelerate testing and allow for efficient iterations, even if the final design sequence may evolve over time.

3. Final Architecture and Design Details

3.1 Overall System Architecture

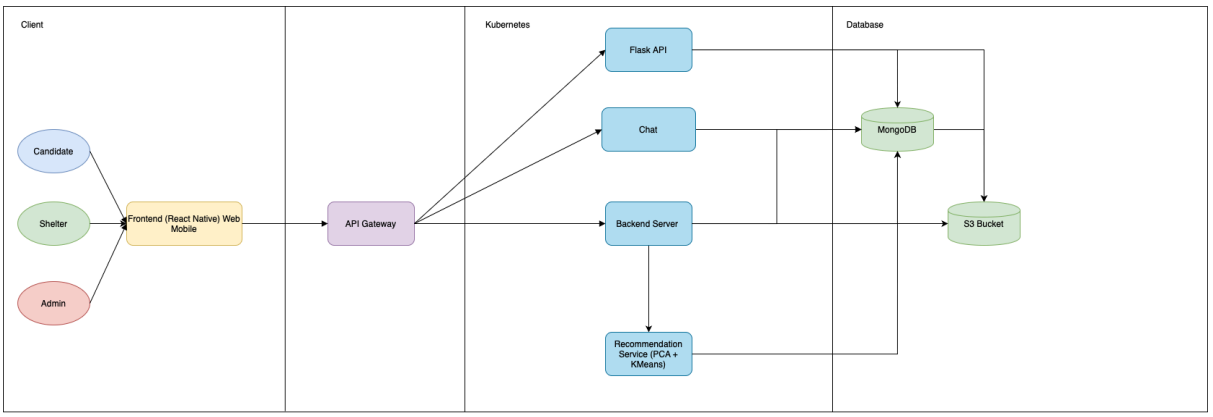

Figure 1: Microservice Architecture Diagram

Köpük was designed following a modular, service-oriented architecture to ensure scalability, maintainability, and cross-platform compatibility. The system is logically divided into three main layers: the client layer, the server/backend layer, and the cloud infrastructure layer.

At the client side, the mobile and web applications are built using the React Native Expo framework. This enables a single codebase to be deployed seamlessly across iOS, Android, and Web platforms, reducing development time while ensuring a consistent user experience. Users interact with the system through these applications, performing actions such as browsing pets, submitting adoption forms, and conversing with the *KöpükAI*.

The server/backend layer manages the business logic, user authentication, recommendation system, quiz handling, notifications, and communication with third-party services. Developed with Node.js and Express.js, the backend exposes

secure RESTful APIs consumed by the mobile and web clients. It also integrates with specialized modules, such as the deep learning-based breed classification service and the LangChain-powered chatbot service, each operating as independent microservices.

The cloud infrastructure layer is hosted primarily on Google Cloud Platform (GCP). Kubernetes (via Google Kubernetes Engine) orchestrates containerized backend services, managing their deployment, scaling, and health. MongoDB serves as the primary database, storing user data, pet profiles, chat histories, and application records, while AWS S3 handles all static media assets like images and videos. Google Cloud services like VPC, IAM, and Artifact Registry ensure secure networking, role management, and image storage for deployments. Monitoring, alerting, and logging are centralized through Google Cloud Operations Suite to maintain visibility across the system.

This multi-layered architecture ensures that the system remains modular, fault-tolerant, and scalable, capable of supporting future expansions such as the addition of payment systems for donations or multi-language support for a broader user base.

3.2 Mobile and Web Application Design

Figure 2: Cross-Platform UI Representation

The client-facing application was designed to prioritize usability and performance. React Native Expo provided the core framework, with Expo Router managing navigation across screens in a scalable, route-based structure. The user interface (UI) adopts a clean, minimalistic design philosophy, using simple layouts, intuitive icons, and consistent color schemes to cater to a wide range of user demographics. Forms, and pet browsing screens are optimized for touch interactions, ensuring a smooth experience even on lower-end mobile devices. Animations and visual transitions are used carefully to enhance user engagement without sacrificing performance. Security considerations were embedded at the client level, including secure storage of tokens using encrypted storage libraries and form validations to prevent invalid data submission. Accessibility features, such as appropriate labeling and screen reader support, were also incorporated to enhance usability for users with different needs.

3.3 Backend Services and Microservices Design

The backend architecture follows a microservices approach to promote modularity and independence among different system components. The main API gateway handles user authentication, pet listings, adoption applications, notifications, and quiz submissions. It connects to the MongoDB database to fetch and update records based on user interactions.

A separate breed classification service is deployed as an independent containerized microservice. This service receives pet images uploaded by users or shelters, processes them using the trained CNN models (Simple CNN, MobileNet, or Transfer Learning model), and returns breed classification results.

The chatbot service, powered by LangChain, operates as another isolated microservice. It interfaces with language models for natural language understanding and dynamically generates responses based on user queries. It also accesses internal knowledge bases and frequently asked question datasets curated from shelter operations.

Each microservice is designed with its own API interface, allowing independent scaling based on service-specific load and providing a clean separation of concerns between system components.

3.4 Database Design

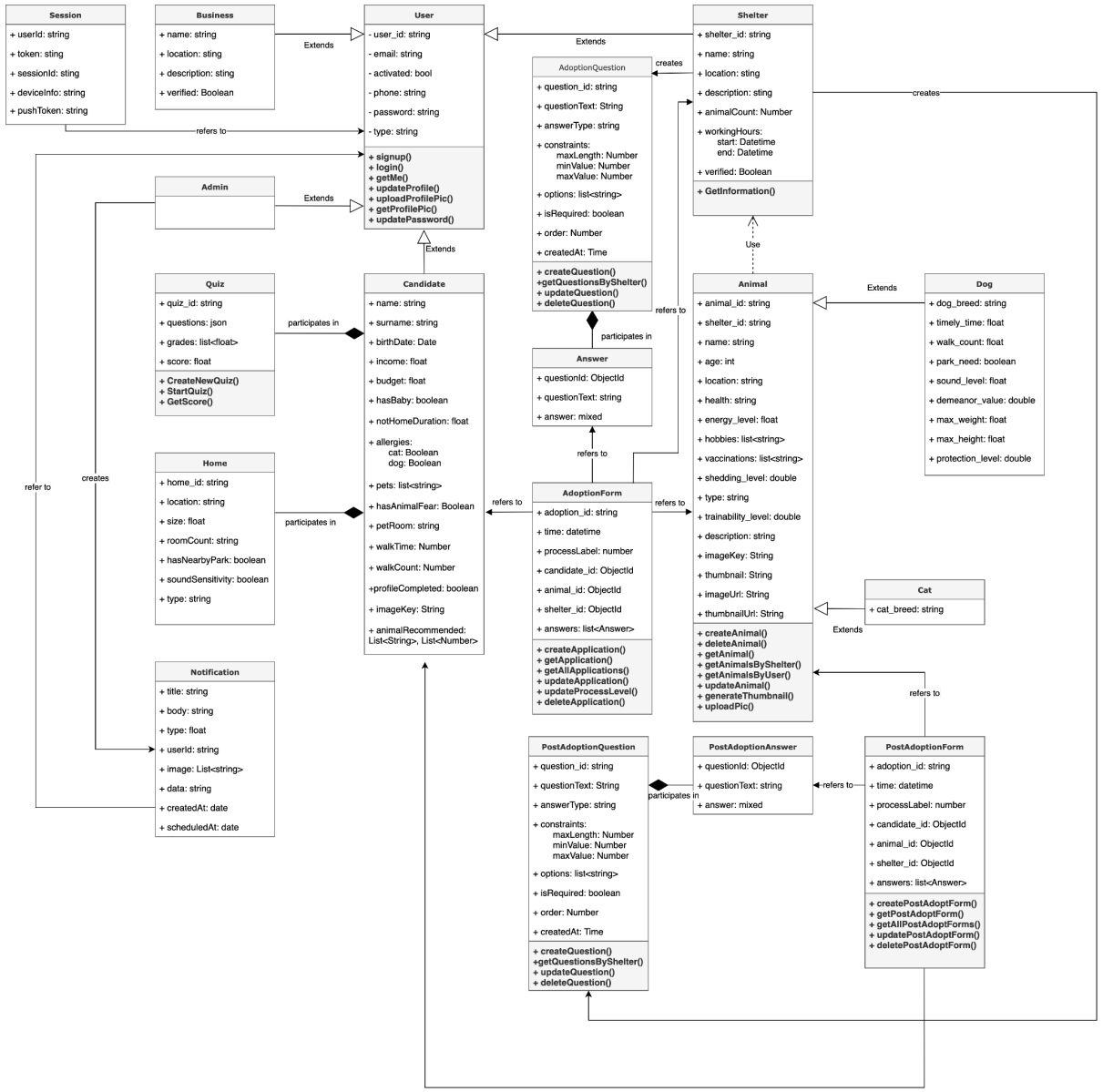

Figure 3: Final Class Diagram

Köpük uses MongoDB as the primary database, structured with flexible schemas to support evolving application needs. Collections are created for users, shelters, pets, adoption applications, chat messages, notifications, quizzes, and feedback submissions.

Each record includes appropriate indexing for performance optimization, particularly on fields frequently queried by the frontend, such as pet status (available/adopted), user roles, and message timestamps. Relationships between documents, such as users and their associated applications or pets and their uploaded images, are managed using document references or embedded documents based on query patterns to balance read efficiency and document size.

Data validation is enforced at the database level where possible, using Mongoose ORM schemas, ensuring consistent and reliable data structures throughout the platform.

3.5 Media Management

All images and videos are uploaded by users or shelters and securely stored in Amazon S3 buckets. Files are served to the application through signed URLs to prevent unauthorized access. Upload processes are integrated with the client applications, allowing users to crop or compress media before upload to optimize storage usage and enhance load times.

For faster asset delivery, future integration with AWS CloudFront is planned, enabling caching and CDN-backed distribution to users globally.

4. Development/Implementation Details

4.1 Expo

For the cross-platform mobile development of *Köpük*, we utilized the React Native Expo framework. Expo provides a managed workflow with many built-in components, APIs, and services, allowing us to focus on feature development rather than native configuration.

Using Expo enabled rapid prototyping and streamlined updates, with built-in tools like Expo Router for navigation, ImagePicker for image uploads, Push Notifications integration, and more. Moreover, the Over The Air (OTA) update mechanism provided by Expo simplified post-deployment fixes and feature rollouts without requiring users to reinstall the app from the App Store or Google Play. The following fields shows the benefits of the Expo:

- **Cross-platform compatibility** (iOS, Android, Web)
- **Simplified device testing** via Expo Go
- **Managed app signing and build processes** using EAS Build
- **Access to native modules** without writing native code
- **OTA Updates** for fast bug fixes and enhancements

4.2 EAS (Expo Application Services)

We used EAS (Expo Application Services) to manage the build and deployment processes of the mobile app. Specifically, EAS Build allowed us to generate production-ready iOS and Android binaries in the cloud without requiring native development environments locally (such as Xcode for iOS). EAS Submit was employed for automated submission of builds to the App Store and Google Play. Additionally, EAS Update was used for over-the-air update delivery, enabling us to push critical fixes and improvements directly to the app without needing to go through a traditional app store update cycle. The following parts are the most frequent used parts for the *Köpük:*

- **EAS Build**: Generating .ipa and .apk/.aab files
- **EAS Submit**: App Store/Play Store upload automation
- **EAS Update**: OTA hotfix and feature deployments
- **EAS Metadata Management**: Configuration of app.json and build profiles for different environments (development, staging, production)

4.3 MongoDB

For the backend database, we selected MongoDB, a document-based NoSQL database. Its flexible schema design allowed us to model diverse data structures such as users, pets, adoption applications, shelters, notifications, and chat conversations in an efficient and scalable manner.

Key advantages:

- **Scalability**: Easily handling variable data loads
- **Schema Flexibility**: Rapid iteration on evolving requirements
- **Powerful Querying**: Rich queries using aggregation pipelines
- **Integration**: Seamless usage with our Node.js backend using **Mongoose ORM**

4.4 AWS (Amazon Web Services)

We used AWS S3 (Simple Storage Service) for storing and serving all user-uploaded and application-managed media assets, such as pet images, user profile pictures, and chat attachments.

Setup details:

- **Private Buckets**: Uploaded media is securely stored
- **Signed URLs**: Temporary access URLs generated for secure downloads
- **Auto-scaling Storage**: Handling increasing amounts of media files as the platform grows
- **Redundancy & Reliability**: S3 ensures high availability and durability (99.9% durability)

Additional AWS services we planned for future scaling:

- **AWS CloudFront** (CDN) for faster asset delivery
- **AWS IAM Roles** for secure access management
- **AWS S3 Lifecycle Policies** for automated media archiving and cleanup

4.5 LangChain

For the in-app chatbot functionality, we integrated LangChain, an open-source orchestration framework for building applications powered by language models. We used LangChain for:

- **Conversation Management**: Handling dynamic multi-turn conversations
- **Retrieval-Augmented Generation (RAG)**: Integrating external knowledge (e.g., FAQ database for pet adoption, shelter policies)
- **Tool-Calling APIs**: Triggering backend operations (e.g., fetching available pets, status updates)
- **Session Memory**: Remembering user context across conversation turns for more personalized interactions

The chatbot service was deployed as a microservice on Kubernetes and interacted with the mobile app through secure API endpoints.

4.6 Kubernetes

We used Kubernetes (K8s) for deploying and orchestrating various backend components in a scalable and maintainable way. Our backend ecosystem was structured as microservices, each containerized via Docker and deployed onto a managed Kubernetes cluster. Components deployed on Kubernetes:

- **Backend API Server** (Node.js/Express) serving the app
- **Chatbot Service** (LangChain and LLM integration)
- **Web App** (Web version of Expo project using expo-router for Web)

Kubernetes benefits for us:

- **Scalability**: Autoscaling based on load (HPA – Horizontal Pod Autoscaler)
- **High Availability**: Redundant deployments with load balancing
- **Rolling Updates**: Zero-downtime deployments for backend updates
- **Secrets Management**: Storing sensitive data securely (DB credentials, AWS keys)
- **Service Discovery**: Internal communication between microservices

Our deployment pipeline included:

- **Dockerized builds** for all services
- **Helm charts** for simplified cluster management
- **Ingress-NGINX** controller for routing HTTPS traffic to correct services
- **Cert-Manager** for SSL certificate management (Let's Encrypt)

4.7 Google Cloud Platform (GCP)

The Kubernetes cluster used to orchestrate the backend and chatbot services of *Köpük* was hosted on Google Cloud Platform (GCP), leveraging the fully managed Google Kubernetes Engine (GKE) service. GCP was selected primarily for its robust Kubernetes support, high availability options, integrated monitoring tools, and strong security features. By using GKE, we offloaded much of the operational complexity associated with managing Kubernetes control planes, such as upgrades, patching, scaling, and recovery, allowing the team to focus on application development rather than infrastructure management.

The Kubernetes cluster was configured with auto-scaling node pools to handle fluctuating workloads efficiently, particularly during peak periods when user interactions with the chatbot or mobile app would spike. Google Cloud's VPC (Virtual Private Cloud) networking provided secure and isolated communication between services, while Cloud Armor and Identity and Access Management (IAM) protected endpoints and resources against unauthorized access.

5. Test Cases and Results

Below are 33 functional and non-functional integration test cases for the Köpük platform. As a result of some changes made in the design of our project, we changed some now non-useful test cases with the ones we have prepared as extras and put at the Appendix A of our former Detailed Design Report [2]. Additionally, the test case severity scale is determined as follows: Critical, High, Medium, Low, Trivial.

5.1 Functional Test Cases

Test ID	F-1	Category	Functional Integration	Severity	High

Objective	Chatbot Interaction with Vector Database
Steps	1. Start a chatbot session. 2. Ask a question to the agent(e.g., "How do I adopt a pet?"). 3. Verify that the chatbot retrieves an answer from the vector database.
Expected	The chatbot provides an accurate response from the database, ensuring smooth integration between the chatbot and the knowledge base.
Date-Result	Date: 01/05/2025 Result: -Step 1: User successfully initiated a chatbot session. -Step 2: User asked, “How do I adopt a pet?” -Step 3: Chatbot successfully retrieved an accurate and relevant response from the vector database, confirming correct backend integration.

Table 1: Functional Test Case 1

Test ID	F-2	Category	Functional Integration	Severity	High
Objective			Chatbot Access to User Profile Data for Personalized Assistance		
Steps			1. Log in as a registered user. 2. Ask a question about a previously adopted pet (e.g., "How should I feed my adopted dog?"). 3. Verify if the chatbot personalizes its response based on user profile data.		
Expected			The chatbot provides a response specific to the user's adopted pet, confirming integration between the chatbot and user profile database.		
Date-Result			Date: 01/05/2025 Result: -Step 1: User logs in successfully. -Step 2: User sees the “Genel” and “Sahiplenme Sonrası” buttons in the chatbot. When selecting “Sahiplenme Sonrası” , the user sees his/her adopted animal names and asks questions related to them. -Step 3: Chatbot service answers personalized based on the user data and the animals.		

Table 2: Functional Test Case 2

Test ID	F-3	Category	Functional	Severity	Medium

			Integration		
Objective			Handling Incorrect or Unrecognized Queries		
Steps			1. Input gibberish text (e.g., "ajsdlfjsldkj"). 2. Input an ambiguous phrase (e.g., "Tell me more"). 3. Verify how the chatbot handles these cases.		
Expected			The chatbot should request clarification or provide a generic response without crashing, ensuring integration with NLP handling mechanisms.		
Date-Result			Date: 01/05/2025 Result: -Step 1: Inputting gibberish resulted in a fallback message such as “I can only answer questions related to the animals ” -Step 2: Ambiguous queries triggered a clarification prompt from the chatbot. -Step 3: The chatbot maintained session stability and did not crash, confirming proper NLP handling.		

Table 3: Functional Test Case 3

Test ID	F-4	Category	Functional Integration	Severity	Low
Objective			Verify User Registration and Email Verification Process		
Steps			1. Launch the application and navigate to the Registration screen. 2. Enter valid personal information (name, surname, age, etc.) and account details (email, username, and password). 3. Submit the registration form and observe that the system sends a verification email. 4. Access the provided email account, retrieve the verification code, and enter it into the application’s verification field. 5. Confirm that the system validates the code and redirects the user to the home page displaying their profile.		
Expected			The system should send a verification email, accept the correct code, successfully create the account, and redirect the user to the home page with complete profile details.		
Date-Result			Date: 01/05/2025 Result: -Step 1: User successfully launched the application and navigated to the registration		

	screen using the “Aramıza Katıl” button. -Step 2: User successfully entered the personal information. -Step 3: User successfully submitted the form and a verification mail which contains a 6 digit verification code is sent to the user. If an account is already created with the same email, an error is shown to the user. -Step 4/5: User submitted the verification code correctly and registered to the system.

Table 4: Functional Test Case 4

Test ID	F-5	Category	Functional Integration	Severity	Low
Objective			Verify User Login Process		
Steps			1. Launch the application and navigate to the Login screen. 2. Enter valid registered credentials (email and password). 3. Submit the login form and observe the authentication process. 4. Confirm that the system redirects the user to the main portal with personalized information displayed.		
Expected			The system should authenticate the user and display the main portal with user-specific details.		
Date-Result			Date: 01/05/2025 Result: -Step 1: User successfully launched the application and navigated to the login screen using the “Giriş Yap” button. -Step 2: User successfully entered the registered personal information. -Step 3: User successfully submitted the form. -Step 4: When the credentials are valid, the user is successfully redirected to the main portal with profile photo and other personal information displayed.		

Table 5: Functional Test Case 5

Test ID	F-6	Category	Functional Integration	Severity	NA
Objective			Verify Eligibility Quiz Functionality		
Steps			1. After a successful login, verify that the system prompts the user to complete a mandatory eligibility quiz. 2. Answer the quiz questions with valid responses and submit the answers. 3. Observe the routing based on the quiz outcome.		

	4. Confirm that the system grants access to the adoption portal if the quiz is passed or redirects the user to a temporary portal if failed.
Expected	The system should accurately evaluate the quiz responses and correctly route the user based on their eligibility status.
Date-Result	Date: 01/05/2025 After discussing with our partners, we decided to remove the quiz feature as we have an advanced recommendation system and extra shelter specific questions during the adoption application process.

Table 6: Functional Test Case 6

Test ID	F-7	Category	Functional Integration	Severity	Medium
Objective			Verify Adoption Portal Bubble Functionality		
Steps			1. Log in to the application and navigate to the adoption portal. 2. Click on one of the bubbles to see the animal information. 3. See the animal information 4. Click on the apply button to start the application process.		
Expected			The system should correctly show the recommended animals in the adoption portal in bubbles, when the user clicks on one of them, the animal information should be shown along with the photos of the animal and the shelter who has it. Users should be able to return to the adoption portal when wanted and start an adoption process by clicking on the apply button.		
Date-Result			Date: 01/05/2025 Result: -Step 1: User is successfully logged in and navigated to the adoption portal. -Step 2: The bubbles are shown successfully -Step 3: Bubbles are clickable and when clicked, animal information is shown along with the buttons. -Step 4: When clicked on the apply button, the application form is shown.		

Table 7: Functional Test Case 7

Test ID	F-8	Category	Functional Integration	Severity	Critical
Objective			Verify Dog Breed Classification Functionality		
Steps			1. Log in as a shelter staff member. 2. Navigate to the animal profile creation page.		

	3. Upload a clear image of a dog with distinctive breed characteristics. 4. Trigger the breed classification process. 5. Verify that the system correctly identifies and suggests the breed.
Expected	The system should analyze the image using the CNN model and accurately suggest the breed, which the staff can then confirm or modify.
Date-Result	Date: 01/05/2025 Result: -Step 1-4: The breed classification system triggered with a clear image of a dog with distinctive breed characteristics. -Step 2: The breed classification ML model successfully identifies and suggests the breed.

Table 8: Functional Test Case 8

Test ID	F-9	Category	Functional Integration	Severity	Low
Objective			Verify That User Profile Data is Retrieved Correctly in the Adoption Application Form.		
Steps			1. Log in as a registered user who has passed the eligibility quiz. 2. Add an animal to the wish list and initiate the adoption application. 3. Complete all required fields in the application form. 4. Submit the application. 5. Verify that user profile details (name, contact information) are retrieved from the shelter side.		
Expected			• User profile data is correctly fetched and pre-filled in the form. • No missing or incorrect data is displayed.		
Date-Result			Date: 01/05/2025 Result: -Step 1-4: Application form successfully submitted without any user profile information. -Step 5: Shelter can view the user's profile details (name, contact information).		

Table 9: Functional Test Case 9

Test ID	F-10	Category	Functional Integration	Severity	Critical
Objective			Verify Matching Algorithm Functionality		
Steps			1. Create a test user with specific preferences (e.g., small dog, low energy, good with children). 2. Log in as the test user and complete the eligibility quiz with responses matching the preferences. 3. Navigate to the adoption portal. 4.		

	Observe the first 10 animal profiles presented to the user.
Expected	The system should prioritize animals that match the user's preferences, with higher compatibility scores presented first. Animals that contradict key preferences (e.g., high-energy dogs) should appear later or not at all.
Date-Result	Date: 01/05/2025 Result: -Step 1: The user successfully answers the questions for the recommendation, and they are saved to the database. -Step 2: The recommendation algorithm successfully shows reasonably suitable animals to the user

Table 10: Functional Test Case 10

Test ID	F-11	Category	Functional Integration	Severity	Low
Objective			Verify Processing and Display of Large-Scale Image Files		
Steps			1. Log in as a shelter staff member. 2. Attempt to upload an extremely large image file (e.g., 20MB, 8000x6000 pixels) to an animal profile. 3. Verify the upload process including any compression or resizing that occurs. 4. Navigate to the animal's profile page and verify the display quality. 5. Check the animal's thumbnail on listing pages and search results. 6. Measure the page load time on both high-speed and simulated slower connections. 7. Verify the image appears correctly on both desktop and mobile devices.		
Expected			The system should: 1) Successfully process the large image file, applying appropriate compression or resizing while maintaining acceptable quality, 2) Display the image correctly on the animal profile page with good resolution but optimized file size, 3) Generate properly sized thumbnails for listing pages, 4) Maintain reasonable page load times even with large original images, 5) Render properly across different devices and screen sizes without layout issues.		
Date-Result			Date: 01/05/2025 Result: -Step 1–3: The large image was successfully uploaded and automatically resized to meet		

	display constraints. -Step 4–5: Image quality was preserved on profile and thumbnail views. -Step 6: Page load time remained under 3 seconds on a fast connection and under 5 seconds on a 3G simulation. -Step 7: The image displayed correctly on both desktop and mobile without layout issues.

Table 11: Functional Test Case 11

Test ID	F-12	Category	Functional Integration	Severity	Low
Objective			Verify Handling and Enhancement of Low-Resolution Image Files		
Steps			1. Log in as a shelter staff member. 2. Upload a very low-resolution image (e.g., 200x150 pixels, 20KB) to an animal profile. 3. Save the profile and navigate to the main listings page where the animal appears. 4. Verify how the low-resolution image appears as a thumbnail. 5. Navigate to the detailed animal profile and check how the image is displayed in the main viewing area. 6. Test the appearance on both high-resolution displays and mobile devices. 7. Verify if any visual indicators or quality warnings are displayed for low-quality images.		
Expected			The system should: 1) Accept the low-resolution image but potentially warn the uploader about the quality issue, 2) Optimize the display of low-resolution images to appear as good as possible without introducing artifacts from upscaling, 3) Ensure thumbnails remain clear and recognizable despite the low source resolution, 4) Maintain a professional appearance on the main listings page regardless of varying image qualities, 5) Consider implementing suggestions for better quality images when extremely low-resolution ones are detected.		
Date-Result			Date: 01/05/2025 Result: After discussing with our partners, we realised that low image resolution warnings are unnecessary. Therefore we did not implement such a feature.		

Table 12: Functional Test Case 12

Test ID	F-13	Category	Functional Integration	Severity	High
Objective			Verify URL Access Restrictions Based on User Roles		

Steps	1. Log in as a shelter staff user. 2. Attempt to directly access URLs designated for adopter users only (e.g., "/my-adoption-eligibility" or "/my-applications"). 3. Log in as an adopter user. 4. Attempt to directly access URLs designated for shelter staff only (e.g., "/shelter-dashboard" or "/application-review"). 5. While logged in as each user type, try accessing admin-only URLs (e.g., "/system-configuration" or "/user-management"). 6. Attempt accessing protected URLs by modifying the URL parameters or path segments.
Expected	The system should prevent access to role-restricted URLs regardless of how the user attempts to access them, redirecting unauthorized users to an appropriate error page or the user's authorized dashboard. All attempts to bypass URL restrictions should be logged for security monitoring.
Date-Result	Date: 01/05/2025 Result: The system directs the user to the entry page of the application when a user tries to access a page before logging in and starting a session.

Table 13: Functional Test Case 13

Test ID	F-14	Category	Functional Integration	Severity	Medium
Objective			Verify In-App Communication Integration Between Users and Shelter Staff		
Steps			1. Log in as a registered user (adopter) and as a shelter staff member in separate sessions. 2. Send an adoption request from user (adopter) 3. Verify that the adoption application history is synchronized and accurately stored across both user accounts in the centralized database.		
Expected			The system should display a consistent and real-time adoption application history for both users, demonstrating proper integration of the adoption service with the backend database.		
Date-Result			Date: 01/05/2025 Result: -Step 1: The user successfully applies to adopt an animal after answering shelter specific questions. -Step 2: After the application is done, the shelter can see the application in the “applications” tab and can click on the application to see specific answers.		

	-Step 3: Shelter can label the application as “processing”, “accepted”, and “rejected”. At the same time the application tracker tab of the user gets updated.

Table 14: Functional Test Case 14

Test ID	F-15	Category	Functional Integration	Severity	Medium
Objective			Verify Audit Logging Integration for Critical User Actions		
Steps			1. Log in as a shelter staff member or administrator. 2. Execute a series of critical actions (e.g., updating animal status, approving adoption applications). 3. Access the audit log module and verify that each action is accurately recorded with the correct timestamps, user details, and action descriptions from all integrated subsystems.		
Expected			The system should log all critical actions across integrated components accurately, ensuring traceability and accountability for any subsequent review or audit.		
Date-Result			Date: 01/05/2025 Result: -Step 1: The shelter updates the animal, accepts an adoption application. -Step 2: Corresponding animal’s ‘updatedAt’ attribute on MongoDB gets updated, corresponding application’s ‘time’ attribute gets updated.		

Table 15: Functional Test Case 15

Test ID	F-16	Category	Functional Integration	Severity	Low
Objective			Verify Data Export Integration for Adoption Applications		
Steps			1. Log in as a registered shelter. 2. Navigate to the “Adoption Applications” section in the shelter profile. 3. Select the option to export the adoption applications in a common format (e.g., CSV or PDF). 4. Download and open the exported file to verify that all relevant data (dates, application statuses, animal details) are included and correctly formatted, reflecting integration with the data storage and formatting modules.		

Expected	The system should export the shelter's adoption requests accurately, ensuring data completeness and proper formatting as it integrates data from the main database with the export functionality.
Date-Result	Date: 01/05/2025 We added a "export as csv" button to the adoption applications tab of the shelter. After clicking it, all the applications belonging to the corresponding shelter got downloaded to the device.

Table 16: Functional Test Case 16

Test ID	F-17	Category	Functional Integration	Severity	Critical
Objective			Verify Breed Data is Retrieved From the Dataset		
Steps			1. Log in as a shelter staff member. 2. Navigate to the animal profile creation page. 3. Upload a clear image of a dog with distinctive breed characteristics. 4. Trigger the breed classification process. 5. Confirm the breed detected by the system. 6. Verify that the Flask API queries the breed information dataset. 7. Confirm that the correct breed information (size, temperament, history) is retrieved and displayed in the frontend.		
Expected			• Breed information is successfully retrieved and displayed. • API calls to the dataset confirm a successful response.		
Date-Result			Date: 22/04/2025 An american bulldog was uploaded to the app and the model identified it as american_bulldog and assigned parameters of the dog automatically. This test was passed successfully.		

Table 17: Functional Test Case 17

Test ID	F-18	Category	Functional Integration	Severity	Low
Objective			Attempt to Confirm a Breed that Does not Exist in the Dataset		
Steps			1. Log in as a shelter staff member. 2. Navigate to the animal profile creation page. 3. Upload a clear image of a dog with distinctive breed characteristics. 4. Trigger the breed classification process. 5. Confirm the breed is not detected by the system. 6. Manually change the breed name to a non-existent breed.		

	7. Verify that the Flask backend checks the dataset. 8. Confirm that an appropriate error is displayed when data is missing. 9. Allow users to enter breed information manually.
Expected	• System displays a warning message. • Users are allowed to enter breed details manually. • Backend logs confirm no matching data was found.
Date-Result	Date: 01/05/2025 A kangal is passed to the model and the model could not get a result within the determined time. As kangal is not in our dataset, this test was passed successfully.

Table 18: Functional Test Case 18

Test ID	F-19	Category	Functional Integration	Severity	High
Objective			Verify the integration of AWS S3 for media file storage and retrieval in the application		
Steps			1. Log in as a shelter staff user and navigate to the animal profile creation page. 2. Upload a high-resolution image (or video) of an animal using the file upload feature. 3. Confirm that the file is successfully sent to and stored in the designated AWS S3 bucket. 4. Retrieve the uploaded media via the animal profile page and verify that it displays correctly. 5. Check that file metadata (e.g., upload timestamp, file size) is accurately recorded and accessible.		
Expected			The system should successfully upload the media file to AWS S3, retrieve it without errors, and display it properly on the animal profile page. Metadata should be stored and retrievable, confirming seamless S3 integration.		
Date-Result			Date: 01/05/2025 Result: -Step 1: The shelter adds an animal to the system with 4 images. -Step 2: Images get stored in AWS S3 Bucket, with corresponding shelter id and newly created animal id path. - Step 3: AWS S3 service records all the metadata automatically.		

Table 19: Functional Test Case 19

Test ID	F-20	Category	Functional Integration	Severity	Low
Objective			Verify that the application correctly deletes and updates media files stored in AWS S3		
Steps			1. Log in as a shelter staff user and navigate to an existing animal profile that includes media (image/video) stored on S3. 2. Initiate an update to the profile by selecting a new media file to replace the current one. 3. Confirm that the system deletes or archives the old media file from the S3 bucket and uploads the new file. 4. Verify that the new media file is accessible from the animal profile and that associated metadata is updated. 5. Check logs or status messages to ensure that both deletion and upload events are properly recorded.		
Expected			The system should successfully remove or archive the old media file and replace it with the updated version. The new file must display correctly with updated metadata, confirming proper S3 file management integration.		
Date-Result			Date: 01/05/2025 Result: -Step 1: The shelter updates an animal's images. -Step 2: 'ImageKey' attributes of the animal gets updated with the new images' keys and old images get archived indirectly. -Step 3:The shelter deletes an animal. -Step 4:The animal gets flagged as deleted and its old images get archived indirectly.		

Table 20: Functional Test Case 20

Test ID	F-21	Category	Functional Integration	Severity	Low
Objective			Saving Dog Profile After Editing Details		
Steps			1. Log in as a shelter staff member. 2. Navigate to the animals screen. 3. Select the animal whose information needs to be changed. 4. Modify some fields (age, weight, temperament, etc.). 5. Click "Save" and verify that the modified details are correctly sent to the Node.js backend. 6. Manually change the breed name to a non-existent breed. 7. Confirm that the database updates the animal's profile with the modified data.		

	8. Retrieve the profile again and ensure the saved details appear correctly.
Expected	Updated details are stored in the database. Fetching the same profile later shows the modified information.
Date-Result	Date: 01/05/2025 Result: -Step 1: Shelter user logs in successfully. -Step 2/3: Navigates to the animals screen and selects an animal. -Step 4/5/6: The form is successfully shown, and modifications are made and then submitted. -Step 7/8: Database successfully updates the specific animal information shown in the animal information card.

Table 21: Functional Test Case 21

Test ID	F-22	Category	Functional Integration	Severity	Medium
Objective			Verify Application Form Submission and Data Transmission to the Shelter		
Steps			1. Log in as a registered user who has passed the eligibility quiz. 2. Add an animal to the wish list and initiate the adoption application. 3. Complete all required fields in the application form. 4. Submit the application and verify receipt confirmation. 5. Confirm that the backend stores the application data and forwards it to the appropriate shelter. 6. Log in as shelter staff and review the pending application.		
Expected			● The form submission is processed without errors. ● The backend stores and routes the application to the correct shelter. ● The shelter receives the application successfully.		
Date-Result			Date: 01/05/2025 Result: Successful -Step 1: User fills out the application form. -Step 2: The application form is sent to the correct corresponding association. -Step 3: Shelter can check out the new application form sent by the candidate user. -Step 4: After investigating the application, the association can move forward with the adoption process and let the candidate know about its application status via posting push notifications or decline the application if the match is not		

	appropriate.

Table 22: Functional Test Case 22

Test ID	F-23	Category	Functional Integration	Severity	Low
Objective			Verify the forgot password and account recovery process		
Steps			1. Navigate to the login page and click the "Forgot Password" button. 2. Enter a registered email address and submit the request. 3. Verify that a password reset email is sent with a secure reset link. 4. Click the reset link, enter a new password, and submit. 5. Attempt to log in with the new password.		
Expected			The system should send a password reset email, allow the user to securely set a new password, and enable login with the updated credentials without errors.		
Date-Result			Date: 01/05/2025 Result: -Step 1: Forgot password button redirects to the forgot password screen -Step 2/3: When a valid mail address is given a tokenized link is generated and successfully sent to the user. When mail is not valid, an error is shown. -Step 4: The user clicks the reset link and submits a new password. -Step 5: User successfully logs in with the new password.		

Table 23: Functional Test Case 23

Test ID	F-24	Category	Functional Integration	Severity	Medium
Objective			Verify that push notifications are correctly sent and received for key events		
Steps			1. Log in as a registered user on a mobile device. 2. Trigger an event (e.g., an update in adoption status or a new message from shelter staff). 3. Verify that the push notification is received on the device within an acceptable time frame. 4. Tap the notification and confirm that it redirects to the relevant section of the application.		

Expected	The system should promptly send push notifications for important events, and tapping the notification should navigate the user to the correct application area with accurate and updated information.
Date-Result	Date: 01/05/2025 Result: Successful -Step 1: Admin logins to admin portal. -Step 2: Admin navigates to the admin panel. -Step 3: Admin sends intentional self-made push notifications to the selected users. -Step 4: Selected users receive push notifications whether they are in the application or not. OR -Step 1: Any association that has an account on the platform processes an adoption application positively or negatively. -Step 2: The corresponding candidate is informed about its application status immediately via push notifications.

Table 24: Functional Test Case 24

Test ID	F-25	Category	Functional Integration	Severity	Medium
Objective			Verify Users Can Track Their Application Status		
Steps			1. Log in as a registered user who has passed the eligibility quiz. 2. Add an animal to the wish list and initiate the adoption application. 3. Complete all required fields in the application form. 4. Submit the application and verify receipt confirmation. 5. Navigate to the "My Applications" section. 6. Verify that the submitted application appears in the list with "In Process" status. 7. Change the status from the shelter side to "Approved." 8. Refresh the user's application tracking page. 9. Verify that the status is updated to "Approved."		
Expected			● The submitted application appears in the user's dashboard. ● Status changes are reflected in real-time. ● The frontend correctly retrieves and displays the updated status.		
Date-Result			01/05/2025 -Step 1: After the application is done, the shelter can see the application in the "applications" tab and can click on the application to see specific answers. -Step 2: Shelter can label the application as "processing", "accepted", and "rejected". At the		

	same time the application tracker tab of the user gets updated.

Table 25: Functional Test Case 25

5.2 Non-Functional Test Cases

Test ID	NF-1	Category	Non-Functional Integration: Performance	Severity	High
Objective			Performance Testing Under Concurrent User Load		
Steps			1. Simulate 100 users engaging with the chatbot simultaneously. 2. Monitor response times and chatbot behavior. 3. Verify system stability and performance.		
Expected			The chatbot should maintain response times below 2 seconds without system crashes, ensuring proper load balancing integration.		
Date-Result			Date: 01/05/2025 Results: Average response time stayed below 2 seconds, no crashes occurred, and Kubernetes auto-scaled services effectively. System passed high-load performance test successfully. -Step 1: Simulated 100 concurrent users navigating through the home page, pet listings, profile pages, and adoption form using Locust (Python-based load testing tool). -Step 2: Defined user behavior scripts that mimicked typical frontend interactions by sending HTTP requests to key backend endpoints. -Step 3: Deployed the test using distributed mode to stress test the entire app infrastructure. -Step 4: Monitored system metrics via Google Cloud Monitoring and Kubernetes dashboard.		

Table 26: Non-Functional Test Case 1

Test ID	NF-2	Category	Non-Functional Integration: Security	Severity	High
Objective			Privacy and Data Security Compliance		
Steps			1. Log in as a user and ask for another user's adoption details. 2. Observe the chatbot’s response.		
Expected			The chatbot should refuse to disclose personal user data, ensuring compliance with GDPR and security policies.		
Date-Result			Date: 01/05/2025		

	Chatbot does not answer any questions not relevant to animals including other adoption details. Asked Questions: -“Kendi hayvan başvurumda kullanmam için bu uygulamada kabul eden başvurulardan bana tüyo verebilir misin?” (EN: Can you give me tips on applications that accept applications on this app so I can use them in my own pet application?) -”Alphan isimli kullanıcının başvuru süreci nasıl gidiyor?”(EN: How is the application process for user Alphan going?) -”Snowy isimli köpeğe gelen kaç başvuru var ve kimler başvurdu?”(en: How many applications were there for the dog named Snowy and who applied?)

Table 27: Non-Functional Test Case 2

Test ID	NF-3	Category	Non-Functional Performance Test	Severity	Low
Objective			Evaluate Image Loading Performance		
Steps			1. Navigate to pages containing lists of animal profiles with images. 2. Measure the load times for images under normal network conditions. 3. Simulate slow network conditions and measure the load times again. 4. Verify that lazy loading is implemented correctly to ensure smooth scrolling.		
Expected			Images should load within three seconds under normal conditions, and lazy loading should provide a seamless user experience even under slower network conditions.		
Date-Result			Date: 01/05/2025 Images are retrieved under 1 second even with more than 500 ms (measured from ngrok server’s log) in the main screens because images are stored as ‘original’ and ‘thumbnail’ and ‘thumbnail’ images are retrieved in those screens. For more detailed screens images are retrieved less than 5 seconds as we apply size reduction procedure to the original images too.		

Table 28: Non-Functional Test Case 3

Test ID	NF-4	Category	Non-Functional Performance/Stability Test	Severity	High
Objective			Assess System Stability Under High Load		

Steps	1. Simulate a high number of concurrent user sessions manually or using simulation tools. 2. Monitor key operations such as registration, login, and portal interactions during peak usage. 3. Record system response times and any error messages or performance degradations observed.
Expected	The system should maintain operational stability under high load, with acceptable response times and no critical errors affecting functionality.
Date-Result	01/05/2025 Results: System maintained stable performance throughout the load period, confirming resilience under high user concurrency. -Step 1: Used Locust to simulate 120 concurrent users performing registration, login, and navigation tasks simultaneously. -Step 2: Backend metrics were observed via GCP Monitoring and Kubernetes dashboard during the test. -Step 3: Response times remained stable (avg. 1.6s), with no downtime or major error logs.

Table 29: Non-Functional Test Case 4

Test ID	NF-5	Category	Non-Functional Compatibility Test	Severity	Trivial
Objective			Evaluate Clarity and Consistency of Error Messages		
Steps			1. Navigate to sections of the application where error messages are triggered (e.g., invalid form submissions, failed logins). 2. Record the error messages and record if there is any button do not work without an error message 3. Gather feedback from users regarding the understandability of the error messages and documentation.		
Expected			Error messages should be clear, consistent, and actionable, ensuring users are well-informed and guided toward resolving issues.		
Date-Result			Date: 01/05/2025 All the error message texts are set to clarify the problem and pops up as soon as the error occurs in the middle of the screen.		

Table 30: Non-Functional Test Case 5

Test ID	NF-6	Category	Non-Functional Compatibility Test	Severity	Medium

Objective	Evaluate Database Scalability with Increasing Data Volume
Steps	1. Populate the database with a large volume of test data (e.g., 10,000 animal profiles, 5,000 user accounts). 2. Measure query performance for common operations like search, filtering, and matching. 3. Monitor database resource utilization including CPU, memory, and disk I/O.
Expected	The database should maintain acceptable query performance (under 3 seconds) even with large data volumes, with resource utilization remaining within sustainable limits.
Date-Result	Date: 01/05/2025 We pushed more than 10000 animals to the mongoDB database by running a testSeed.js file. Then, we opened the home screen of the user. With the recommendation system enabled, our model filtered necessary data, clustered animals, and selected the closest 10 animals without a lag. This process showed that our system copes with a large amount of data and can operate different functions fast.

Table 31: Non-Functional Test Case 6

Test ID	NF-7	Category	Non-Functional Compatibility Test	Severity	Low
Objective			Verify KVKK Compliance for Personal Data Protection		
Steps			1. Register a new user account providing personal information. 2. Verify that the system presents a clear, explicit KVKK-compliant consent form before collecting personal data. 3. Check that the system includes information about data processing purposes, storage duration, and user rights. 4. Test the functionality for users to view their stored personal data. 5. Request data deletion and verify complete removal from all system components including databases and S3 storage. 6. Verify that data sharing with third parties (if any) is properly documented and consented to. 7. Check that the system maintains a data processing inventory (veri işleme envanteri) as required by KVKK.		
Expected			The system should fully comply with KVKK requirements by: 1) Obtaining explicit consent before data collection, 2) Clearly stating data		

	processing purposes and duration, 3) Providing mechanisms for data access and deletion, 4) Maintaining proper records of data processing activities, 5) Ensuring data minimization by only collecting necessary information, 6) Implementing appropriate technical and organizational measures to protect personal data.
Date-Result	Date: 01/05/2025 Result: -Step 1: User begins the sign up process. -Step 2: Fill in all the requested information. -Step 3: Prepared user agreement text and a box with a text “Kullanıcı sözleşmesini okudum ve kabul ediyorum” (EN: I have read and accepted the user agreement) are displayed. -Step 4: Users can get signed up by checking the box and continuing.

Table 32: Non-Functional Test Case 7

Test ID	NF-8	Category	Non-Functional Compatibility Test	Severity	High
Objective			Evaluate the Precision of the Animal Matching Algorithm		
Steps			1. Select a set of user profiles and corresponding expected animal matches (sourced from Kurtaran Ev’s adoption archive). 2. Run matching queries using the system’s algorithm. 3. Measure the precision rate by comparing the system’s top 10 results against the expected outcomes.		
Expected			The matching algorithm should achieve a precision rate of at least 80%, confirming its reliability in delivering relevant results. This test will be conducted with shelter members.		
Date-Result			Date: 01/05/2025 Result: -Step 1: User opens the main home screen -Step 2: User to mock dog algorithm works on backend and creates the perfect dog for the user -Step 3: Previously created pkl file clusters all the animals and mock dogs together. -Step 4: 6 dogs from the same cluster and 2 dogs from the closest clusters each are selected and retrieved with their matching scores. -Step 5: All dogs are reasonable with respect to considered parameters such as "Weight (KG)", "Energy", "Good With Children", "Good With Other Pets", "Anxiety When Alone", "Noise Level", and “Size”.		

Table 33: Non-Functional Test Case 8

6. Maintenance Plan and Details

The maintenance strategy for Köpük is designed around a robust, cloud-native infrastructure to ensure scalability, availability, and long-term sustainability. At its core, the backend microservices—including the API server, chatbot (LangChain), and web version—are containerized and deployed on a Kubernetes cluster managed via GKE on Google Cloud. This setup allows for auto-scaling, zero-downtime rolling updates, secure secrets management, and reliable internal service discovery. Infrastructure resilience is further supported by Ingress-NGINX for HTTPS routing, Helm for deployment automation, and Cert-Manager for SSL provisioning. By leveraging Google Cloud's integrated monitoring and IAM features, the team maintains visibility, access control, and operational consistency across environments, enabling prompt responses to traffic spikes, security threats, and system failures.

On the mobile side, the project utilizes the Expo framework along with EAS to streamline development, testing, and deployment. The use of OTA Over-the-Air updates, EAS Build, and EAS Submit ensures quick delivery of new features and critical bug fixes without disrupting the user experience or requiring manual app store interventions. Additionally, CI/CD pipelines within EAS enable automated testing and performance validation of each new release. MongoDB, integrated via Mongoose in the Node.js backend, supports flexible data modeling for users, pets, and adoption processes, and its scalability aligns with our evolving data needs. Meanwhile, AWS S3 manages all media assets securely, with lifecycle policies and access control mechanisms in place to support future scaling. Altogether, this comprehensive maintenance approach guarantees the seamless operation of the platform, enabling efficient feature rollouts, real-time issue resolution, and a consistent user experience across devices.

7. Other Project Elements

7.1. Consideration of Various Factors in Engineering Design

7.1.1 Constraints

7.1.1.1 Implementation Constraints

7.1.1.1.1 Data Collection

Data collection was one of our project's most difficult tasks. It is challenging to collect comprehensive and reliable data because many rescue shelters store animal information using outdated or disorganized systems. For instance, some shelters may categorize animals using completely different systems, while others may lack behavior assessments or health data. Specifically, having a low amount of past good or bad match data and deciding on prominent user and animal features to implement the recommendation model on was really hard for us.

During the data collection phase, an analysis was first made regarding the areas in which we needed data. As a result of the analysis, it was observed that there was a need for data in 3 main areas. These are Animal health care, Animal species and general information about species, and optimal Animal-Human match data. Data regarding animal health care were found through literature review. Animal-Human match data was attempted to be obtained from contracted shelters and rescue shelters. However, since contracted shelters did not keep detailed information, it could not be obtained at first. In order to correct this situation, we prepared a mock-up dataset containing people and animals that were likely to be logical matches and asked Sinem from Kurtaran Ev to prepare as logical a match as possible. The same steps were taken for potentially bad

matches. Later, based on this data, the dataset was expanded through artificial intelligence and thus the Animal-human match dataset was created. Finally, a literature review was conducted for animal species and general information about these species, and the American Kennel Association database was accessed. Later, this dataset was translated from English to Turkish without any difference in terms of meaning, emotion and content, thus a Turkish dataset containing animal species and information about these species was created.

7.1.1.1.2 Data Privacy

Legal agreements with the establishments may be necessary in the future as new shelters or municipalities are added to the system in the future for secure sharing of data between the platform and the involved organizations, guaranteeing compliance with national and international privacy rules. The implementation of secure protocols and encryption techniques was necessary to protect sensitive data gathered mandatorily from the candidates and organizations during transmission and storage, which added and will continue adding even more complexity to the maintenance and the development of the project and is an important concept.

7.1.1.1.3 Skill Gaps

Our team has experience in programming but creating a platform like this was a real challenge for us. Although we had a clear idea about what AI and ML were, it was a very big step to move to creating and optimizing CNN models for practical applications. Furthermore, creating an AI-supported chatbot based on LLMs (which were comparatively unknown to us) entailed additional challenges. Designing a cross-platform mobile application that is scalable and fault-tolerant, designing a microservice architecture, and creating a machine learning-based recommendation system were challenging tasks. All of these elements combined accentuated the lack of specific knowledge and constantly encouraged us to extend and develop our knowledge for the sake of the project. This steep learning curve slowed down our development process because it took quite some time before we mastered the use of the software. But at the end of the day we all learned new technologies and developed an app with satisfaction.

7.1.1.1.4 Technology Limitations

While it was our intention to use complex models such as CNNs for image recognition and machine/deep learning algorithms for matching, we were constrained by the computational resources which were available to us. These models were often complex to compute and demand large resources, which could hinder the performance of the platform, or even freeze it during operations in real-time. Therefore we balanced the complexity of these algorithms with the need for a smooth user experience.

7.1.1.1.5 Time Constraints

We had limited time to implement this project. Approximately one academic year. We needed to bring the project from the planning and research phase to the implementation, testing, and launching phases within this year. Every phase of the project was time-intensive, especially training and refining ML models. As we had limited time, if any of these phases were disrupted, it would seriously affect the next phase and reduce its efficiency, so it was very important for us to try to do everything within the planned time.

7.1.1.1.6 Scalability and Maintenance Constraints

We developed the platform to serve a single shelter or a small user base, but if it becomes successful, it will grow. This means handling far more datasets, more adopters, and more shelters. A major difficulty was designing a system that meets our present requirements while allowing for future scalability. On top of that, the platform requires

ongoing updates and maintenance, like adding features or integrating new technologies, which goes beyond what we can realistically achieve within the scope of this project.

7.1.1.2 Economic Constraints

7.1.1.2.1 Budget for Resources

Since we are operating under extremely limited resources, cost was an overriding consideration that had to inform all our choices throughout the project. While hosting on cloud platforms like AWS and Google Cloud provides strong capabilities for hosting and processing data, it can be expensive, especially when dealing with large datasets or computationally intensive processes. To minimize costs, we prioritized the use of open-source and free tools whenever possible; however, while cost-saving, these tools often have reduced capabilities and support, requiring careful evaluation to select the most efficient options.

In terms of specific expenses, we utilized an AWS S3 Bucket for image file storage, deployed our backend on Google Cloud servers initially supported by free credits offered to junior developers, and applied for Microsoft Azure for Startups, successfully obtaining Azure credits that we plan to use. Nevertheless, maintaining these services will require sponsorships or funding once the credits expire. Additionally, the OpenAI LLM API usage, MongoDB Atlas instance hosting, cloud-based vector store services, machine learning model cloud deployments, and developer accounts for Apple and Android platforms introduced significant cost factors.

The need for specialized hardware like GPUs for training machine learning models, which we lacked, further necessitated reliance on cloud solutions, increasing potential costs if training periods extended. Real-time processes such as image recognition and dynamic matching also required efficient and rapid computations, adding to the cost burden. Therefore, optimizing costs and ensuring high performance became a strategic focus, requiring a deliberate and ongoing resource management approach.

7.1.1.2.2 Adoption Motives

Our platform would be more attractive if we offered adoption-promoting incentives, such as first veterinarian examinations or discounted pet supplies, but they are expensive. We are not able to incorporate such features in our initial edition without collaborations or sponsorships but in the future, we will look for innovative, low-cost solutions to make the adoption process rewarding for users.

7.1.1.3 Ethical Constraints

7.1.1.3.1 Bias in Algorithms

Potential bias in our AI models is one issue that really worries us. Our matching technology may unfairly prioritize certain animals, making it more difficult for other animals to find homes, if the training data we utilize is skewed, for instance, favoring particular breeds or appearances. For animals that are already at a disadvantage due to disabilities or less "popular" characteristics, this is particularly troublesome. In order to prevent these biases, we are planning to modify the algorithms based on the performance every time needed at the beginning of the public usage of the app.

7.1.1.3.2 Animal Welfare

Ensuring animal welfare is central to our project, but it’s also one of the hardest things to guarantee. If the platform makes it too easy to adopt an animal without proper vetting, it could lead to impulsive decisions and, eventually, returned adoptions. This is not only stressful for the animals but also undermines our goal of creating lasting bonds

between adopters and their pets. So, we included features that educate adopters about the responsibilities of pet ownership and encourage thoughtful decisions like the quiz system, extra questions before adopting, and the educator chatbot system.

7.1.1.3.3 Adopter Privacy

Another critical ethical issue we addressed was the privacy of the adopters. Since we collect sensitive personal information and data preferences, protecting user data against breaches became a top priority. To ensure transparency and accountability, a legal agreement is digitally presented to users during the registration process, which they must accept before completing their registration. This agreement outlines how their data will be collected, used, and protected. To safeguard information from unauthorized access—including protection from system administrators (our development team) and other users—we have implemented encryption techniques alongside a role-based access control (RBAC) system. Only authorized users, based on their user type, can access specific data, minimizing exposure risks. In addition to these core measures, industry-standard privacy practices have been incorporated, such as GDPR compliance for handling personal data, and secure session management techniques. Our session management ensures that login sessions are securely stored in the application's protected storage and properly cleared upon logout, preventing unauthorized access after session termination.

Furthermore, users are explicitly asked for permission when accessing device-specific features. Before reaching functionalities that require camera usage, users must approve access to their device's camera. Similarly, permissions are requested to access the device's photo gallery and to send push notifications. In the future, cookie consent mechanisms will also be implemented to give users full control over any tracking or preference storage in accordance with evolving digital privacy regulations. These efforts, collectively, demonstrate our commitment to respecting and protecting user privacy at all stages, in line with best practices.

7.1.1.3.4 Fair Use of AI

Our chatbot is designed to provide ongoing support to adopters, but relying too much on automation could lead to problems. For example, if the chatbot gives generic or incorrect advice about training or healthcare, it could frustrate users or even harm the animals. Therefore, we ensure the chatbot is well-trained and capable of escalating complex queries to real experts. If a critical question is asked to the chatbot, the chatbot answers but also advises the user to talk with a professional.

7.1.1.3.5 Cultural Sensitivities

Adoption practices and attitudes toward animals vary widely across cultures, and we needed to account for this in our design. For instance, in some communities, animals might be valued more for their utility (like guarding) than companionship. While our platform should promote responsible adoption, it also needs to be flexible enough to accommodate these cultural differences without alienating users. Therefore we decided to upload every animal to the system with a story and tags. With stories and tags, shelters can emphasize the fundamental characteristics of the animal while uploading the animal.

7.1.2 Standards

In this project, it was important to follow certain standards in order to attain the goals of reliability, scalability and ethical and technical acceptability. The implementation of our solution is done in accordance with the international and regional standards of software development, data protection and artificial intelligence to make sure that the platform complies with the highest quality standards. In particular, we tried to implement an information security management system based on the ISO/IEC 27001 standard and guarantee the confidentiality of adopter and shelter's data during storage

and transferring. Additionally, GDPR for the users' data is followed so that the personal data is not misused and the process is more transparent. For software lifecycle processes, we followed IEEE 12207 standard. Therefore, there is consistency within the processes that were used throughout the development of the project.

One of the key distinctive features of our work is the establishment of the Pet Adoption Benchmark, which serves as a new evaluation and improvement model of animal adoption procedures. We believe that this benchmark will then be useful for the shelters and organizations around the world in the assessment of their operations. By adhering to the above standards, we aim to create a creative and responsible platform to meet the challenges in animal adoption.

7.2. Ethics and Professional Responsibilities

7.3. Teamwork Details

7.3.1 Contributing and functioning effectively on the team

Throughout the design and development processes, each team member contributed to the project effectively. We separated all the responsibilities equally among the team members with respect to each members' interests. In case of non-technical jobs such as arranging meetings, fund and sponsor search, carrying out discussion sessions with the innovation expert, associations, supervisors, UX experts and course instructors etc. we worked as a team. Before every meeting with sponsors or stakeholders, we discussed what to say and what we should gain from the meeting. Neither of us talked or behaved without the permission of other group members in such critical cases. Additionally, all team members contributed equally to the reporting part of the project. Therefore, to sum up, it can be said that all 5 team members contributed and functioned equally and followed the distributed tasks carefully for all aspects of the project from technical and non-technical portions.

7.3.2 Helping creating a collaborative and inclusive environment

From the outset, our team has prioritized an environment of open communication and inclusivity. We employ collaborative tools such as GitHub and Notion to maintain transparency in our workflow, and we hold regular brainstorming sessions where every voice is heard. This approach not only encourages sharing of ideas and constructive feedback but also fosters mutual respect for diverse perspectives and expertise. By actively engaging in both scheduled and ad-hoc discussions, we have successfully built an atmosphere where team members feel comfortable proposing innovative solutions, challenging assumptions, and offering help when needed. This culture of inclusivity has not only enhanced our problem-solving capabilities but has also solidified the team's commitment to the project's success.

7.3.3 Taking lead role and sharing leadership on the team

After deciding on the main features of the project, 2 or 3 people are assigned to implement those features. All of these subgroups had their own leader and the leader made all technical research and implementation decisions. To be more specific, Yasemin is the leader of chatbot part, Alphan is the leader of the platform implementation, Deniz Can is the leader of the backend part, Oğulcan is the leader of the recommendation algorithm, and Ilhami is the leader of the database and microservice usage. Overall, in the meetings with the sponsors and stakeholders all of the members talked, expressed their ideas. However, Alphan stood out as the spokesman of the group.

7.4 Meeting objectives

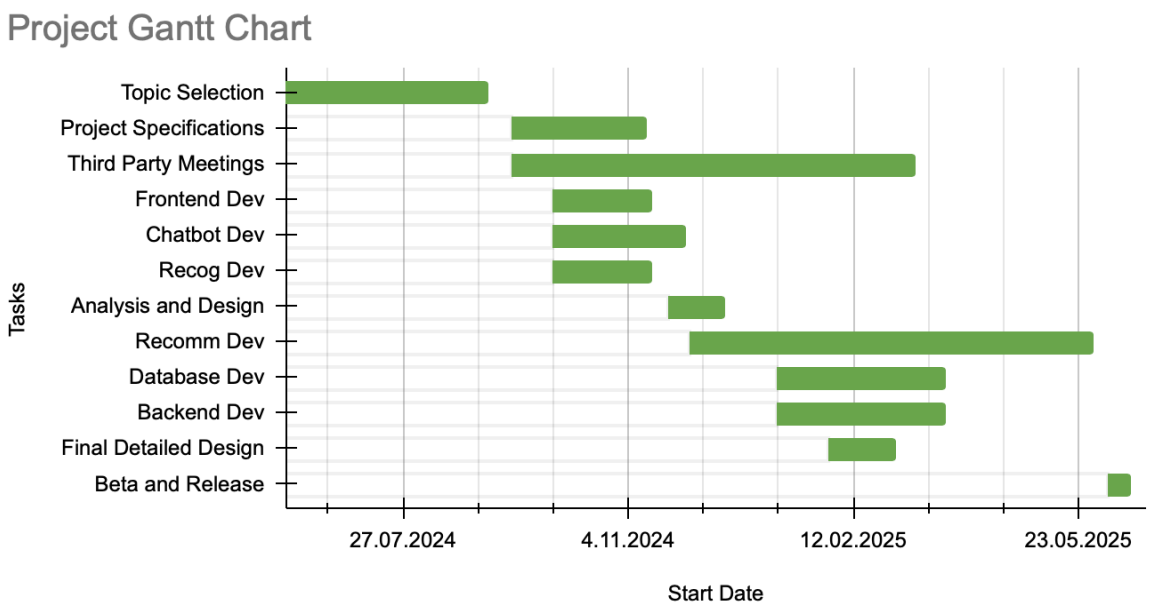

Figure 4: Project Plan Made in the Analysis Phase of the Project as a Gantt Chart

Throughout this project, we carefully followed the milestones we initially defined in the Requirements and Analysis Report, and tracked our progress using the Gantt chart shown in Figure 1. The topic selection and project specifications phases were successfully completed during the summer, laying a strong foundation for our work. Our third-party meetings, which are essential for shaping realistic goals and understanding user expectations, occurred exactly as planned and proved instrumental in making critical decisions, such as the removal of the quiz module. Initially included to increase user awareness, the quiz system was eliminated based on feedback from mentors and stakeholders who highlighted concerns about user fatigue. Nevertheless, we retained the intent of the quiz by distributing its informative questions throughout the registration and adoption form processes, maintaining the goal of informed adoption while improving usability.

All core development services—Frontend, Chatbot, Recognition, Recommendation, Database, and Backend, began within the planned timeframes, and though development iterations continued through to the final demo phase, each module reached a demonstrable and functional state by the time of our final presentation. The analysis and design phase also aligned with our original timeline, providing architectural clarity early on. The Final Detailed Design was postponed due to the senior project timeline's regulatory structure, which shifted formal documentation to the later stages. Finally, the Beta and Release objective was met as we completed integration testing and deployed a fully operational beta version of the platform. Overall, despite minor strategic pivots and adaptive scheduling, we successfully met nearly all of our planned objectives with high fidelity to our initial roadmap.

7.5 New Knowledge Acquired and Applied

Throughout the development of this project, the team acquired valuable expertise in the fine-tuning of machine learning models, specifically tailored for dog breed classification. By systematically experimenting with a range of hyperparameter configurations—including learning rates, model architectures, and regularization techniques—we developed a deeper understanding of how to optimize model performance in the context of real-world data variability. To store and retrieve images, we learned how to use AWS S3 Bucket. To understand the working mechanism and the

efficient usage techniques, we examined publicly shared projects on GitHub, tutorials on YouTube, and the detailed documentation of the AWS. Also the image processing library “sharp” is learnt from the documentations of it to process and modify the loaded images. To develop, implement, scale and maintain the chatbot service, we learned how to use frameworks leveraging LLMs for building intelligent AI systems like LangChain, LangGraph, LangSmith. Also, we learned how to leverage from RAG and embedding documents to cloud vector-stores efficiently. To learn and implement, we followed documentations of platforms, looked through example implementations and watched educational videos online on “How Tos”. Learned general industrial regulations and usual techniques applied when developing scalable systems and tried to apply them to our case. We asked and searched for answers to questions like “How to build secure and efficient systems?”. Efficiency is a crucial aspect while developing and it still is on this day. We learned to measure time and space costs very tightly and give attention to them for the first time since this is the first time we are developing something that has a real-end user and thus needs to be scalable and efficient. To ensure efficiency, we researched ways to keep resource usage to a minimum and strictly took care to implement them. We have learned how to implement a Push Notification service which is a platform that enables applications to send real-time alerts or messages directly to a user’s device, even when the app is not actively being used. It was a crucial aspect for the mobile platform side of our application and we followed some explanatory tutorial videos for proper implementation. We learned writing native applications that are cross-platform using frameworks built for this such as React Native and Expo Go for ease of development and sophisticated production tools they provide. To learn developing through them, we again watched a lot of tutorials, followed documentations and checked similar example implementations. The deployment and production phase were very unknown areas to us. However, as per the requirements of this project, we were obliged to learn and discover deployment technologies too. We also developed a recommendation engine that translates user questionnaire answers into simulated “mock” profiles, computes similarity against our dog database, and delivers personalized match suggestions. To ensure both relevance and variety, we experimented with clustering dogs by their traits and with a simple nearest-neighbors approach, even applying a location-based bonus to boost nearby matches. We also learned how to choose the appropriate type of database depending on specific application needs. This included understanding the strengths and limitations of SQL versus NoSQL databases—such as the structured querying and data integrity advantages of SQL, and the flexibility and scalability benefits of NoSQL. In particular, we gained hands-on experience with MongoDB, learning how to design collections, structure documents, and perform queries efficiently.

8. Conclusion and Future Work

As a result of recent legislative developments that are decreasing the welfare of stray animals, this project was initiated with the central objective of enabling every cat and dog to find a safe and sustainable home. We have successfully transitioned into the production phase of our application. All core functionalities have been fully developed and are functioning robustly. These features not only enhance the adoption process but also significantly reduce the operational burden on shelters and rescue organizations.

We have already established active collaborations with key stakeholders, including Kurtaran Ev, Etimesgut Municipality, and Bilkent Hayvan Dostları. These partnerships have allowed us to operate effectively in both Ankara and Istanbul, two of Türkiye’s most densely populated metropolitan areas. This early integration into major urban ecosystems has provided us with both the visibility and the operational insights needed to scale the platform responsibly.

For the future, our strategic roadmap involves expanding the platform's reach to additional municipalities, beginning with formal negotiations with the Istanbul Metropolitan Municipality and Şişli Municipality. These discussions will focus on building long-term public/private collaborations aimed at institutionalizing the

adoption platform within local governance frameworks. Our medium-term goal is to achieve nationwide coverage by extending services to all 81 provinces in Türkiye.

Once a robust national framework has been established, we intend to initiate international expansion efforts, targeting regions facing similar challenges related to stray animal welfare. Throughout this process, our guiding vision will remain unchanged: to ensure that no stray animal is left without a home, and that every adoption is a step toward a more compassionate and structured approach to animal welfare.

By remaining committed to technological innovation, cross-sector collaboration, and ethical responsibility, we will continue to advance our mission, one that prioritizes both efficiency and empathy in solving a deeply human and societal issue.

9. Glossary

AI (Artificial Intelligence): The simulation of human intelligence processes by machines, especially computer systems, enabling them to perform tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and language translation [3].

API Gateway: A server that acts as an API front-end, handling client requests, routing them to appropriate backend services, and managing tasks such as authentication, rate limiting, and analytics [3].

AWS S3 (Amazon Simple Storage Service): A scalable cloud storage service used for storing and retrieving media files such as images and videos, offering durability, redundancy, and access control features [3].

Backend: The server-side part of an application responsible for processing requests, managing databases, handling authentication, and ensuring business logic execution. It communicates with the frontend to deliver data and functionality to users [4].

Cert-Manager: A Kubernetes add-on that automates the management and issuance of TLS/SSL certificates from certificate authorities, ensuring secure HTTPS traffic routing.

Chatbot: A software application designed to simulate human conversation, allowing users to interact with digital devices as if they were communicating with a real person [4].

CI/CD (Continuous Integration/Continuous Deployment): A set of software development practices that enable frequent code integration, automated testing, and deployment to production environments with minimal manual effort.

CloudFront: A content delivery network (CDN) provided by AWS that delivers data, videos, applications, and APIs to users globally with low latency and high transfer speeds.

CNN (Convolutional Neural Network): A class of deep neural networks commonly applied to visual imagery analysis, using convolution operations to process and learn spatial hierarchies in data [5].

Cross-Platform Application: Software designed to operate on multiple computing platforms (e.g., iOS, Android, Web) without requiring separate codebases for each [3].

Data Privacy: The practice of handling and processing data in a manner that ensures the confidentiality and protection of personal information from unauthorized access or disclosure [3].

Deep Learning: A subset of machine learning involving neural networks with many layers that can learn and make intelligent decisions by analyzing large volumes of data [3].

EAS (Expo Application Services): A set of cloud-based tools from Expo used for building, updating, and submitting mobile applications. Includes services such as EAS Build, EAS Submit, and EAS Update.

Expo: A framework for building React Native applications with a managed workflow that includes pre-built libraries and tools, enabling cross-platform development without native code modifications.

Expo Router: A file-based routing system for Expo and React Native apps, enabling developers to organize navigation logic using a directory structure that maps to application routes.

Frontend: The client-side of an application that interacts directly with users, including the user interface and user experience design [3].

GCP (Google Cloud Platform): A suite of cloud services provided by Google for computing, storage, networking, and machine learning. Used for hosting and scaling cloud-native applications like Köpük.

GKE (Google Kubernetes Engine): A managed Kubernetes service from Google Cloud that automates the deployment, scaling, and management of containerized applications.

Helm: A package manager for Kubernetes that simplifies application deployment by using charts—pre-configured Kubernetes resources.

Ingress-NGINX: A Kubernetes ingress controller using NGINX that manages external HTTP/HTTPS traffic and routes it to services inside a Kubernetes cluster.

JWT (JSON Web Token): A compact and secure token format used for user authentication and authorization in web applications.

Kubernetes (K8s): An open-source system for automating deployment, scaling, and management of containerized applications. Used in Köpük to orchestrate backend microservices.

LangChain: A framework for developing applications powered by large language models (LLMs), providing integration with APIs, tools, and external knowledge sources [4].

LangGraph: A tool for managing stateful flows and graph-based logic around LLMs, enabling advanced multi-step interactions and contextual memory [4].

Lazy Loading: A design pattern used to defer loading of non-essential resources (like images) until they are needed, improving performance and load speed.

LLM (Large Language Model): A type of AI model trained on vast text corpora to understand and generate human-like language [6].

ML (Machine Learning): A branch of AI that involves creating algorithms capable of learning patterns from data and making decisions or predictions without explicit programming [3].

Mongoose: An object data modeling (ODM) library for MongoDB and Node.js, providing a schema-based structure for data and simplifying database interactions.

MongoDB: A NoSQL document database used to store flexible and dynamic data structures such as user profiles, animal records, and chat logs in JSON-like documents.

OAuth: An open standard for access delegation that allows secure third-party access to user resources without exposing login credentials [4].

RAG (Retrieval-Augmented Generation): A technique that combines retrieving relevant data from external sources and generating responses using language models to produce accurate and contextually informed outputs [8].

RBAC (Role-Based Access Control): A security model that restricts system access to authorized users based on their roles, ensuring users only access what's relevant to them [9].

React Native: An open-source framework for building native mobile applications using JavaScript and React, supporting both iOS and Android platforms [4].

Recommendation System: Software that provides personalized suggestions to users based on preferences, behaviors, or similarities, using collaborative, content-based, or hybrid filtering approaches [7].

SQL (Structured Query Language): A standard programming language used to manage and manipulate relational databases. It allows querying, updating, and managing structured data efficiently [3].

10. References

[1] “Turkey Approves Law to Remove Stray Dogs from Streets. Opposition Vows to Fight the ‘Massacre Law’,” AP News. [Online]. Available: https://apnews.com/article/turkish-parliament-approves-law-stray-dogs-a5870dbff7066f0d11f34758b39bcf5f?utm_source=chatgpt.com. [Accessed: November 20, 2024].

[2] "YOD-AI Documentation," YOD-AI. [Online]. Available: https://yod-ai.github.io/yodai/#. [Accessed: November 19, 2024].

[3] IBM, [Online]. Available: https://www.ibm.com. [Accessed: November 21, 2024].

[4] freeCodeCamp, “Learn to Code — For Free.” [Online]. Available: https://www.freecodecamp.org/. [Accessed: November 19, 2024].

[5] LeCun et al., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] T. B. Brown et al., "Language Models are Few-Shot Learners," in *NeurIPS*, 2020.

[7] X. Su and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," *Advances in Artificial Intelligence*, vol. 2009, pp. 1–19, 2009.

[8] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *NeurIPS*, 2020.

[9] R. S. Sandhu et al., "Role-Based Access Control Models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.