CALIFORNIA STATE UNIVERSITY
FULLERTON
COLLEGE OF ENGINEERING
AND COMPUTER SCIENCE

Rescue Rovers Mobile Application

Addressing Dog Overpopulation and Euthanasia

By:

Dharmesh Tewari (CWID: 885550574)
Hina Moin Syed (CWID: 885685420)

Department of Computer Science
College of Engineering and Computer Science (ECS)

A Project Report
Submitted to the Department of Computer Science for the
Study Leading to a Project Report in Partial Fulfillment of the requirements for the
CPSC 597 and Award of the Degree of Master of Science in Computer Science of
California State University, Fullerton

Supervisor:
Prof. Lidia Morrison

Department of Computer Science
College of Engineering and Computer Science (ECS)

California State University, Fullerton
Spring, 2024

CERTIFICATE

This is to certify that the Computer Science Project titled **"RESCUE ROVERS MOBILE APPLICATION"** submitted by Dharmesh Tewari and Hina Moin Syed in the partial fulfillment of the requirements for the Degree of Master of Science in Computer Science, is an authentic and genuine work carried out by them under our supervision. The Engineering Project and its corresponding detailed Project Report are records of the candidates' own work that was carried out under our own guidance.

Date:

Lidia Morrison

Professor,

Department of Computer Science,

California State University, Fullerton

Place:

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it neither contains any material previously published or written by another person nor material which has to a substantial extent been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

Signature:

Name: Dharmesh Tewari

CWID: 885550574

Date: May 11, 2024

Signature:

Name: Hina Moin Syed

CWID: 885685420

Date: May 11, 2024

ACKNOWLEDGMENT

We express our deepest gratitude to our project supervisor, Lidia Morrison, whose unwavering support and inspiration have been integral to the success of this endeavor. Her guidance and encouragement have been instrumental in navigating the challenges of our study, and we are truly thankful for her dedication to our progress.

Special thanks are extended to our friends and families for their unwavering support and understanding during the demanding moments and stresses of the research project. Their encouragement has been a source of strength, enabling us to persevere and succeed.
We are grateful to California State University, Fullerton for providing us with the opportunity to collaborate as a team, fostering an environment that promotes teamwork and communication skills. This experience has been invaluable, enriching our academic journey and preparing us for future challenges.

Additionally, we appreciate the individual contributions of each group member, whose teamwork and solidarity have been essential to our collective achievements. Together, we have navigated obstacles and celebrated successes, forging strong bonds along the way.
In conclusion, we extend our heartfelt thanks to all who have supported us throughout this project. Your contributions have been invaluable, and we are honored to have had the opportunity to work alongside such dedicated individuals.

ABSTRACT

Rescue Rovers represents a comprehensive digital platform meticulously crafted to enrich the dog adoption process through fostering collaboration between Adoption Centers and Foster Homes. This sophisticated system is intricately designed to manage every aspect of the adoption journey, from the initial registration of dogs to their placement in loving homes.

Central to Rescue Rovers is its provision of a robust database for Adoption Centers, offering detailed canine profiles comprising comprehensive descriptions, medical histories, and behavioral assessments. This facilitates precise matching of dogs with potential adopters, thus enhancing the likelihood of successful adoptions. The system's intuitive search function empowers Adoption Centers to efficiently locate suitable dogs based on specific criteria such as breed, size, and temperament.

Foster Homes benefit from a user-friendly portal within Rescue Rovers, enabling them to showcase their dogs, update their statuses, and communicate seamlessly with interested Adoption Centers. The integration of real-time mapping technologies, reminiscent of the Google Maps API, facilitates Foster Homes in identifying and connecting with nearby Adoption Centers, streamlining the process of visit arrangements or transfers.

Administration of the system is overseen by vigilant administrators, responsible for managing user access, resolving disputes, and upholding the integrity of the adoption process. They ensure strict adherence to legal requirements and ethical standards, thereby fostering a safe and equitable environment for all participants.

Moreover, Rescue Rovers places significant emphasis on community engagement and support, incorporating features such as direct messaging, notifications, and feedback mechanisms. This cultivates an interconnected community driven by a shared passion for animal welfare. The system's design reflects an unwavering commitment to continual enhancement, with regular updates informed by user input and technological advancements.

In summary, Rescue Rovers transcends being a mere platform; it embodies a mission to revolutionize the dog adoption landscape, striving to make it more accessible, transparent, and successful for all stakeholders involved.

Table of Contents

Table of Figures

CHAPTER 1

1. INTRODUCTION

The purpose of this project is to help dog foster and rescue groups. Rescue Rovers connect potential adopters to the dogs available for adoption. As a group, Rescue Rovers takes dogs in from high kill shelters and places them into foster homes to be cared for. The group hosts bi-weekly adoption events where fosterers can bring their dogs to a public space for potential adopters to meet. Over the past 10 years, the group has rescued and found homes for over 10,000 dogs, with more dogs being taken each month.

1.1 Problem Domain

Dog overpopulation is a major problem worldwide, leading to high rates of euthanasia, abandonment, and mistreatment of animals. The American Society for the Prevention of Cruelty to Animals (ASPCA) reports that each year approximately 6.5 million companion animals are received by animal shelters across the United States, out of which approximately 3.3 million are dogs, and 1.6 million dogs are euthanized. High-kill shelters, which are often overcrowded, are forced to euthanize dogs due to a lack of space or resources. To combat this issue, the Rescue Rover program rescues dogs from high-kill shelters and works to place them in loving homes. To develop Rescue Rover, I have surveyed the current advances in the field, including existing adoption platforms and animal welfare programs. My research has shown that while there are several adoption platforms available, they often lack features that can make the adoption process more efficient and user-friendly. Additionally, many animal welfare programs focus on spaying and neutering but do not provide adequate support for adoption.

1.1.1 Key Issues in the Problem Domain and Background Information

One of the key issues in the problem domain is the lack of resources and space in shelters to accommodate the enormous number of dogs that are surrendered or found as strays. This results in overcrowding and high rates of euthanasia. Another issue is the lack of public education on the importance of spaying and neutering pets, which contributes to the overpopulation of dogs in shelters. Additionally, some breeds are often targeted for euthanasia due to misconceptions about their behavior or temperament.

Rescue Rover was founded in 2015 by a group of animal lovers who were concerned about the high rates of euthanasia in high-kill shelters. The organization is based in Utah, and partners with shelters in the area to rescue dogs and find them loving homes. The Rescue Rover team comprises volunteers who socialize and train dogs, improving their adaptability. The organization also provides medical care and vaccinations to dogs to ensure their health before adoption. Rescue

Rovers is primarily based around Facebook and use it as their main form of communication. When a dog is available for adoption, they get advertised through the group's Facebook page. If a potential adopter does not use Facebook regularly, it's difficult to know when a new dog is posted. Our project hopes to provide the group with a platform off Facebook to advertise their dogs too. In addition, there is no way to be notified when the right dog for you is available for adoption.

1.1.2 Current Advances and Technologies Associated with the Problem Domain

Animal rescue organizations, such as Rescue Rover, can now leverage technological advancements to connect with prospective adopters and increase awareness about their dogs. Social media platforms, including Twitter, Facebook, and Instagram, have emerged as popular channels for animal welfare organizations to showcase their animals and attract potential adopters. Furthermore, medical technology has improved significantly, enabling the delivery of advanced medical care to dogs, thus improving their overall health, and increasing their chances of finding a new home.

1.1.3 Problem Solution

My project will notify users when a dog they might be interested in is available for adoption, actively reaching out to potential adopters to get dogs adopted. On the administrator's side, running an organization is a difficult and manual process. With our project, I hope to alleviate some of the manual labor that goes into event management to both streamline the process as well as advertise adoption events to a larger audience.

The application will have three users:

- **Administrators:** These are the people running the Rescue Rovers organization, planning events, arranging adoptions, and finding fosterers. They need a way to stay better organized.
- **Fosterers:** They take dogs into their homes while they are awaiting adoption. They need a more efficient way to find adopters and send dogs to adoption events.
- **Adopters:** They adopt dogs that are being fostered. They need a better way of seeing what dogs are available at any time.

By creating a mobile app, I can connect potential adopters to dogs, give the group a platform off Facebook, and aid administrators in event management.

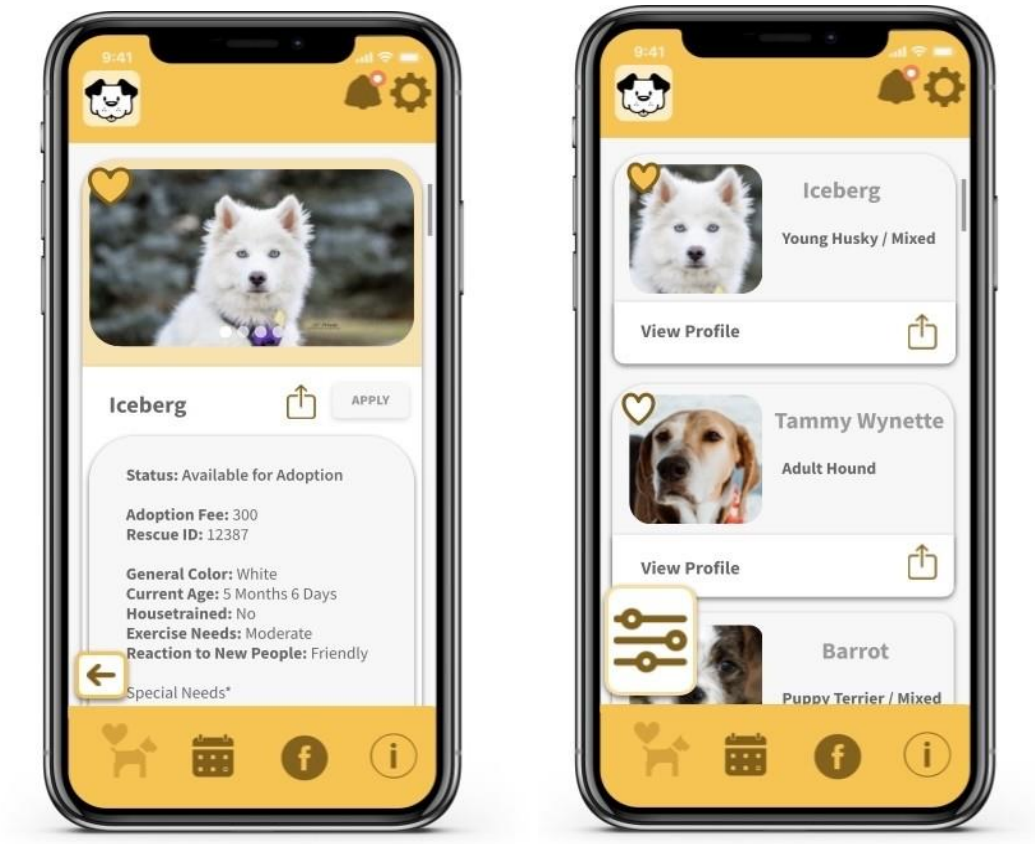

Figure 1: *An example of the application*

1.1.4 Challenges and Potential Research

There are similar programs to ours such as PetFinder and WeRescue that allow you to adopt dogs, but I have yet to see an app that incorporates foster care despite there being well over a hundred rescue organizations in the state of Utah alone. Our software will have an adoption feature much like currently available apps, but it will also have a more specific and local focus. Our app will be focused on one operation which will provide a quality of information that cannot be beaten. Other apps might service larger areas, but ours will help people find the perfect rescue for their home. My app will also allow the organization to advertise its events directly to potential adopters along with which dogs will be attending. I will also allow fosterers and Rescue Rovers administrators to upload information directly through the app instead of requiring them to send updates to an outside party.

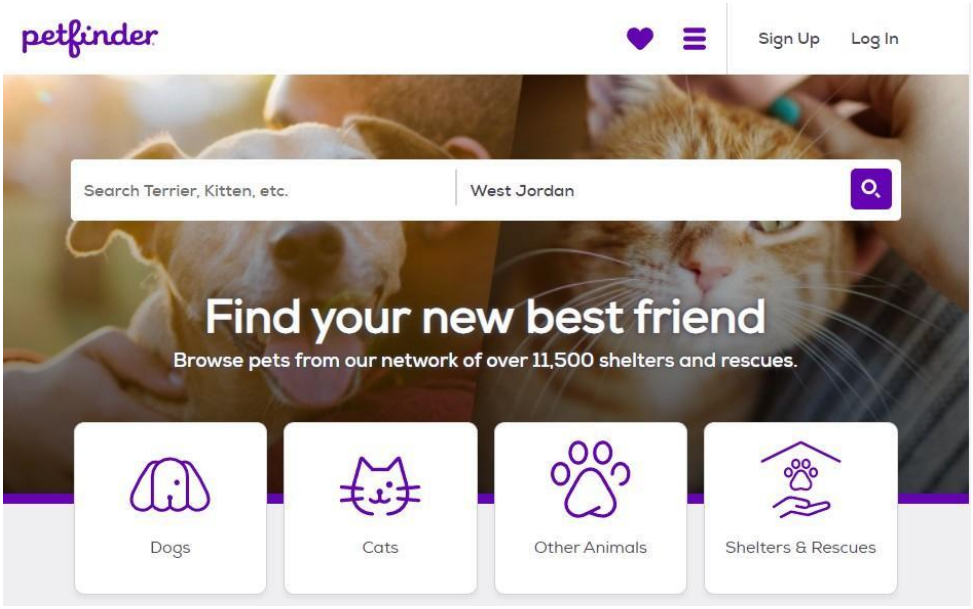

Figure 2: *Petfinder website*

Currently, the group uses Facebook to advertise adoption events and Google Docs to store information about the dogs in the system. This system is inefficient as Facebook use has declined recently and organizing through multiple Google Docs and Sheets requires updating and searching every document individually. Additionally, the current system does not provide a way to easily save information about the dogs in a printable format for adoption event information sheets, leading to time being wasted writing and printing previously available information. It also does not make it easy to see what dogs are attending, which makes it difficult for organizers to determine what equipment (e.g., dog crates) they need to bring to the event. This app will reduce the number of updates and searches necessary for event organization and will provide a way to easily produce and replicate adoption info sheets.

1.2 Project Objectives

The project objective is to create a user-friendly and efficient platform that addresses the issue of dog overpopulation and euthanasia. The primary goal is to develop an Android/IOS/Web application using Flutter that can be easily accessed by users on all three devices. The application will have a mobile-first approach and a user-friendly interface that is intuitive and easy to navigate.

The application will have a server-based back end developed on Firebase, which is a scalable and proven back-end technology. The application will use two databases, including the Firestore (NoSQL) database from Rescue Rovers that contains information about every dog that has been

through the organization, and a supplemental MongoDB database that will hold information about users and events that are not stored in the Rescue Rovers database. The two databases will be synced regularly to avoid the need for constant polling of the Rescue Groups database.

The application will be available for download on the App Store and Play Store, and users will be able to access additional features and conveniences through the web application (stretch goal). During the development process, an older laptop will be used as a server, with the intention of moving to an AWS machine in the future. The overall objective of the Rescue Rover application is to provide a solution that improves the efficiency of the adoption process, reduces dog overpopulation and euthanasia rates, and provides a convenient and user-friendly platform for adopters, animal shelters, and rescue organizations.

CHAPTER 2

2. LITERATURE SURVEY

The following papers were surveyed to gain a better understanding of the Rescue Rover program and the problem domain it addresses:

- "Rescue Dogs: Health, Welfare, and Selection" by Christine Arhant and Jörg Aurich (2016). This paper discusses the health and welfare issues that can arise when rescuing dogs from shelters and the importance of selecting dogs that are healthy and have a good temperament.
- The Impact of Shelter Housing on Dog Welfare: A Critical Review" by Emily Weiss, Heather Mohan-Gibbons, and Margaret Slater (2016). This paper reviews the research on the impact of shelter housing on dog welfare and provides recommendations for improving the welfare of shelter dogs.
- "The Use of Social Media in Animal Advocacy" by John J. Hetzler and Gary J. Patronek (2013). This paper explores the use of social media in animal advocacy and discusses the potential benefits and challenges of using social media to promote animal welfare.
- "Preventing Dog Bites: The Need for a Coordinated Community Response" by Karen L. Overall (2017). This paper discusses the issue of dog bites and the importance of a coordinated community response to prevent them.
- "Socialization of Puppies and Adult Dogs" by Ilana R. Reisner (2016). This paper discusses the importance of socialization for puppies and adult dogs and provides recommendations for socializing dogs to prevent behavioral problems.

2.1 Comparison of Current Advances Surveyed

The surveyed papers shed light on various facets of the problem domain and Rescue Rover's program. Arhant and Aurich's paper underscore the significance of selecting dogs that are both physically healthy and have a good temperament when rescuing from shelters. Weiss, Mohan Gibbons, and Slater's paper highlights the importance of enhancing the welfare of shelter dogs and offers suggestions for achieving this objective. Hetzler and Patronek's paper examines the potential advantages and challenges of leveraging social media to promote animal welfare, which is a technique Rescue Rover uses extensively. Overall's paper stresses the significance of a coordinated community response to prevent dog bites, which has implications for the adoption process and educating potential adopters. Reisner's paper underscores the importance of socialization to prevent behavioral issues, which is a crucial aspect of the Rescue Rover program.

2.2 Mechanics of Paper

The paper presents a concise and informative introduction and summary of the Rescue Rover program and its target problem domain. It offers a well-written account of the major issues and background information related to the domain, and in-depth discussion of the recent technological advancements and developments associated with it. The surveyed papers are properly referenced and analyzed, offering valuable perspectives into the Rescue Rover program and the problem domain. The comparison of the technological advancements assessed provides a comprehensive understanding of the primary issues and advancements in the problem domain.

CHAPTER 3

3. RELATED WORK

The Rescue Rovers Dog Adoption System is built upon a foundation of existing applications and research in the field of animal welfare and technology. This section outlines the related work that has influenced the development of Rescue Rovers, providing insights into the system's innovative approach to dog adoption.

3.1 Existing Adoption Platforms

- **Angel City Pit Bulls:** Utilizes a generic adoption application that collects comprehensive information from potential adopters, including personal references and home environment details.
- **Rescue A Dog:** Offers a suite of forms for various purposes such as adoption, fostering, and surrendering dogs. Their adoption application is designed to gather extensive information to ensure the best match between the dog and the adopter.
- **Cognito Forms:** Provides a free pet adoption application form template that intelligently adjusts to whether applicants are applying to adopt a cat or a dog, with follow-up questions based on the responses provided.
- **JotForm:** Offers a dog adoption application form template that can be used by shelters or rescues to streamline the process of gathering applicant information.

3.2 Animal Welfare Research

- **Best Practices:** The system incorporates findings from academic and industry research on animal welfare, focusing on the psychological and physical well-being of dogs.
- **Ethical Standards:** Rescue Rovers adheres to ethical standards in dog adoptions, ensuring that the matching process is conducted with the utmost care for the animals' needs.

3.3 Technological Advancements

- **Innovative Features:** Rescue Rovers integrates advanced features such as real-time mapping and search algorithms, inspired by the latest technological developments.
- **System Scalability:** The platform is designed to be scalable, considering the growing needs of users and the increasing volume of data.

3.4 Community Engagement and Feedback

- **User Surveys:** Feedback from potential users has been instrumental in shaping the features and functionalities of Rescue Rovers, ensuring that it meets the community's needs.
- **Collaborative Efforts:** Partnerships with animal welfare organizations and technology experts have enriched the system's development, making it a community-driven project.

3.5 Legal and Regulatory Considerations

- **Compliance:** Rescue Rovers is developed with a keen awareness of the legal framework governing dog adoptions, ensuring full compliance with regulations.
- **Data Privacy:** The system prioritizes user data protection and privacy, incorporating best practices in cybersecurity to safeguard sensitive information.

Through the examination of related applications and research, Rescue Rovers has been crafted to offer a unique and comprehensive solution to the challenges of dog adoption. It stands as a testament to the collaborative effort between technology and animal welfare communities, aiming to create a better future for dogs and adopters alike.

3.6 Overview of Existing Systems

The landscape of dog adoption platforms is diverse, with various applications offering a range of services to facilitate the connection between dogs and potential adopters. In this section, we will review the existing systems mentioned previously, analyzing their strengths and identifying opportunities for the Rescue Rovers system to fill gaps and enhance the adoption experience.

3.6.1 City Pit Bulls

- **Strengths:** Provides a detailed adoption application that collects extensive information from potential adopters, ensuring a thorough vetting process.
- **Limitations:** The platform may lack real-time interaction capabilities between adopters and the organization, which can slow down the adoption process.

3.6.2 Rescue A Dog

- **Strengths:** Offers a comprehensive suite of forms for various stages of the adoption process, including fostering and surrendering, which simplifies paperwork.
- **Limitations:** May not provide a centralized database for dog profiles, which can lead to difficulties in tracking and managing dog information.

3.6.3 Cognito Forms

- **Strengths:** Offers a customizable pet adoption application form that adjusts based on the type of pet, streamlining the application process.
- **Limitations:** As a form builder, it lacks integrated communication tools and a dedicated platform for adoption management.

3.6.4 JotForm

- **Strengths:** Provides a template for dog adoption applications that can be easily used by shelters or rescuers, aiding in standardizing the application process.
- **Limitations:** Similar to Cognito Forms, it does not offer a comprehensive system for managing the entire adoption workflow.

3.7 Opportunities for Rescue Rovers

- **Real-Time Communication:** Implementing a messaging system that allows for instant communication between Adoption Centers and Foster Homes, expediting the adoption process.
- **Centralized Dog Profiles:** Creating a unified database for all dog profiles, complete with medical records and behavioral assessments, accessible by all stakeholders.
- **Integrated Management Platform:** Offering a full-fledged platform that encompasses user management, dog profile handling, communication, and support, all in one place.
- **Community Building:** Facilitating a sense of community among users by providing forums, success stories, and resources for post-adoption support.
- **Advanced Matching Algorithms:** Utilizing data-driven algorithms to match dogs with suitable adopters based on compatibility scores derived from detailed profiles and preferences.

By addressing the limitations of existing systems, Rescue Rovers aims to provide a more holistic and efficient platform for dog adoptions. The system leverages technology to ensure that every dog finds a loving home and that every stakeholder has a positive and fulfilling experience.

CHAPTER 4

4. PROBLEM STATEMENT AND CONCEPT

In the realm of dog adoption, prospective pet owners and animal welfare organizations face a myriad of challenges that hinder the efficiency and effectiveness of the adoption process. The existing applications, while serving as valuable tools, exhibit several shortcomings that Rescue Rovers aims to address.

4.1 Problems Identified in Existing Systems:

- **Fragmented Communication:** Current platforms lack a unified system for real-time communication between Adoption Centers and Foster Homes, leading to delays and potential miscommunications.
- **Inadequate Profile Management:** Many applications do not offer a centralized, easily accessible database for dog profiles, complicating the tracking and updating of vital information.
- **Limited Search Functionality:** The search capabilities of existing systems are often basic, making it difficult for Adoption Centers to find suitable matches based on specific criteria.
- **Lack of Community Engagement:** There is a noticeable absence of features that promote community building and post-adoption support within the current ecosystem of adoption services.
- **Our Concept:** Rescue Rovers introduces a novel concept that revolves around a holistic, integrated platform designed to overcome the identified challenges. The concept is built on the pillars of connectivity, convenience, and community.

4.2 Solutions Proposed by Rescue Rovers:

- **Enhanced Communication Tools:** By incorporating advanced messaging systems and notifications, Rescue Rovers facilitates immediate and direct dialogue between all parties involved.
- **Comprehensive Dog Profiles:** A centralized database will be established, featuring detailed profiles that include medical records, behavioral assessments, and real-time availability status.
- **Advanced Search and Matching:** Utilizing sophisticated algorithms, Rescue Rovers will offer enhanced search options that allow for precise filtering and smarter matching of dogs with potential adopters.

- **Community-Centric Features:** The system will integrate forums, success stories, and resources for post-adoption, fostering a supportive network for all users.

By addressing these core issues with innovative solutions, Rescue Rovers sets out to redefine the dog adoption experience, making it more streamlined, transparent, and community focused. The system is not just a tool but a catalyst for change, promoting responsible pet ownership and compassionate animal care.

CHAPTER 5

5. TARGET AUDIENCE

The success of the Rescue Rovers Dog Adoption System hinges on its ability to meet the needs of its diverse target audience. Each group has distinct interactions with the system, and understanding their specific requirements is crucial for the system's design and functionality. Below are the key target audiences and a detailed explanation of each.

5.1 Admins

- **Role and Responsibilities:** Admins are the backbone of the Rescue Rovers system, responsible for overseeing the entire operation. They manage user accounts, ensure the smooth functioning of the system, and resolve any disputes that arise.
- **Needs and Requirements:** Admins require robust tools for monitoring system activity, comprehensive reporting features, and the ability to intervene in the adoption process when necessary. They also need access to analytics to make informed decisions about system improvements.

5.2 Adoption Centers

- **Role and Responsibilities:** Adoption Centers are the primary interface for potential adopters. They browse the database of available dogs, initiate contact with Foster Homes, and facilitate the adoption process.
- **Needs and Requirements:** Adoption Centers need a user-friendly interface to search for dogs based on specific criteria, efficient communication tools to interact with Foster Homes, and a streamlined process for documenting adoptions.

5.3 Foster Homes

- **Role and Responsibilities:** Foster Homes provide temporary care for dogs and play a critical role in preparing them for adoption. They register dogs on the system, manage their profiles, and coordinate with Adoption Centers.
- **Needs and Requirements:** Foster Homes require the ability to easily update dog profiles, including photos and health information, and a convenient way to receive and respond to inquiries from Adoption Centers. They also benefit from features that help them locate nearby Adoption Centers.

5.4 Potential Adopters

- **Role and Responsibilities:** While not direct users of the system, potential adopters are an essential part of the adoption ecosystem. They rely on the information provided by Adoption Centers to make adoption decisions.
- **Needs and Requirements:** Potential adopters look for detailed dog profiles, transparent adoption procedures, and access to support and resources that can help them after the adoption.

5.5 Animal Welfare Organizations

- **Role and Responsibilities:** These organizations advocate for the well-being of animals and often work closely with Adoption Centers and Foster Homes to promote responsible adoption practices.
- **Needs and Requirements:** They require access to data and reports that can help them understand adoption trends and identify areas where they can provide support or resources.

5.6 Veterinary Services

- **Role and Responsibilities:** Veterinarians and veterinary clinics provide medical care for dogs in the system and may also offer advice to potential adopters.
- **Needs and Requirements:** They need a reliable way to access and update medical records within the system and communicate with Foster Homes and Adoption Centers regarding the health status of dogs.

By thoroughly understanding the needs and roles of each target audience, Rescue Rovers can tailor its features and functionalities to provide a seamless and efficient adoption experience for all involved.

CHAPTER 6

6. FEATURES

Rescue Rovers is designed to be a comprehensive system that caters to all aspects of the dog adoption process. The following features are integral to achieving the system's objectives and ensuring a smooth experience for all users:

6.1 User Management

- **Account Creation**: Secure registration process for Admins, Adoption Centers, and Foster Homes.
- **Profile Customization:** Users can personalize their profiles with relevant information and preferences.
- **Access Control:** Admins can manage user permissions and access levels.

6.2 Dog Profile Management

- **Detailed Dog Profiles:** Each dog has a profile with photos, medical history, temperament, and other essential details.
- **Real-Time Updates:** Foster Homes can update dog profiles as needed to reflect status and information.
- **Adoption Tracking:** Adoption Centers can track the adoption status and history of each dog.

6.3 Search and Matching

- **Advanced Search Filters:** Users can search for dogs based on a variety of criteria such as breed, age, size, and health status.
- **Smart Matching Algorithms:** The system suggests potential matches between dogs and adopters based on compatibility.

6.4 Communication Tools

- **Direct Messaging:** A built-in messaging system allows for real-time communication between Adoption Centers and Foster Homes.
- **Notifications:** Users receive notifications about important updates, inquiries, or system changes.

6.5 Community Engagement

- **Forums and Discussion Boards:** Spaces for users to share experiences, seek advice, and discuss topics related to dog adoption.
- **Success Stories:** A section dedicated to showcasing successful adoptions and positive outcomes.

6.6 Support and Resources

- **Help Center:** A comprehensive resource for users to find answers to common questions and guidance on system usage.
- **Post-Adoption Support:** Information and resources available for new pet owners to help with the transition after adoption.

6.7 Administrative Tools

- **Reporting and Analytics:** Admins can generate reports on system usage, adoption rates, and user activity.
- **Dispute Resolution:** Features to assist in resolving conflicts between users, ensuring a fair and positive experience.

6.8 Legal and Compliance

- **Data Protection:** The system adheres to strict data privacy regulations to protect user information.
- **Regulatory Compliance:** Ensures that all aspects of the adoption process comply with relevant laws and ethical standards.

6.9 Integration with External Services

- **Mapping Services:** Integration with mapping APIs to help users locate nearby Adoption Centers or Foster Homes.
- **Veterinary Partnerships:** Links with veterinary services for easy access to medical records and health checks.

These features are carefully crafted to address the needs of the target audience and to solve the problems identified in existing systems. Rescue Rovers is committed to providing a feature-rich platform that enhances the dog adoption process for everyone involved.

CHAPTER 7

7. TOOLS AND DEVELOPMENT ENVIRONMENT

For the development of the Rescue Rovers mobile application, a suite of modern tools and technologies will be utilized to ensure a robust, scalable, and user-friendly product. The following section outlines the key tools and the development environment setup for the project.

7.1 Development Tools:

- **Flutter:** As the primary SDK for building mobile applications, Flutter offers a rich set of pre-designed widgets and the ability to create a native user experience on both Android and iOS platforms from a single codebase.
- **Dart:** The programming language used by Flutter, Dart enables developers to write concise and effective code with a strong emphasis on UI composition and reactive programming.
- **Backend Services:**
    - **Firebase:** A comprehensive suite of services that provides backend functionality such as real-time database, authentication, and hosting. Key Firebase services to be used include:
    - **Firestore:** A NoSQL cloud database that will store and sync data for client- and server-side development.
    - **Firebase Authentication:** Provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users.
    - **Firebase Storage:** For storing and serving user-generated content such as photos and videos.
    - **Firebase Cloud Messaging:** To enable real-time notifications.
- **Version Control:**
    - **Git:** For tracking changes in the source code during development.
    - **GitHub:** As a repository hosting service, GitHub will be used for collaboration, code review, and managing the project's codebase.

- **Development Environment:**
    - **IDE:** The use of an Integrated Development Environment (IDE) like Android Studio or Visual Studio Code, which offers support for Flutter and Dart development.
    - **DevOps Tools:** Utilization of continuous integration and deployment (CI/CD) pipelines to automate the testing and deployment process.

- **Testing and Quality Assurance:**
    - **Unit Tests:** To test individual functions and methods.

- **Integration Tests:** To test complete flows and interactions between various parts of the application.
- **UI Tests:** To ensure that the application's interface functions correctly across different devices and screen sizes.

- **Design and Prototyping:**
  - **Figma:** For UI/UX design and prototyping, ensuring that the application's design is intuitive and user-friendly.

- **Project Management:**
  - **Trello:** For tracking tasks, sprints, and agile project management.

7.2 Hardware Requirements

To ensure the Rescue Rovers mobile application runs smoothly and provides an optimal user experience, certain hardware specifications are recommended for devices. These requirements are designed to accommodate the application's features and functionalities.

7.2.1 For Users:

- **Mobile Devices:**
  - **Operating System:** Android 5.0 (Lollipop) or later, iOS 11 or later.
  - **Processor:** Quad-core 1.4 GHz or higher.
  - **Memory:** At least 2 GB of RAM.
  - **Storage:** A minimum of 100 MB of free space for installation and additional space for data caching.
  - **Screen Resolution:** 1280x720 pixels or higher.
  - **Network:** Active internet connection with at least 3G speeds for real-time data synchronization.

7.2.2 For Admins and Adoption Centers (Web Portal Access):

- **Computers:**
  - **Operating System:** Windows 7/8/10, macOS X Yosemite (10.10) or later, or a modern Linux distribution.
  - **Processor:** Intel Core i3 or equivalent.
  - **Memory:** At least 4 GB of RAM.
  - **Storage:** At least 1 GB of free disk space.
  - **Display:** 1024x768 screen resolution or higher.
  - **Network:** Broadband Internet connection for administrative tasks and system management.

7.2.3 Additional Considerations:

- **GPS:** For devices with mapping features, GPS capability is required for accurate location services.
- **Camera:** A camera with at least 5 MP resolution for uploading pictures of dogs to the system.

These requirements are subject to change based on further development and testing of the Rescue Rovers application. It is also important to note that while the application may run on lower-specification hardware, the experience may not be as intended, and some features may not be available or perform optimally.

CHAPTER 8

8. MODULES

8.1 Use Case Diagram

Figure 3: *Use Case Diagram*

8.1.1 Manage User Accounts (Admin):

- **Description:** The admin manages user accounts within the system. This includes creating, updating, and deactivating accounts for Adoption Centers and Foster Homes.
- **Purpose:** Ensures authorized access and maintains accurate user information.

8.1.2 Resolve Dispute (Admin):

- **Description:** The admin handles disputes that arise between Adoption Centers and Foster Homes. This involves mediating conflicts, investigating issues, and making fair decisions.
- **Purpose:** Maintains a harmonious adoption process and resolves conflicts promptly.

8.1.3 Monitor System Activity (Admin):

- **Description:** The admin monitors system activity, including user interactions, dog adoptions, and communication. This involves tracking logs, analyzing patterns, and identifying anomalies.
- **Purpose:** Ensures system integrity, security, and adherence to policies.

8.1.4 Generate Reports (Admin):

- **Description:** The admin generates reports summarizing adoption activities, communication effectiveness, and overall system performance. These reports inform decision-making and improvements.
- **Purpose:** Provides insights for system optimization and transparency.

8.1.5 Search for Dogs:

- **Description**: Adoption Centers can search the system's dog database based on specific criteria (e.g., breed, age, health status).
- **Purpose**: Helps Adoption Centers find suitable dogs for potential adopters.

8.1.6 View Dog Profiles:

- **Description**: Adoption Centers can access detailed profiles of available dogs. These profiles include information such as breed, temperament, medical history, and contact details of the corresponding Foster Homes.
- **Purpose**: Allows Adoption Centers to evaluate dogs and make informed recommendations to potential adopters.

8.1.7 Initiate Communication:

- **Description**: Adoption Centers can directly communicate with Foster Homes regarding specific dogs. They can inquire about availability, behavior, and any additional details.
- **Purpose**: Facilitates collaboration between Adoption Centers and Foster Homes during the adoption process.

8.1.8 Coordinate Visits:

- **Description**: Adoption Centers coordinate visits with Foster Homes to allow potential adopters to meet the dogs in person. These visits help assess compatibility.
- **Purpose**: Ensures a smooth transition from interest to adoption.

8.1.9 Finalize Adoptions:

- **Description**: Adoption Centers oversee the adoption process. Once a suitable match is found, they finalize the adoption by coordinating paperwork, transfer of ownership, and any necessary fees.
- **Purpose**: Ensures successful adoptions and happy outcomes for both dogs and adopters.

8.1.10 Provide Feedback:

- **Description**: Adoption Centers can provide feedback to the system Admin. This may include reviews of the adoption process, suggestions for improvement, or reporting any issues.
- **Purpose**: Contributes to system enhancements and transparency.

8.1.11 Browse Dogs:

- **Description**: Potential adopters can explore the system's database of available dogs. They can view profiles, photos, and details about each dog.
- **Purpose**: Helps potential adopters find a dog that matches their preferences and lifestyle.

8.1.12 Contact Adoption Centers:

- **Description**: Potential adopters can reach out to Adoption Centers directly. They can inquire about specific dogs, ask questions, and express interest in adoption.
- **Purpose**: Initiates communication and allows potential adopters to gather more information.

8.1.13 Visit Dogs:

- **Description**: Potential adopters can arrange visits with Foster Homes. These visits provide an opportunity to meet the dog in person, assess compatibility, and interact with the dog.
- **Purpose**: Ensures a positive connection between the potential adopter and the dog.

8.1.14 Adopt a Dog:

- **Description**: Once a potential adopter has found the right match, they can proceed with the adoption process. This involves paperwork, fees, and officially welcoming the dog into their home.
- **Purpose**: Bring joy to both the adopter and the dog, creating a loving forever home.

8.1.15 Register Dogs:

- **Description**: Foster Homes can register dogs within the system. This involves adding new dogs to the database and providing details such as breed, age, temperament, and health status.
- **Purpose**: Ensures that available dogs are visible to Adoption Centers and potential adopters.

8.1.16 Update Dog Profiles:

- **Description**: Foster Homes manages dog profiles. They can update information about registered dogs, including any changes in health, behavior, or availability.
- **Purpose**: Keeps dog profiles accurate and up to date for potential adopters.

8.1.17 Arrange Visits:

- **Description**: Foster Homes coordinates visits with potential adopters. These visits allow people to meet the dogs in person, assess compatibility, and ask questions.
- **Purpose**: Facilitates the adoption process by creating a personal connection between dogs and potential adopters.

8.1.18 Transfer Dogs:

- **Description**: When an adoption is finalized, Foster Homes arranges for the transfer of the dog to the new owner. This includes paperwork, medical records, and ensuring a smooth transition.
- **Purpose**: Ensures successful adoptions and a happy transition for the dog.

8.2 Sequence Diagram

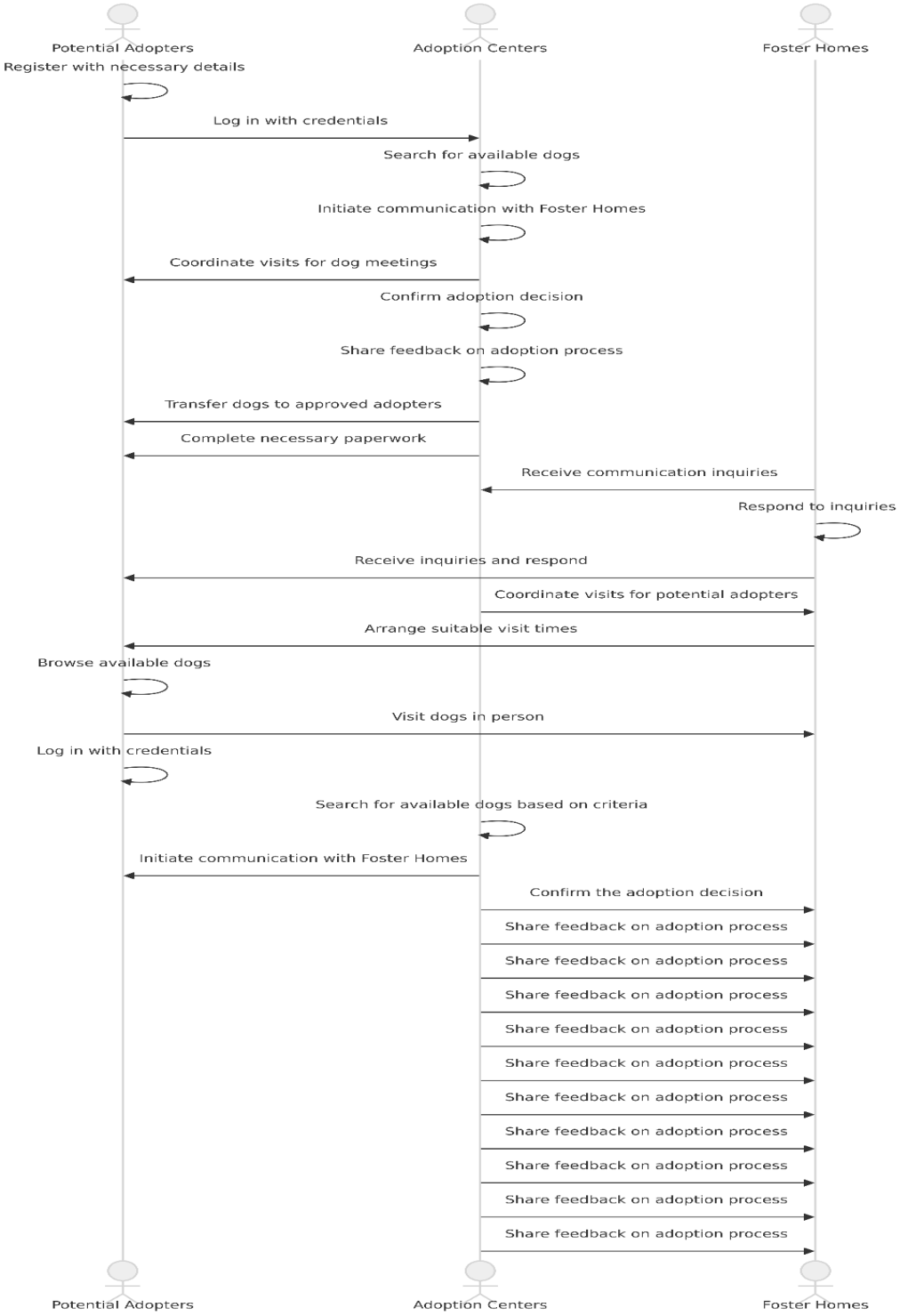

Figure 4: *Sequence Diagram*

Let’s delve into the detailed flow of the Dog Adoption System:

8.2.1 User Registration and Authentication:

- Potential Adopters, Adoption Centers, and Foster Homes register by providing necessary details (such as name, contact information, and preferences).
- The system validates the information and creates user accounts.
- Users log in using their credentials (username and password).

8.2.2 Search for Dogs:

- Adoption Centers search for available dogs based on specific criteria (e.g., breed, age, temperament).
- The system retrieves matching dog profiles, including details like photos, health status, and behavior.
- Potential adopters explore the available dogs, narrowing down their choices.

8.2.3 Contact Foster Homes:

- Adoption Centers initiate communication with Foster Homes regarding specific dogs.
- They inquire about dog availability, behavior, and any additional information.
- Foster Homes promptly respond, providing insights into the dogs’ personalities and care.

8.2.4 Coordinate Visits:

- Adoption Centers coordinate visits for potential adopters to meet dogs in person.
- Foster Homes arrange suitable times for these visits.
- Potential adopters interact with the dogs, assessing compatibility and forming connections.

8.2.5 Finalize Adoptions:

- Once a suitable match is found, Adoption Centers confirm the adoption decision.
- Foster Homes transfer the dog to the approved adopter, ensuring all necessary paperwork is completed.
- The dog officially becomes part of the adopter’s family.

8.2.6 Provide Feedback:

- Adoption Centers share feedback on the adoption process with the System Admin.
- This feedback helps improve the system, making future adoptions even smoother.
- In this intricate dance of compassion, dogs find loving homes, and hearts are filled with joy.

8.3 Activity Diagram

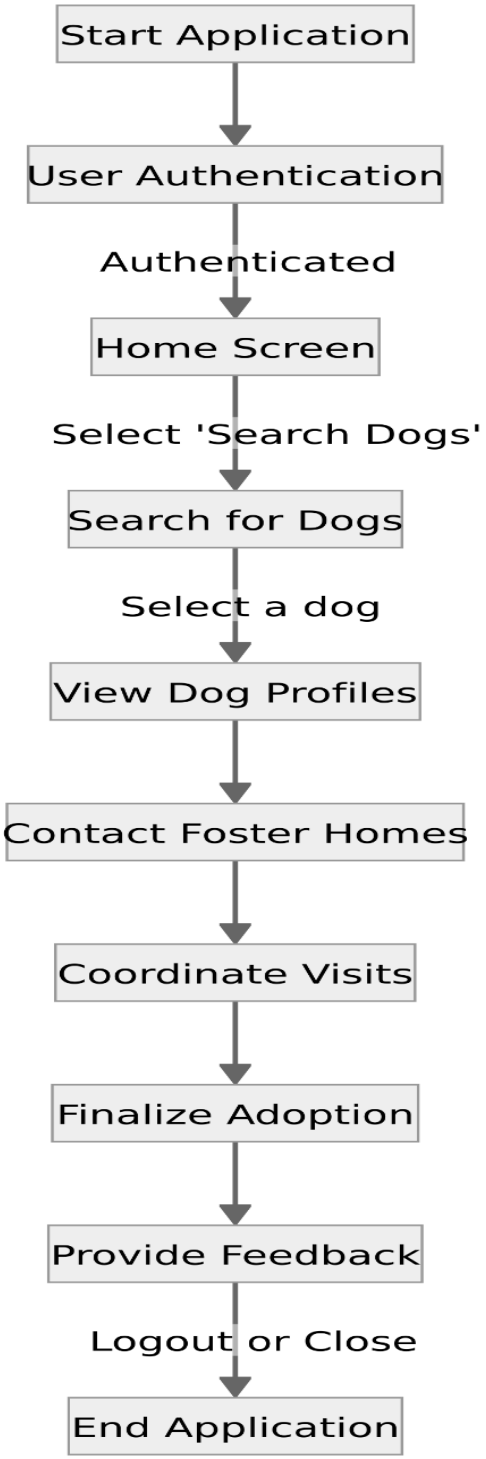

Figure 5: *Activity Diagram*

The activity diagram above outlines the process flow for adopting a dog through Rescue Rovers application. Here's a step-by-step explanation:

- **Start Application**: The user begins by launching the application.
- **User Authentication**: The user must authenticate themselves, likely through a login process.
- **Authenticated**: Once authenticated, the user gains access to the system.
- **Home Screen**: The user is directed to the home screen of the application.
- **Select 'Search Dogs'**: The user selects the option to search for dogs.
- **Search for Dogs**: The user searches the database for available dogs.
- **Select a dog**: From the search results, the user selects a dog to learn more about.
- **View Dog Profiles**: The user views the profile of the selected dog, which includes detailed information.
- **Contact Foster Homes**: The user contacts the foster home to inquire about the dog.
- **Coordinate Visits**: The user arranges a visit to meet the dog in person.
- **Arrange Adoption**: If the user decides to adopt, they proceed with arranging the adoption.
- **Final Adoption**: The adoption is finalized, and the dog is transferred to the new owner.

This diagram serves as a visual guide for the sequential steps involved in the dog adoption process within our application.

CHAPTER 9

9. IMPLEMENTATION

9.1 Startup Screen

The startup screen of the application is designed to be both informative and visually appealing. Here's a description of what it might include:

- **Carousels**:
    - **First Carousel**: This could showcase happy dogs with their new families, highlighting successful adoptions facilitated by the app.
    - **Second Carousel:** This might feature testimonials from satisfied adopters or foster homes, providing social proof of the app's impact.
    - **Third Carousel:** The final carousel could display the various features of the app, like searching for dogs, viewing profiles, and contacting foster homes.

Each carousel would be visually striking, with high-quality images and engaging captions that succinctly convey the app's purpose and success stories.

- **Skip Button:**

    - Located at the bottom or corner of the screen, the skip button allows users to bypass the carousels and directly enter the main part of the app if they prefer not to view the introductory slides.

This startup screen serves as an engaging introduction to the app, giving users a glimpse into its functionality and the joy it brings to dogs and adopters alike.

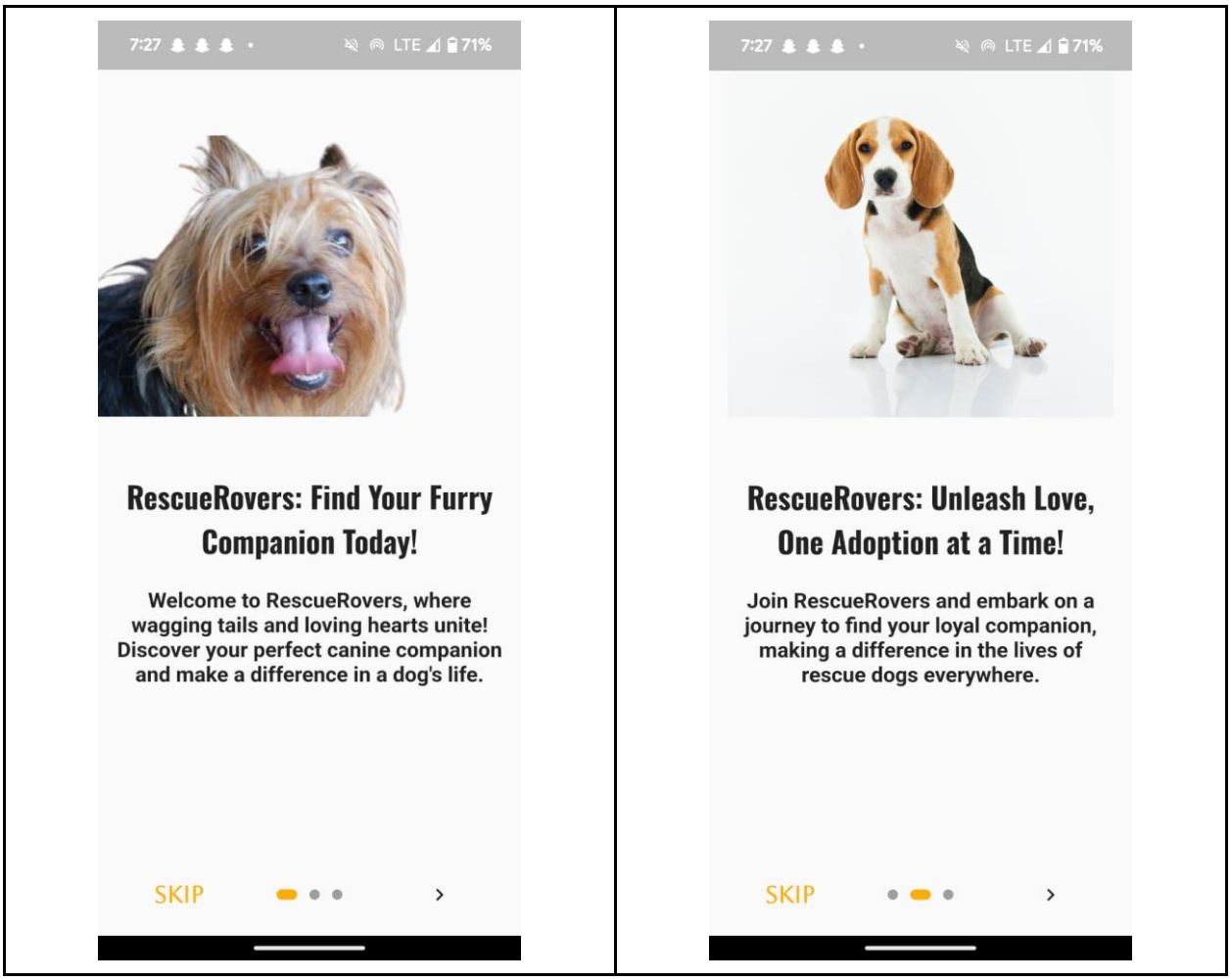

Figure 6: *Startup Screens*

9.2 Sign Up Page:

The sign-up screen serves as the entry point for adoption centers and foster homes to create accounts within our dog adoption system. By registering, users gain access to features tailored to their specific requirements.

9.2.1 Components

- **Name**: Adoption centers or foster homes enter their organization's name (e.g., "Xyz Adoption Center").
- **Email Address**: Users provide their official email address (e.g., "xyzadoption@xyz.com").
- **Password**: A secure password is created for account access.
- **User Type Selection**: A dropdown menu allows users to choose their role:

- **Adoption Center**: For organizations facilitating dog adoptions.
- **Other user types** (e.g., "Foster Home," "Admin") may also be available.
- **Sign-Up Button**: A prominent "Sign up" button enables users to submit their information and create an account.
- **Existing Account Option**: Users with existing accounts can log in directly via the "Already have an account? Login" link.

This streamlined sign-up process ensures that adoption centers and foster homes can seamlessly engage with our system.

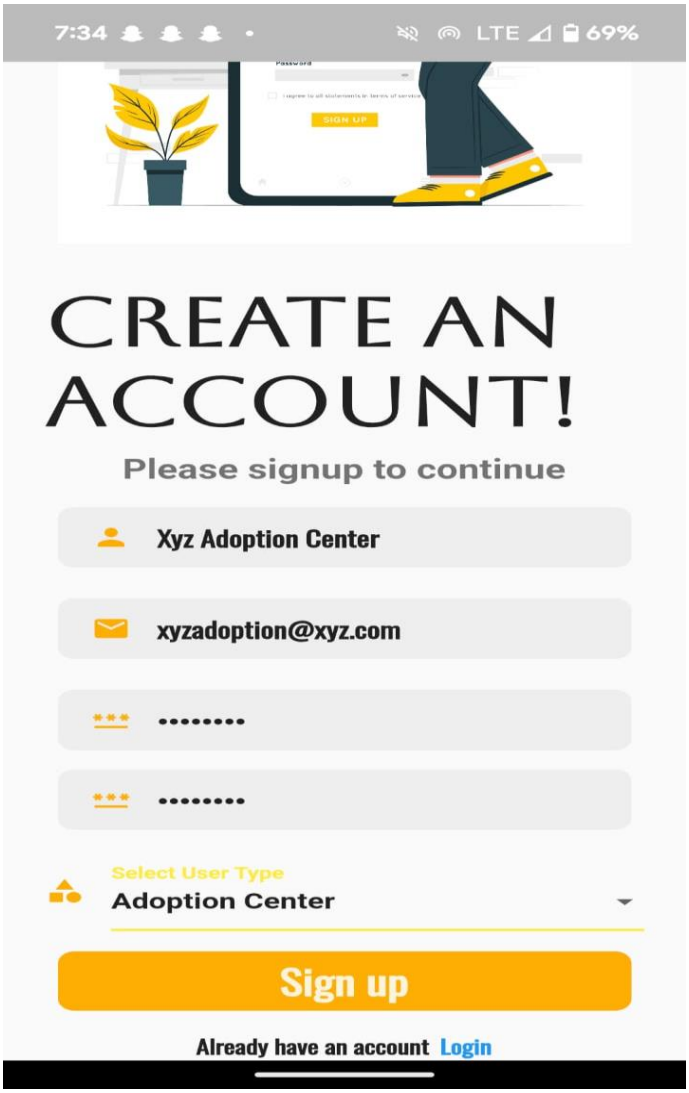

Figure 7: *Sign Up Screen*

9.3 Login Screen:

The login screen serves as the gateway for authorized users (such as adoption centers and foster homes) to access the dog adoption system.

Its primary objectives are:

- **Authentication**: To verify the user's identity by validating their credentials (email and password).
- **Access Control**: To ensure that only registered users can enter the system.
- **Seamless Experience**: To provide a user-friendly interface for logging in, minimizing friction during the sign-in process.
- **Security**: To protect sensitive information and maintain the integrity of user accounts.

Let's discuss the login screen:

- **Title and Introduction**: The screen begins with a welcoming message: "WELCOME BACK!" Below that, a secondary line reinforces the purpose: "Please Sign in to continue."
- **Input Fields**: Email Address: Users enter their registered email address (e.g., "xyzadoption@xyz.com").
- **Password**: A secure password is required for account access.
- **Login Button**: A prominent orange button labeled "Login" allows users to submit their credentials.
- **Forgot Password Option**: For users who forget their password, there's an option: "Forgot password?" Clicking this link would likely lead to a password reset process.
- **Account Creation Option**: At the bottom of the screen, a link reads: "Doesn't have an account? Create One." New users can click this link to register and create an account.

This concise and user-friendly login screen ensures a smooth sign-in experience for adoption centers and foster homes

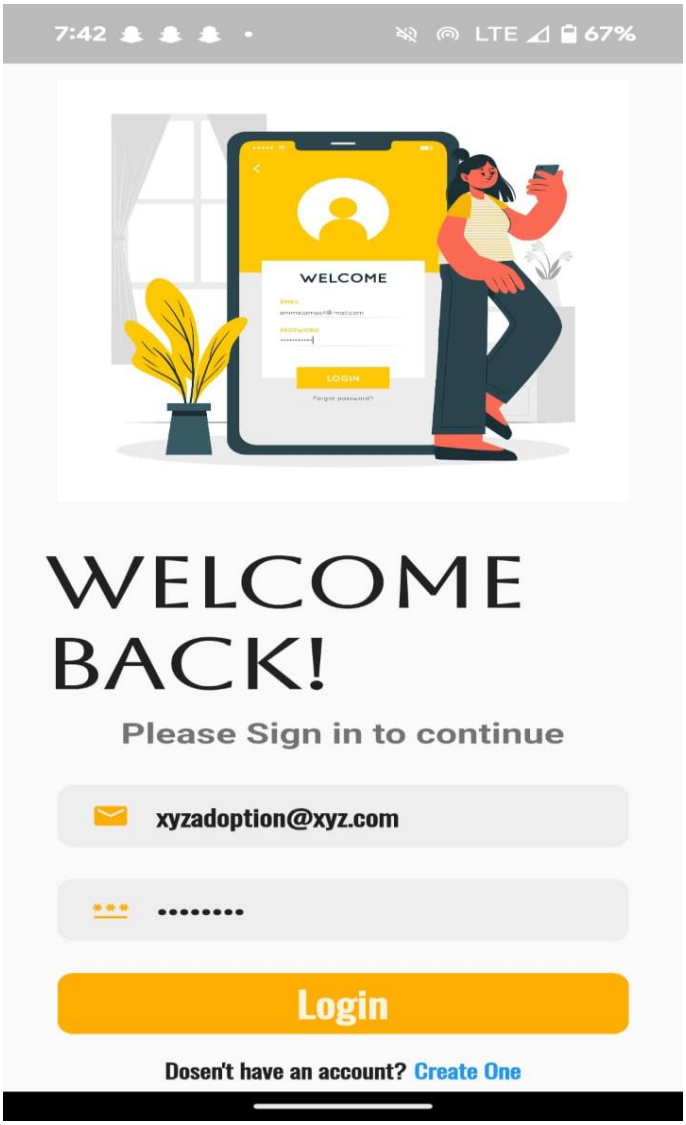

Figure 8: *Login Screen*

9.4 Adoption Center View:

9.4.1 Homepage:

9.4.1.1 Homepage Overview:

- **Title**: The homepage prominently features the "RESCUEROVERS" logo, reinforcing brand identity.
- **Welcome Message**: A message encourages users to engage with the app's purpose: "WE WISH YOU TO MAKE A NEW FRIEND."
- **Search Functionality**: A search bar labeled "Search for a dog" allows users to quickly find

specific dogs or browse the database.

9.4.1.2 Dog Postings Section

- **Section Title**: “New Pets” introduces the area where new dog postings from foster homes are displayed.
- **Postings Layout**: Each posting includes:
    - A photo of the dog, providing a visual representation.
    - The dog’s name, age, and location, offering essential information at a glance.
    - An interactive heart icon, enabling users to like or favorite the postings.

9.4.1.3 Navigation Bar

- **Functionality**: The bottom navigation bar includes icons for Home, Favorites, Locations, and User Profile, facilitating easy movement between different sections of the app.

9.4.1.4 Purpose of the Homepage

- **Adoption Center View**: Tailored for adoption centers, the homepage streamlines the process of finding and favoriting dogs for potential adoption.
- **Foster Home Postings**: Showcases dogs available for adoption, highlighting key details to assist adoption centers in making informed decisions.

This homepage serves as the central hub for adoption centers to browse, search, and express interest in dogs posted by foster homes, ultimately aiding in the adoption process.

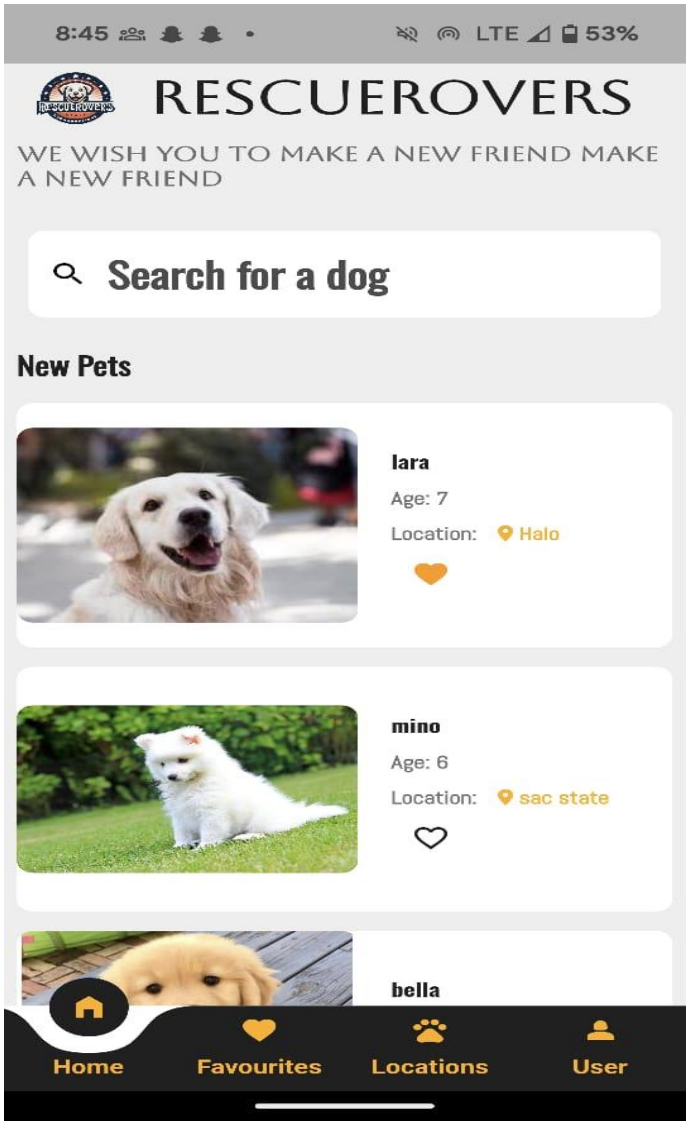

Figure 9: *Adoption Center View*

9.4.2 Dog Posting:

9.4.2.1 *Overview*

- **Navigation Bar**: Includes a back arrow and the dog's name, "LARA," indicating the current page.
- **Dog Photo**: A prominent photo of Lara, likely a Golden Retriever or similar breed, captures the attention and provides a visual introduction to the dog.

9.4.2.2 Dog Information

- **Name**: Displayed in large capital letters for easy identification.
- **Age**: “7 years” indicates Lara’s age, an important factor for potential adopters.
- **Breed**: Listed as “idk,” suggesting uncertainty about her exact breed.
- **Description**: A brief note describes Lara as “a female beautiful dog,” providing a quick insight into her appearance and gender.

9.4.2.3 Contact Details

- **Location**: Marked with a location symbol followed by “HALO,” possibly the name of the foster home or shelter.
- **Phone Number**: “637535852829” is provided for direct contact.

9.4.2.4 Adoption Actions

- **Adopt Now Button**: An orange button invites users to take immediate action if they wish to adopt Lara.
- **Contact on WhatsApp**: A green button offers an alternative method of communication, indicating the possibility of instant messaging for inquiries.

This posting is designed to give potential adopters all the necessary information at a glance, with clear calls to action for further engagement.

Figure 10: *Dog Posting Screen*

9.4.3 Favorites:

This section includes all the dog postings that are marked as favorites by the adoption center so that they can come later and view them or open them.

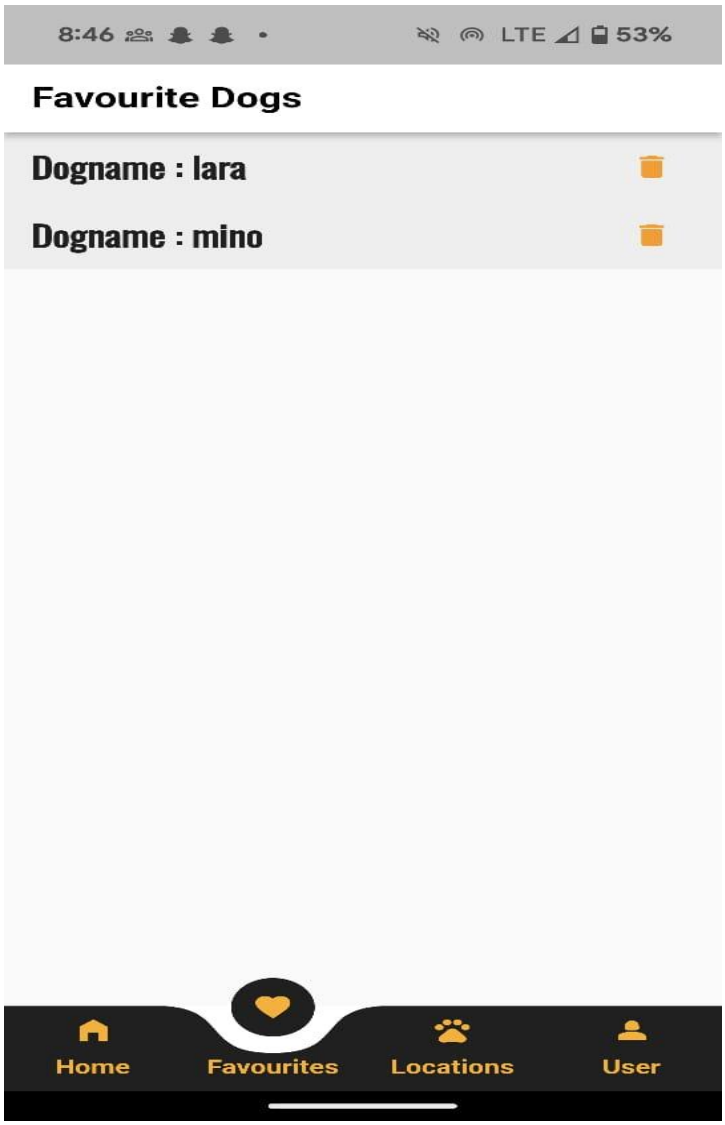

Figure 11: *Favorites Screen*

9.4.4 Locations:

This section indicates the nearby foster homes within the 50km range. It is done to make visits easier. The Google Map API has been integrated to calculate the estimated distance. The names and contact details of Foster Homes are visible. The adoption centers can contact and visit Foster Homes to see the available dogs for adoption.

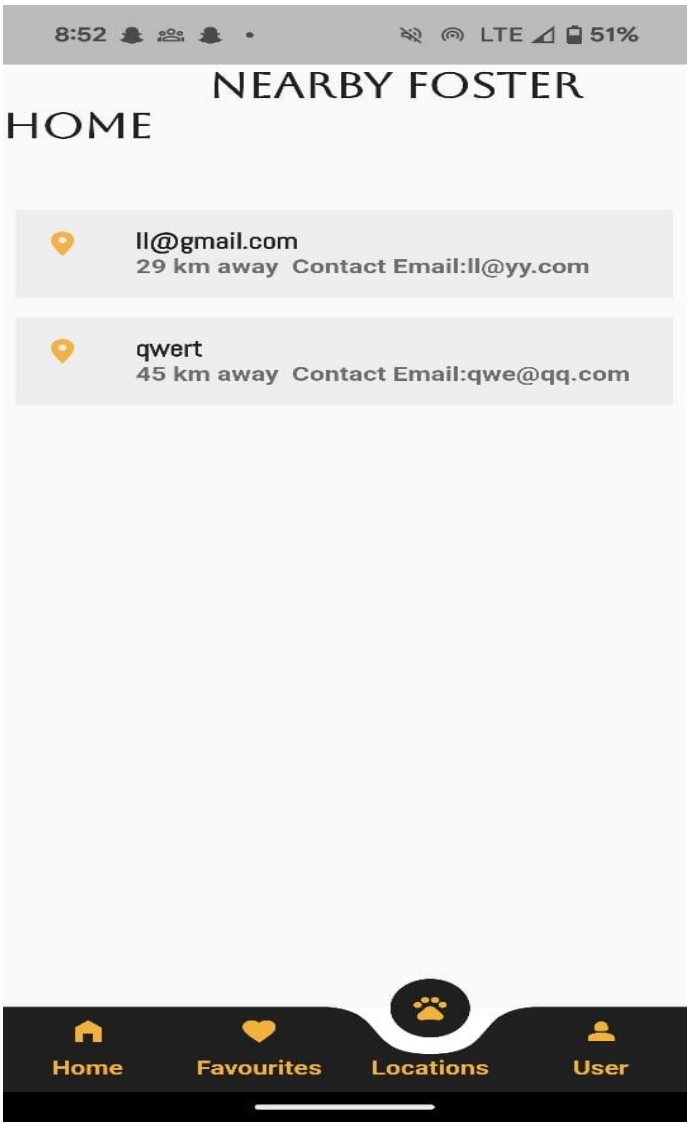

Figure 12: *Location Screen*

9.5 Foster Home View:

9.5.1 Homepage:

This page provides foster homes with an overview of the dogs they have posted for adoption:

9.5.1.1 Title and Logo:

- The top left corner displays the logo of "RESCUEROVERS," reinforcing brand identity.
- The title "DOG PROFILE" indicates the purpose of this page.

9.5.1.2 Dog Profile Display:

- **Dog Name**: The profile showcases a specific dog named Kelly.
- **Dog Photo**: A photo of Kelly, a black and fluffy dog, provides a visual introduction.
- **Age**: Kelly's age is listed as 5 years.
- **Location**: The location is marked with a pin icon and labeled as "New York".

9.5.1.3 Action Options:

- **Create A Dog Profile**: Below Kelly's profile, there's an option for foster homes to create profiles for additional dogs. This encourages foster homes to actively participate in the adoption process.
- **Navigation Bar:** At the bottom of the interface, there are navigation options:
    - **Home**: Returns users to the homepage.
    - **Adoption Center**: This likely leads to a broader adoption center view.
    - **User**: Provides access to user-specific settings or account management.

9.5.1.4 Purpose of the Homepage

- **Foster Home Perspective**: Tailored for foster homes, this homepage allows them to manage and showcase dogs available for adoption.
- **Efficient Access**: Foster homes can easily review their posted dogs, update profiles, and engage with potential adopters.

By providing a clear overview of posted dogs, this homepage streamlines the adoption process and encourages foster homes to actively participate in finding loving homes for their furry companions.

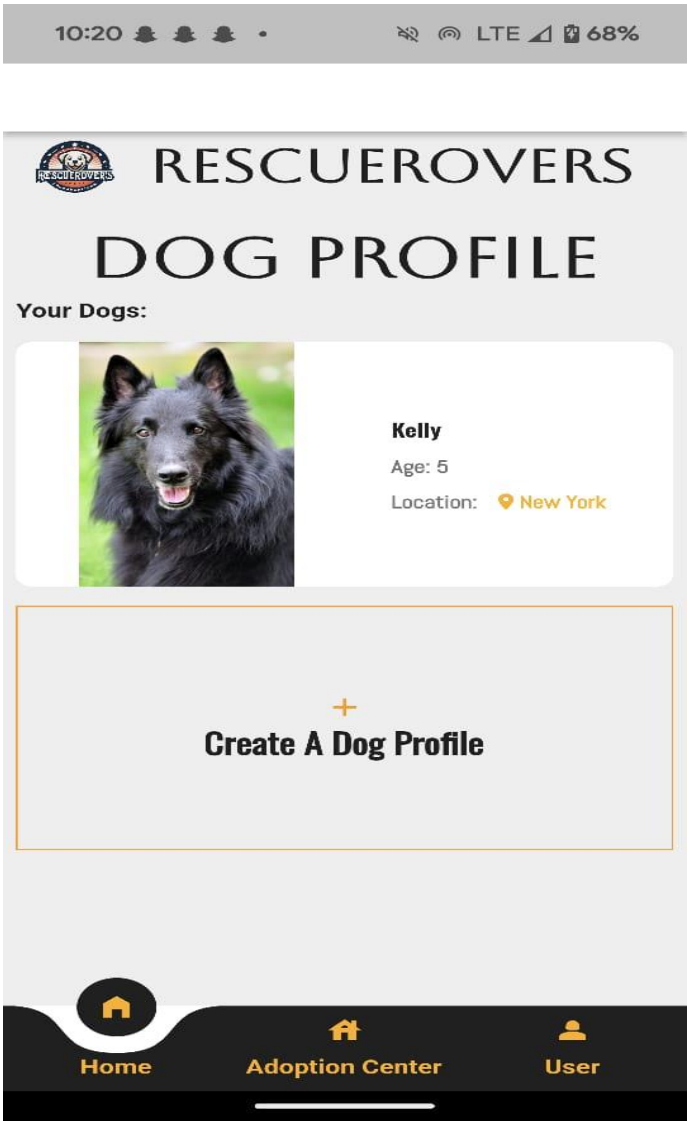

Figure 13: *Foster Home View*

9.5.2 Create Dog Profile:

The purpose of this form is to streamline the process of adding dogs to the adoption system, making it easier for foster homes to showcase their available pets.

The form contains the following fields:

- **Dog Name**: Users can input the name of the dog they want to create a profile for. For example, “Buddy” or “Luna.”
- **Dog Age**: This field allows users to specify the age of the dog. They can enter the dog’s

age in years or months (e.g., “3 years” or “6 months”).

- **Dog Breed**: Users provide information about the dog’s breed. They can type the breed’s name (e.g., “Labrador Retriever” or “Mixed Breed”).
- **Foster Home’s City**: Users enter the city where the foster home is located. For instance, **“New York” or “Los Angeles.”**
- **Description About Dog**: In this section, users can write a brief description of the dog. They might mention the dog’s personality, behavior, or any special characteristics (e.g., “Friendly, loves belly rubs”).
- **Foster Home’s Phone Number**: Users input the phone number associated with the foster home. This allows potential adopters to contact the foster home for inquiries.
- **Upload Dog Image**: Users have the option to upload a photo of the dog. This image will be displayed alongside the profile.
- **Create Profile Button**: The orange “Create Profile” button at the bottom allows users to submit the information and create the dog’s profile.

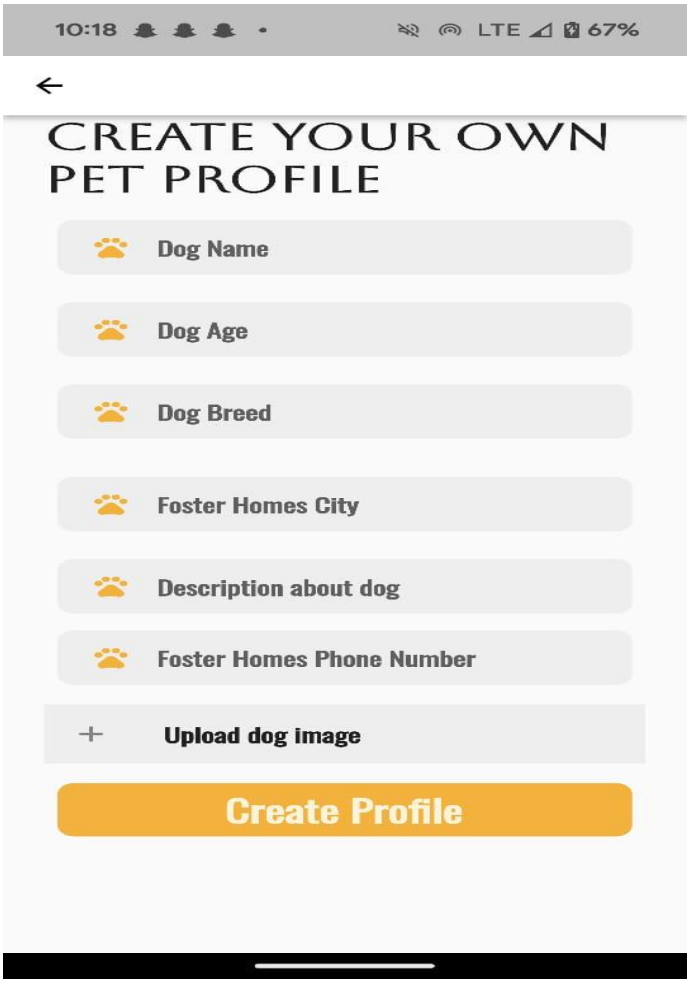

Figure 14: *Create Dog Profile Screen*

9.5.3 Edit Dog Profile:

This form is intended for Foster Homes to edit the dog profiles and rectify if there are any mistakes. Foster Homes can also delete the Dog Profile by clicking on the **Delete Dog Profile Button.**

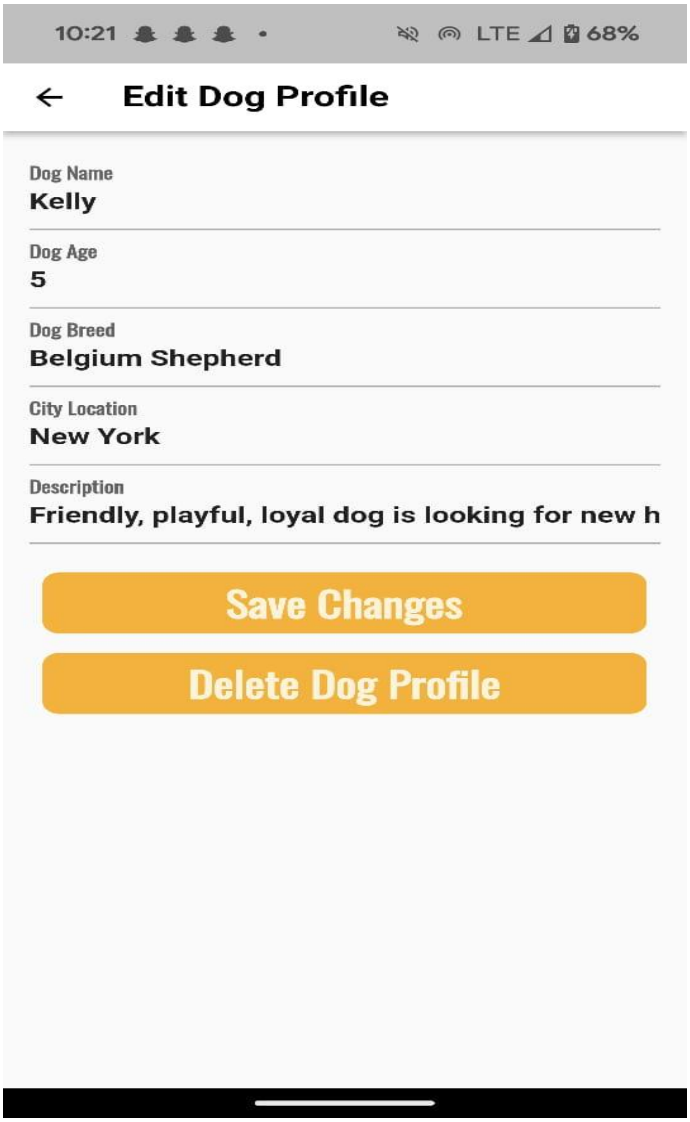

Figure 15: *Edit Dog Profile Screen*

9.5.4 Nearby Adoption Centers:

This section indicates the nearby Adoption Centers within the 50km range. It is done to make visits easier. The Google Map API has been integrated to calculate the estimated distance. The names and contact details of Adoption Centers are visible. The Foster Homes can contact and visit the Adoption Centers to make the adoption process smoother.

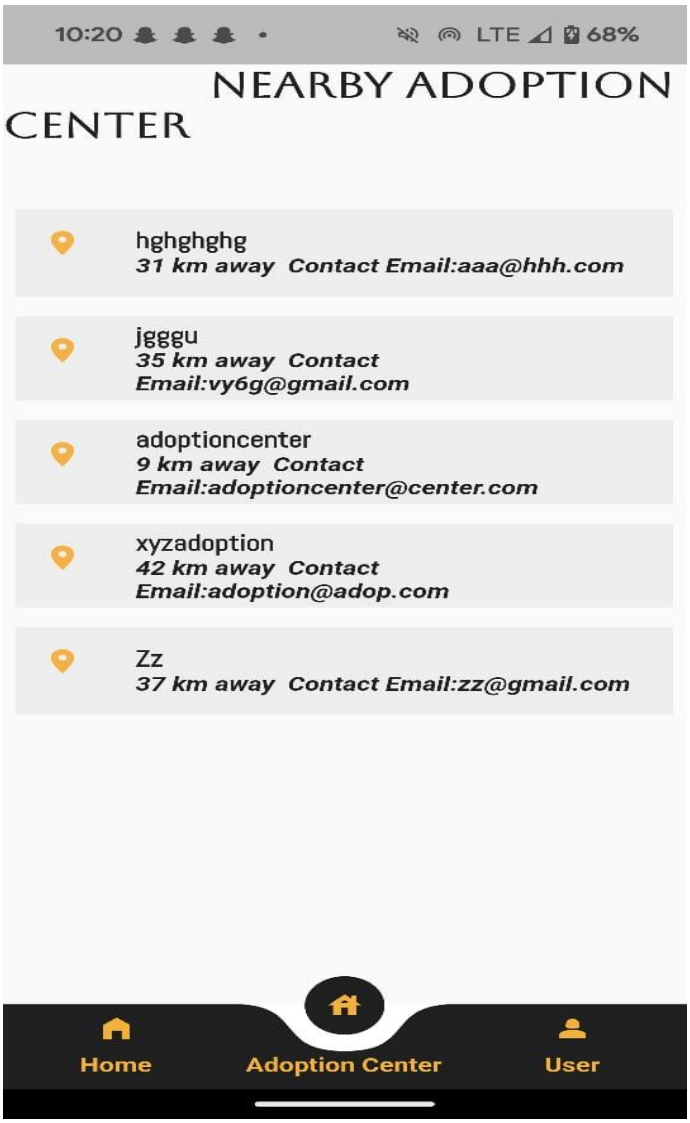

Figure 16: *Nearby Adoption Center Screen*

CHAPTER 10

10. CONCLUSION AND FUTURE WORK:

10.1 Conclusion:

In conclusion, the Rescue Rovers Dog Adoption System represents a significant stride forward in the mission to connect dogs in need with loving homes. Throughout this document, we have meticulously outlined the system's architecture, functionalities, and user interfaces, all designed with the goal of streamlining the adoption process for all parties involved.

The system's architecture is robust and flexible, ensuring that it can handle the dynamic nature of dog adoptions. With a secure database, intuitive user management, and comprehensive dog profiles, Rescue Rovers provides a reliable platform for adoption centers and foster homes to collaborate effectively. The communication tools embedded within the system facilitate seamless interactions, allowing for quick responses and real-time updates, which are crucial in the fast-paced environment of pet adoptions.

From the perspective of adoption centers, the system offers a powerful search functionality that enables them to find suitable dogs for potential adopters efficiently. The ability to view detailed dog profiles, initiate communication with foster homes, and coordinate visits are all integral components that enhance the adoption experience. Moreover, the system's feedback mechanism allows adoption centers to provide valuable insights that can be used to further refine and improve the platform.

For foster homes, the system provides an equally beneficial experience. The ability to register dogs, update their profiles, and manage inquiries from adoption centers empowers foster homes to advocate for the dogs in their care effectively. The system's design ensures that foster homes can easily navigate through the various functionalities, from arranging visits to transferring dogs to their new families.

Potential adopters are not left out of the equation. The system's user-friendly interface allows them to browse available dogs, contact adoption centers, and arrange visits, all of which are steps toward finding their new furry family member. The system's design ensures that potential adopters have a positive experience, which is essential in encouraging more people to consider adoption.

The document has also provided detailed descriptions of various screens within the system, such as the sign-up screen, login screen, homepage views for both adoption centers and foster homes, and the create dog profile form. Each of these screens has been designed with the user in mind, ensuring that the system is accessible, easy to use, and visually appealing.

As we look to the future, the Rescue Rovers Dog Adoption System stands as a testament to the power of technology in making a positive impact on society. By leveraging the capabilities of this

system, we can increase the efficiency of dog adoptions, reduce the time dogs spend in foster care, and ultimately ensure that more dogs find their way into loving homes.

The success of this system hinges on the collaboration between adoption centers, foster homes, and potential adopters. It is a collective effort that requires the dedication and commitment of all stakeholders. With the Rescue Rovers Dog Adoption System, we have laid the groundwork for a brighter future for dogs in need, and we look forward to seeing the joy and companionship that each successful adoption brings.

In closing, the Rescue Rovers Dog Adoption System is more than just a technological solution; it is a beacon of hope for dogs awaiting adoption and a tool for those who work tirelessly to make those adoptions possible. As we move forward, we will continue to listen, learn, and adapt, ensuring that the system evolves to meet the ever-changing needs of the dog adoption community. Together, we can make a difference, one adoption at a time.

10.2 Future Work:

As we look ahead, the Rescue Rovers Dog Adoption System is poised for growth and innovation. The system has laid a strong foundation, but the journey towards perfection is continuous. In this section, we outline the future work that will enhance the system's capabilities, improve user experience, and expand its reach.

10.2.1 Integration with External Services

To further streamline the adoption process, integration with external services such as veterinary databases, microchip registries, and national pet lost-and-found networks will be considered. This will allow for real-time updates on medical records, vaccination statuses, and microchip information, providing a comprehensive view of each dog's health and identification details.

10.2.2 Advanced Search and Matching Algorithms

Developing more sophisticated search and matching algorithms is a priority. By leveraging machine learning and artificial intelligence, the system can analyze user preferences, dog characteristics, and past successful adoptions to suggest optimal matches between dogs and potential adopters.

10.2.3 Mobile Application Development

While the current system is mobile-responsive, a dedicated mobile application for both Android and iOS platforms will be developed. This app will offer push notifications, location-based services, and offline capabilities, making the adoption process more accessible and convenient for users on the go.

10.2.4 User Experience Enhancements

User feedback will be instrumental in refining the system's interface and functionalities. Future updates will focus on creating a more personalized experience, with customizable dashboards, advanced filtering options, and interactive elements that engage users more deeply with the platform.

10.2.5 Foster Home Support and Training

Recognizing the critical role of foster homes in the adoption process, we plan to develop a suite of tools and resources to support them. This includes training modules on dog care, behavior management, and adoption counseling, as well as forums for foster homes to share experiences and advice.

10.2.6 Adoption Center Collaboration

Efforts will be made to foster stronger collaborations with adoption centers. This includes sharing best practices, coordinating adoption events, and developing joint marketing campaigns to raise awareness about the importance of dog adoption.

10.2.7 Community Building

Building a community around the Rescue Rovers brand will be a key focus. This includes creating social media groups, organizing meetups, and hosting webinars that bring together adopters, foster homes, and adoption centers to celebrate successes and share stories.

10.2.8 Reporting and Analytics

Enhanced reporting and analytics features will be added to provide insights into adoption trends, system usage, and user behavior. These insights will inform decision-making and help identify areas for improvement.

10.2.9 International Expansion

The system will explore opportunities for international expansion, adapting to different legal and cultural environments. This will involve localizing the platform, establishing partnerships with international rescue organizations, and ensuring compliance with local regulations.

10.2.10 Sustainability Initiatives

Sustainability initiatives will be introduced, such as promoting eco-friendly pet products and

practices within the community. The system will also explore ways to offset its carbon footprint and support environmental conservation efforts.

10.2.11 Research and Development

A dedicated research and development team will be established to stay abreast of technological advancements and explore new features that could benefit the system. This includes virtual reality dog visits, blockchain for secure record-keeping, and voice-activated commands for accessibility.

10.2.12 Financial Sustainability

To ensure the long-term sustainability of the system, various revenue models will be explored. This includes premium features for users, partnerships with pet-related businesses, and grant opportunities for technological innovation in animal welfare.

10.2.13 Continuous Improvement

Above all, the system will commit to a philosophy of continuous improvement. Regular updates, user surveys, and beta testing of new features will be standard practices to ensure that the Rescue Rovers Dog Adoption System remains at the forefront of facilitating dog adoptions.

In conclusion, the future of the Rescue Rovers Dog Adoption System is bright and full of potential. With a commitment to innovation, user satisfaction, and the welfare of dogs, the system will continue to evolve and make a positive impact on the world of dog adoption.

11. REFERENCES:

- American Veterinary Medical Association. (2018). AVMA Guidelines for the Euthanasia of Animals. Retrieved September 27, 2021, from https://www.avma.org/resources-tools/avma-policies/euthanasia-animals
- American Society for the Prevention of Cruelty to Animals (ASPCA). (2021). Pet Statistics. Retrieved September 27, 2021, from https://www.aspca.org/animal-homelessness/shelter-intake-and-surrender/pet-statistics
- Carlisle-Frank, P., & Frank, J. M. (2019). The dilemma of dog intake management in the United States: A case for evidence-based decision-making. Frontiers in veterinary science, 6, 180.
- Gazzano, A., Zilocchi, M., Massoni, E., & Mariti, C. (2020). Effects of rescue-dog training on behavioral and physiological parameters. Journal of Veterinary Behavior, 35, 19-26.
- Hsu, Y. (2018). The effectiveness of animal-assisted intervention in managing anxiety and depression of institutionalized elderly. International Journal of Geriatric Psychiatry, 33(11), e163-e171.
- Save a Dog, Save a Cat. (2021). Adoption Statistics. Retrieved September 27, 2021, from https://saveadogsaveacat.org/adoption-statistics/
- Vladez, F., Armitage-Chan, E., & McConnell, F. (2019). Animal welfare and ethics in veterinary practice: A survey of North American veterinarians. Journal of veterinary medical education, 46(4), 465-476.
- Arhant, C., & Aurich, J. (2016). Rescue dogs: Health, welfare, and selection. Journal of Veterinary Behavior, 14, 30-38.
- Weiss, E., Mohan-Gibbons, H., & Slater, M. (2016). The impact of shelter housing on dog welfare: A critical review. Journal of Applied Animal Welfare Science, 19(3), 236-253.
- Hetzler, J. J., & Patronek, G. J. (2013). The use of social media in animal advocacy. Journal of Applied Animal Welfare Science, 16(4), 338-346.
- Overall, K. L. (2017). Preventing dog bites: The need for a coordinated community response. Journal of Veterinary Behavior, 19, 7-9.
- Reisner, I. R. (2016). Socialization of puppies and adult dogs. Veterinary Clinics of North America: Small Animal Practice, 46(5), 915-928
- Rescue Rovers organization, from https://www.rescuerovers.org/
- Flutter from https://docs.flutter.dev/ui
- Firebase NoSQL database from, https://firebase.google.com/products/firestore#:~:text=Cloud%20Firestore%20is%20a%20NoSQL,web%20apps%20%2D%20at%20global%20scale.
- Dart from https://dart.dev/guides
- Android Studio from https://developer.android.com/develop

12. APPENDICES:

12.1 Installation Instructions

- **Clone the Repository:**

  - **Step 1:** Open a terminal or command prompt on your machine.
  - **Step 2:** Navigate to the directory where you want to clone the repository.
  - **Step 3:** Run the following command to clone the repository:

```
git clone https://github.com/DharmeshTewari/RescueRovers-Mobile-Application_FinalYearProject_Flutter.git
```

- **Install Flutter:**

```
A few resources to get you started if this is your first Flutter project:

- [Lab: Write your first Flutter app](https://docs.flutter.dev/get-started/codelab)
- [Cookbook: Useful Flutter samples](https://docs.flutter.dev/cookbook)

For help getting started with Flutter development, view the
[online documentation](https://docs.flutter.dev/), which offers tutorials,
samples, guidance on mobile development, and a full API reference.
```

- **Download and Install Android Studio:**

```
## Download and Install Android Studio IDE

https://developer.android.com/studio?gad_source=1&gclid=CjwKCAjwrvyxBhAbEiwAEg_KgsB3MDFD5oWxhOvVh4V-8CoDNfflYz5YOJ3CSTnEGaJkF_MWJr_wsRoC5PYQAvD_BwE&gclsrc=aw.ds
```

- **Add/Configure mobile Device in android studio:**

```
## Add/Configure mobile Device in android studio

Go to Device Manager and configure the mobile device
```

- **Create account in Firebase:**

```
## Create account in Firebase

https://firebase.google.com/docs/database
```

12.2 Programming of an application

12.2.1 Main.dart

```
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:rescuerovers/Login&Signup/main_page.dart';
import 'package:rescuerovers/Onboarding/on_boarding_screen.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:firebase_core/firebase_core.dart';
import 'Login&Signup/Register.dart';
import "Login&Signup/Login.dart";

bool show = true;
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  final prefs = await SharedPreferences.getInstance();
  show = prefs.getBool('ON_BOARDING') ?? true;
  WidgetsFlutterBinding.ensureInitialized();
  Platform.isAndroid
      ? await Firebase.initializeApp(
          options: FirebaseOptions(
              apiKey: "AIzaSyBi4Bj7JRpv4kKRABvCixRlPjEJhP256R8",
              appId: "1:633754370973:android:e0644850ac8f8c84ade574",
              messagingSenderId: "633754370973",
              projectId: "dogadopt-e284a",
              storageBucket: "dogadopt-e284a.appspot.com"),
        )
      : Firebase.initializeApp();
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        debugShowCheckedModeBanner: false,
        title: 'RescueRovers',
        theme: ThemeData(primarySwatch: Colors.yellow),
        // home: show ? OnBoardingScreen() : RegisterPage(),
        home: show ? OnBoardingScreen() : MainPage());
  }
}
```

12.2.2 Home Page Code:

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:google_fonts/google_fonts.dart';
```

```dart
import 'package:rescuerovers/DogsProfiles.dart';
import 'package:rescuerovers/deatilsDog.dart';
import 'package:rescuerovers/petslistview.dart';
import 'package:shared_preferences/shared_preferences.dart';

class HomePage extends StatefulWidget {
  HomePage({Key? key}) : super(key: key);

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  late Stream<QuerySnapshot> _namesStream;
  late List<DocumentSnapshot> _searchResults = [];

  @override
  void initState() {
    super.initState();
    _namesStream =
        FirebaseFirestore.instance.collection('dogsprofiles').snapshots();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[200],
      body: SafeArea(
        child: SingleChildScrollView(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            children: [
              ListTile(
                leading: Image.asset("assets/logo.png"),
                title: Text(
                  "RescueRovers",
                  style: GoogleFonts.aboreto(
                      fontSize: 33, fontWeight: FontWeight.bold),
                ),
              ),
              Padding(
                padding: const EdgeInsets.all(8.0),
                child: Text(
                  '''WE wish for you to make a New Friend''',
                  style: GoogleFonts.aboreto(
                      fontWeight: FontWeight.bold,
                      fontSize: 15,
                      color: Colors.black54),
                ),
              ),
              SizedBox(height: 10),
              Padding(
                padding: const EdgeInsets.all(15.0),
                child: Container(
                  width: MediaQuery.of(context).size.width,
                  decoration: BoxDecoration(
                    borderRadius: BorderRadius.circular(10),
```

```
                    color: Colors.white,
                    boxShadow: [],
                  ),
                  child: TextField(
                    onChanged: (value) {
                      _searchDog(value);
                    },
                    style: TextStyle(color: Colors.black),
                    decoration: InputDecoration(
                      hintText: 'Search for a dog',
                      hintStyle: GoogleFonts.oswald(
                        color: Colors.black.withOpacity(0.7),
                        fontSize: 25,
                      ),
                      prefixIcon: Icon(Icons.search, color: Colors.black),
                      border: InputBorder.none,
                      contentPadding:
                          EdgeInsets.symmetric(horizontal: 20, vertical: 14),
                    ),
                  ),
                ),
              ),
              Padding(
                padding: const EdgeInsets.all(8.0),
                child: Row(
                  mainAxisAlignment: MainAxisAlignment.start,
                  children: [
                    Text(
                      "New Pets",
                      style: GoogleFonts.oswald(fontSize: 18),
                    ),
                  ],
                ),
              ),
              Container(
                child: StreamBuilder(
                  stream: _namesStream,
                  builder: (context, AsyncSnapshot<QuerySnapshot> snapshot) {
                    if (snapshot.connectionState == ConnectionState.waiting)
{
                      return Center(child: CircularProgressIndicator());
                    }
                    if (snapshot.hasError) {
                      return Center(child: Text('Error: ${snapshot.error}'));
                    }
                    final List<DocumentSnapshot<Object?>> documents =
                        _searchResults.isEmpty
                            ? snapshot.data!.docs
                            : _searchResults;
                    if (documents.isEmpty) {
                      return Center(child: Text('No dogs found'));
                    }
                    return ListView.builder(
                      shrinkWrap: true,
                      physics: NeverScrollableScrollPhysics(),
                      itemCount: documents.length,
                      itemBuilder: (context, index) {
```

```
                        final doc = documents[index];
                        return GestureDetector(
                          onTap: () {
                            Navigator.push(
                              context,
                              MaterialPageRoute(
                                builder: (context) => DogDetails(
                                  Fosterhomesemail: doc["Fosterhomesemail"],
                                  dogAge: doc["dogage"],
                                  dogdiscription: doc["dogdescription"],
                                  dogBreed: doc["dogbreed"],
                                  dogName: doc["dogname"],
                                  dogCityLocation: doc["dogcitylocation"],
                                  imgUrl: doc["downloadUrl"],
                                  fosterhomesphonenumber:
                                      doc["FosterHomesPhoneNumber"],
                                ),
                              ),
                            );
                          },
                          child: DogListView(
                            Dogname: doc['dogname'],
                            Dogage: doc['dogage'],
                            Dogbreed: doc['dogbreed'],
                            dogcitycloaction: doc['dogcitylocation'],
                            imgurl: doc['downloadUrl'],
                          ),
                        );
                      },
                    );
                  },
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }

  void _searchDog(String query) {
    setState(() {
      _searchResults = [];
    });

    if (query.isNotEmpty) {
      FirebaseFirestore.instance
          .collection('dogsprofiles')
          .where('dogname', isEqualTo: query)
          .get()
          .then((snapshot) {
        setState(() {
          _searchResults = snapshot.docs;
        });
      });
    }
```

```
  }
}
```

12.2.3 Register Page Code:

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:google_fonts/google_fonts.dart';
import 'ReqWidgets.dart';

class RegisterPage extends StatefulWidget {
  RegisterPage({super.key, required this.showloginpage});
  final VoidCallback showloginpage;
  @override
  State<RegisterPage> createState() => _RegisterPageState();
}

class _RegisterPageState extends State<RegisterPage> {
  final emailcontroller = TextEditingController();
  final passcontroller = TextEditingController();
  final confirmpasscontroller = TextEditingController();
  final namecontroller = TextEditingController();

  String? _userType; // To store the selected user type

  Future signup() async {
    // Validate the form
    if (!_formKey.currentState!.validate()) {
      return;
    }

    try {
      if (passconfirmed()) {
        UserCredential userCredential = await FirebaseAuth.instance
            .createUserWithEmailAndPassword(
                email: emailcontroller.text.trim(),
                password: passcontroller.text.trim());
        // Check user type and store data accordingly
        await FirebaseFirestore.instance
            .collection("users")
            .doc(userCredential.user!.uid)
            .set({
          "userEmail": emailcontroller.text.trim(),
          "userType": _userType,
          "name": namecontroller.text.trim()
        });
      }
    } catch (e) {
      Navigator.pop(context);
      showDialog(
          context: context,
          builder: (context) {
            return AlertDialog(
```

```
            title: Text(
              "Error While Creating an Account",
            ),
            backgroundColor: Color.fromARGB(255, 255, 174, 0),
          );
        });
    }
  }

  bool passconfirmed() {
    if (confirmpasscontroller.text.trim() == passcontroller.text.trim()) {
      return true;
    } else {
      return false;
    }
  }

  @override
  void dispose() {
    emailcontroller.dispose();
    passcontroller.dispose();
    super.dispose();
  }

  String? _validateField(String? value) {
    if (value == null || value.isEmpty) {
      return 'Please enter a value';
    }
    return null;
  }

  final _formKey = GlobalKey<FormState>();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Center(
          child: Container(
            margin: EdgeInsets.only(left: 25, right: 25),
            child: SingleChildScrollView(
              child: Form(
                key: _formKey,
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: Image.asset("assets/bgsignup.jpg"),
                    ),
                    SizedBox(
                      height: 25,
                    ),
                    Text(
                      "Create an Account!",
                      style: GoogleFonts.aboreto(
                          fontSize: 50, fontWeight: FontWeight.bold),
                    ),
```

```
                    SizedBox(
                      height: 10,
                    ),
                    Text(
                      "Please signup to continue",
                      style: TextStyle(fontSize: 20, color:
Colors.black54),
                    ),
                    SizedBox(
                      height: 10,
                    ),
                    TextFeildWidg(
                      controller: namecontroller,
                      hinttext: "Full Name of adoptioncenter/foster home",
                      Obsecuretext: false,
                      icon: Icon(
                        Icons.person,
                        color: Color.fromARGB(255, 255, 174, 0),
                      ),
                    ),
                    SizedBox(
                      height: 10,
                    ),
                    TextFeildWidg(
                      controller: emailcontroller,
                      hinttext: "Email",
                      Obsecuretext: false,
                      icon: Icon(
                        Icons.email,
                        color: Color.fromARGB(255, 255, 174, 0),
                      ),
                    ),
                    SizedBox(
                      height: 10,
                    ),
                    TextFeildWidg(
                        icon: Icon(
                          Icons.password,
                          color: Color.fromARGB(255, 255, 174, 0),
                        ),
                        controller: passcontroller,
                        hinttext: "Password",
                        Obsecuretext: true),
                    TextFeildWidg(
                        icon: Icon(
                          Icons.password,
                          color: Color.fromARGB(255, 255, 174, 0),
                        ),
                        controller: confirmpasscontroller,
                        hinttext: "Confirm Password",
                        Obsecuretext: true),
                    SizedBox(
                      height: 10,
                    ),
                    // Add a dropdown to select user type
                    DropdownButtonFormField<String>(
                      value: _userType,
```

```
                    onChanged: (String? value) {
                      setState(() {
                        _userType = value;
                      });
                    },
                    items: <String>['Adoption Center', 'Foster Home']
                        .map<DropdownMenuItem<String>>((String value) {
                      return DropdownMenuItem<String>(
                        value: value,
                        child: Text(value),
                      );
                    }).toList(),
                    decoration: InputDecoration(
                      labelText: 'Select User Type',
                      icon: Icon(
                        Icons.category,
                        color: Color.fromARGB(255, 255, 174, 0),
                      ),
                    ),
                  ),
                  SizedBox(
                    height: 10,
                  ),
                  Buttonsubmit(Textt: "Sign up", onTap: signup),
                  SizedBox(
                    height: 10,
                  ),
                  Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                      Text(
                        "Already have an account",
                        style: GoogleFonts.oswald(),
                      ),
                      SizedBox(
                        width: 6,
                      ),
                      GestureDetector(
                        onTap: widget.showloginpage,
                        child: Text(
                          "Login",
                          style: GoogleFonts.oswald(color: Colors.blue),
                        ),
                      )
                    ],
                  )
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

12.2.4 Login Page Code:

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:rescuerovers/BottomNavBar.dart';
import 'package:rescuerovers/Bottomforfoster.dart';
import 'ReqWidgets.dart';

class LoginPage extends StatefulWidget {
  LoginPage({super.key, required this.showregisterpage});
  final VoidCallback showregisterpage;
  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final emailcontroller = TextEditingController();
  final passcontroller = TextEditingController();

  Future signin() async {
    try {
      await FirebaseAuth.instance.signInWithEmailAndPassword(
          email: emailcontroller.text.trim(),
          password: passcontroller.text.trim());
      // After successful login, you can check the user type and navigate
accordingly
      // if (_userType == 'Adoption Center') {
      //   Navigator.pushReplacement(
      //     context,
      //     MaterialPageRoute(builder: (context) => BottomForFosters()),
      //   );
      //   // Navigate to Adoption Center dashboard
      // } else if (_userType == 'Foster Home') {
      //   // Navigate to Foster Home dashboard
      //   Navigator.pushReplacement(
      //     context,
      //     MaterialPageRoute(builder: (context) => BottomNavbarrr()),
      //   );
      // }
    } catch (e) {
      Navigator.pop(context);
      showDialog(
          context: context,
          builder: (context) {
            return AlertDialog(
              title: Text(
                "Wrong Credential",
              ),
              backgroundColor: Color.fromARGB(255, 255, 174, 0),
            );
          });
    }
  }
```

```
  final _formKey = GlobalKey<FormState>();
  @override
  void dispose() {
    emailcontroller.dispose();
    passcontroller.dispose();
    super.dispose();
  }

  String? _validateField(String? value) {
    if (value == null || value.isEmpty) {
      return 'Please enter a value';
    }
    return null;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Center(
          child: Container(
            margin: EdgeInsets.only(left: 25, right: 25),
            child: SingleChildScrollView(
              child: Form(
                key: _formKey,
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: Image.asset("assets/bgsignin.jpg"),
                    ),
                    SizedBox(
                      height: 25,
                    ),
                    Text(
                      "Welcome!!!",
                      style: GoogleFonts.aboreto(
                          fontSize: 50, fontWeight: FontWeight.bold),
                    ),
                    SizedBox(
                      height: 10,
                    ),
                    Text(
                      "Please Sign in to continue",
                      style: TextStyle(fontSize: 20, color:
Colors.black54),
                    ),
                    SizedBox(
                      height: 10,
                    ),
                    // Add a dropdown to select user type
                    // DropdownButtonFormField<String>(
                    //   value: _userType,
                    //   onChanged: (String? value) {
                    //     setState(() {
                    //       _userType = value;
```

```
                    //     });
                    //   },
                    //   items: <String>['Adoption Center', 'Foster Home']
                    //       .map<DropdownMenuItem<String>>((String value)
{
                    //     return DropdownMenuItem<String>(
                    //       value: value,
                    //       child: Text(value),
                    //     );
                    //   }).toList(),
                    //   decoration: InputDecoration(
                    //     labelText: 'Select User Type',
                    //     icon: Icon(
                    //       Icons.category,
                    //       color: Color.fromARGB(255, 255, 174, 0),
                    //     ),
                    //   ),
                    // ),
                    SizedBox(
                      height: 10,
                    ),
                    TextFeildWidg(
                      controller: emailcontroller,
                      hinttext: "Email",
                      Obsecuretext: false,
                      icon: Icon(
                        Icons.email,
                        color: Color.fromARGB(255, 255, 174, 0),
                      ),
                    ),
                    SizedBox(
                      height: 10,
                    ),
                    TextFeildWidg(
                        icon: Icon(
                          Icons.password,
                          color: Color.fromARGB(255, 255, 174, 0),
                        ),
                        controller: passcontroller,
                        hinttext: "Password",
                        Obsecuretext: true),
                    SizedBox(
                      height: 10,
                    ),
                    Buttonsubmit(Textt: "Login", onTap: signin),
                    SizedBox(
                      height: 10,
                    ),
                    Row(
                      mainAxisAlignment: MainAxisAlignment.center,
                      children: [
                        Text(
                          "Dosen't have an account?",
                          style: GoogleFonts.oswald(),
                        ),
                        SizedBox(
                          width: 4,
```

```
                        ),
                        InkWell(
                          onTap: widget.showregisterpage,
                          child: Text(
                            "Create One",
                            style: GoogleFonts.oswald(color: Colors.blue),
                          ),
                        )
                      ],
                    )
                  ],
                ),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

12.2.5 Foster Home Page Code:

```
import 'dart:io';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'Login&Signup/ReqWidgets.dart';
import 'package:image_picker/image_picker.dart';
import 'package:firebase_storage/firebase_storage.dart' as
firebase_storage;
import 'package:image_picker/image_picker.dart';
import 'package:file_picker/file_picker.dart';
import 'package:firebase_auth/firebase_auth.dart';

class FosterHomesPage extends StatefulWidget {
  FosterHomesPage({super.key});

  @override
  State<FosterHomesPage> createState() => _FosterHomesPageState();
}

class _FosterHomesPageState extends State<FosterHomesPage> {
  final dognamecontroller = TextEditingController();

  final dogagecontroller = TextEditingController();

  final dogbreedcontroller = TextEditingController();
  final User = FirebaseAuth.instance.currentUser;
  final doglocationpathcontroller = TextEditingController();
  final dogdescriptioncontroller = TextEditingController();
  final fosterhomesphonenumbercontroller = TextEditingController();

  File? _file;
  String _downloadUrl = '';
```

```
  Future<void> _pickFile() async {
    final picker = ImagePicker();
    final pickedFile = await picker.pickImage(source:
ImageSource.gallery);

    if (pickedFile != null) {
      setState(() {
        _file = File(pickedFile.path);
      });
    }
  }

  Future<void> _uploadFile() async {
    if (_file == null) {
      return;
    }

    final fileName = '${DateTime.now()}.png';
    final destination = 'files/$fileName';

    // Upload file to Firebase Storage
    await firebase_storage.FirebaseStorage.instance
        .ref(destination)
        .putFile(_file!);

    // Get download URL
    final downloadUrl = await firebase_storage.FirebaseStorage.instance
        .ref(destination)
        .getDownloadURL();

    setState(() {
      _downloadUrl = downloadUrl;
    });

    // Update Firestore document
    await FirebaseFirestore.instance.collection('dogsprofiles').add({
      "dogname": dognamecontroller.text.trim(),
      "dogbreed": dogbreedcontroller.text.trim(),
      "dogage": dogagecontroller.text.trim(),
      "dogcitylocation": doglocationpathcontroller.text.trim(),
      "dogdescription": dogdescriptioncontroller.text.trim(),
      'downloadUrl': downloadUrl,
      "FosterHomesPhoneNumber":
fosterhomesphonenumbercontroller.text.trim(),
      "CreateduserUid": User!.uid.trim(),
      "Fosterhomesemail": User!.email
    });

    ScaffoldMessenger.of(context).showSnackBar(SnackBar(
      content: Text('File uploaded successfully!'),
    ));

    _formKey.currentState!.validate();
  }

  String? _validateField(String? value) {
```

```
    if (value == null || value.isEmpty) {
      return 'Please enter a value';
    }
    return null;
  }

  final _formKey = GlobalKey<FormState>();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.white,
      ),
      body: SafeArea(
        child: Container(
          margin: EdgeInsets.only(left: 25, right: 25),
          child: SingleChildScrollView(
            child: Form(
              key: _formKey,
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Center(
                    child: Text(
                      "Create your own pet profile",
                      style: GoogleFonts.aboreto(
                          fontSize: 30, fontWeight: FontWeight.bold),
                    ),
                  ),
                  SizedBox(
                    height: 10,
                  ),
                  TextFeildWidg(
                    // Add validator function
                    controller: dognamecontroller,
                    hinttext: "Dog Name",
                    Obsecuretext: false,
                    icon: Icon(
                      Icons.pets,
                      color: Color.fromARGB(255, 255, 174, 0),
                    ),
                  ),
                  SizedBox(
                    height: 10,
                  ),
                  TextFeildWidg(
                      controller: dogagecontroller,
                      hinttext: "Dog Age",
                      Obsecuretext: false,
                      icon: Icon(
                        Icons.pets,
                        color: Color.fromARGB(255, 255, 174, 0),
                      )),
                  SizedBox(
                    height: 10,
                  ),
                  TextFeildWidg(
```

```
                    controller: dogbreedcontroller,
                    hinttext: "Dog Breed",
                    Obsecuretext: false,
                    icon: Icon(
                      Icons.pets,
                      color: Color.fromARGB(255, 255, 174, 0),
                    )),
                SizedBox(
                  height: 10,
                ),
                SizedBox(
                  height: 10,
                ),
                TextFeildWidg(
                    controller: doglocationpathcontroller,
                    hinttext: "Foster Homes City",
                    Obsecuretext: false,
                    icon: Icon(
                      Icons.pets,
                      color: Color.fromARGB(255, 255, 174, 0),
                    )),
                SizedBox(
                  height: 10,
                ),
                TextFeildWidg(
                    controller: dogdescriptioncontroller,
                    hinttext: "Description about dog",
                    Obsecuretext: false,
                    icon: Icon(
                      Icons.pets,
                      color: Color.fromARGB(255, 255, 174, 0),
                    )),
                TextFeildWidg(
                    controller: fosterhomesphonenumbercontroller,
                    hinttext: "Foster Homes Phone Number",
                    Obsecuretext: false,
                    icon: Icon(
                      Icons.pets,
                      color: Color.fromARGB(255, 255, 174, 0),
                    )),
                SizedBox(
                  height: 10,
                ),
                GestureDetector(
                  onTap: _pickFile,
                  child: ListTile(
                    tileColor: Colors.grey[200],
                    leading: Icon(Icons.add),
                    title: Text(
                      "Upload dog image",
                      style: GoogleFonts.oswald(),
                    ),
                  ),
                ),
                SizedBox(
                  height: 10,
                ),
```

```
                  Buttonsubmit(Textt: "Create Profile", onTap:
_uploadFile),
                  SizedBox(
                    height: 50,
                  )
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

12.2.6 Dog Profile Page Code:

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:rescuerovers/editdogprofile.dart';
import 'package:rescuerovers/fosterhomes.dart';
import 'package:rescuerovers/petslistview.dart';
import 'package:rescuerovers/petslistview2.dart';

class DogCreated extends StatefulWidget {
  DogCreated({Key? key}) : super(key: key);

  @override
  State<DogCreated> createState() => _DogCreatedState();
}

class _DogCreatedState extends State<DogCreated> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[200],
      appBar: AppBar(
        backgroundColor: Colors.white,
      ),
      body: SingleChildScrollView(
        child: Column(
          children: [
            ListTile(
              leading: Image.asset("assets/logo.png"),
              title: Text(
                "RescueRovers",
                style: GoogleFonts.aboreto(
                    fontSize: 33, fontWeight: FontWeight.bold),
              ),
            ),
            SizedBox(
              height: 20,
            ),
```

```
            Text(
              "Dog Profile",
              style: GoogleFonts.aboreto(fontSize: 40),
            ),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: Row(
                mainAxisAlignment: MainAxisAlignment.start,
                children: [
                  Text(
                    "Your Dogs:",
                    style: TextStyle(fontWeight: FontWeight.bold),
                  ),
                ],
              ),
            ),
            Container(
              child: StreamBuilder(
                stream: FirebaseFirestore.instance
                    .collection("dogsprofiles")
                    .where("CreateduserUid",
                        isEqualTo: FirebaseAuth.instance.currentUser!.uid)
                    .snapshots(),
                builder: (context, AsyncSnapshot<QuerySnapshot> snapshot)
{
                  if (snapshot.connectionState == ConnectionState.waiting)
{
                    return CircularProgressIndicator(); // Show a loading
indicator while data is being fetched
                  }
                  if (snapshot.hasError) {
                    return Text('Error: ${snapshot.error}');
                  }
                  final List<QueryDocumentSnapshot> documents =
                      snapshot.data!.docs;
                  return ListView.builder(
                    shrinkWrap: true,
                    physics: NeverScrollableScrollPhysics(),
                    itemCount: documents.length,
                    itemBuilder: (context, index) {
                      final doc = documents[index];
                      return InkWell(
                        onTap: () {
                          Navigator.push(
                              context,
                              MaterialPageRoute(
                                  builder: (context) =>
EditDogProfilePage(
                                        docId: doc
                                            .id, // Pass the docId to
EditDogProfilePage
                                        initialDogName: doc['dogname'],
                                        initialDogAge: doc['dogage'],
                                        initialDogBreed: doc['dogbreed'],
                                        initialDogCityLocation:
                                            doc['dogcitylocation'],
                                        initialDogDescription:
```

```
                                          doc['dogdescription'],
                                      )));
                          },
                          child: DogListVieww(
                            Dogname: doc['dogname'],
                            Dogage: doc['dogage'],
                            Dogbreed: doc['dogbreed'],
                            dogcitycloaction: doc['dogcitylocation'],
                            imgurl: doc['downloadUrl'],
                          ),
                        );
                      },
                    );
                  },
                ),
              ),
              Padding(
                padding: const EdgeInsets.all(8.0),
                child: GestureDetector(
                  onTap: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(builder: (context) =>
FosterHomesPage()),
                    );
                  },
                  child: Container(
                    child: Column(
                      mainAxisAlignment: MainAxisAlignment.center,
                      children: [
                        Icon(
                          Icons.add,
                          color: Colors.orange,
                        ),
                        Text(
                          "Create A Dog Profile",
                          style: GoogleFonts.oswald(fontSize: 20),
                        )
                      ],
                    ),
                    width: MediaQuery.of(context).size.width,
                    height: 200,
                    decoration: BoxDecoration(
                      color: Colors.grey[200],
                      border: Border.all(color: Colors.orange),
                    ),
                  ),
                ),
              ),
            ],
          ),
        ),
      );
  }
}
```

12.2.7 Dog Details Page Code:

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:rescuerovers/fosterhomes.dart';
import 'package:url_launcher/url_launcher.dart';
import 'Login&Signup/ReqWidgets.dart';
import 'package:flutter_email_sender/flutter_email_sender.dart';

class DogDetails extends StatefulWidget {
  DogDetails(
      {super.key,
      required this.dogAge,
      required this.dogBreed,
      required this.dogName,
      required this.dogCityLocation,
      required this.imgUrl,
      required this.dogdiscription,
      required this.fosterhomesphonenumber,
      required this.Fosterhomesemail});
  final String dogAge;
  final String dogBreed;
  final String dogName;
  final String dogCityLocation;
  final String imgUrl;
  final String dogdiscription;
  final String fosterhomesphonenumber;
  final String Fosterhomesemail;

  @override
  State<DogDetails> createState() => _DogDetailsState();
}

class _DogDetailsState extends State<DogDetails> {
  void _sendadoptionemail() async {
    Uri uri = Uri.parse(
        "mailto:${widget.Fosterhomesemail}?subject=I want to adopt your
${widget.dogName}.Please contact me");
    if (await canLaunchUrl(uri)) {
      launchUrl(uri);
    } else {
      AlertDialog(
        title: Text("Error"),
        content: Text("An error occurred."),
        actions: <Widget>[
          ElevatedButton(
            onPressed: () {
              Navigator.of(context).pop();
            },
            child: Text("OK"),
          ),
        ],
      );
    }
  }
```

```
  void _launchwhatsapp() async {
    String message = 'Hello,I Want Your Dog';
    final url = Uri.parse(

'https://wa.me/${widget.fosterhomesphonenumber}?text=${Uri.encodeFull(mess
age)}');
    if (await canLaunchUrl(url)) {
      await launchUrl(url);
    } else {
      AlertDialog(
        title: Text("Error"),
        content: Text("An error occurred while contacting on WhatsApp."),
        actions: <Widget>[
          ElevatedButton(
            onPressed: () {
              Navigator.of(context).pop();
            },
            child: Text("OK"),
          ),
        ],
      );
    }
  }

  //
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Center(
            child: Text(
          widget.dogName,
          style: GoogleFonts.oswald(),
        )),
        backgroundColor: Colors.white,
      ),
      backgroundColor: Colors.grey[200],
      body: SingleChildScrollView(
        child: Column(
          children: [
            Padding(
              padding: const EdgeInsets.all(14.0),
              child: Container(
                width: MediaQuery.of(context).size.width,
                height: (MediaQuery.of(context).size.height) / 2,
                decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(20.0),
                ),
                child: ClipRRect(
                  borderRadius: BorderRadius.circular(20.0),
                  child: Image.network(
                    widget.imgUrl,
                    fit: BoxFit.cover,
                  ),
                ),
              ),
            ),
```

```
            Text(
              widget.dogName,
              style: GoogleFonts.aboreto(
                  fontSize: 50, fontWeight: FontWeight.bold),
            ),
            SingleChildScrollView(
              scrollDirection: Axis.horizontal,
              child: Row(
                children: [
                  Padding(
                    padding: const EdgeInsets.all(8.0),
                    child: Container(
                      height: 70,
                      width: (MediaQuery.of(context).size.width) / 3,
                      decoration: BoxDecoration(
                          borderRadius: BorderRadius.circular(20.0),
                          color: Colors.white),
                      child: Column(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: [
                          Text(
                            "Dog Age",
                            style: GoogleFonts.oswald(),
                          ),
                          Text(
                            ("${widget.dogAge} years"),
                            style: GoogleFonts.oswald(),
                          )
                        ],
                      ),
                    ),
                  ),
                  Padding(
                    padding: const EdgeInsets.all(8.0),
                    child: Container(
                      height: 70,
                      width: (MediaQuery.of(context).size.width) / 3,
                      decoration: BoxDecoration(
                          borderRadius: BorderRadius.circular(20.0),
                          color: Colors.white),
                      child: Column(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: [
                          Text(
                            "Dog Breed",
                            style: GoogleFonts.oswald(),
                          ),
                          Text(
                            widget.dogBreed,
                            style: GoogleFonts.oswald(),
                          )
                        ],
                      ),
                    ),
                  ),
                ],
              ),
```

```
            ),
            Text(
              "Description",
              style: GoogleFonts.aboreto(
                  fontWeight: FontWeight.bold, fontSize: 25),
            ),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: Text(
                widget.dogdiscription,
                style: TextStyle(color: Colors.black54),
              ),
            ),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: Row(
                mainAxisAlignment: MainAxisAlignment.start,
                children: [
                  Text(
                    "Contact details",
                    style: GoogleFonts.aboreto(
                        fontSize: 20, fontWeight: FontWeight.bold),
                  ),
                ],
              ),
            ),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: Row(
                mainAxisAlignment: MainAxisAlignment.start,
                children: [
                  Text(
                    "Location:",
                    style: GoogleFonts.aboreto(
                      fontSize: 15,
                    ),
                  ),
                  SizedBox(
                    width: 10,
                  ),
                  Icon(Icons.location_on),
                  Text(
                    widget.dogCityLocation,
                    style: GoogleFonts.aboreto(
                        fontWeight: FontWeight.bold,
                        fontSize: 15,
                        color: Color.fromARGB(255, 255, 174, 0)),
                  ),
                ],
              ),
            ),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: Row(
                mainAxisAlignment: MainAxisAlignment.start,
                children: [
                  Text(
```

```
                    "Phone Number:",
                    style: GoogleFonts.aboreto(
                      fontSize: 15,
                    ),
                  ),
                  SizedBox(
                    width: 10,
                  ),
                  Text(
                    widget.fosterhomesphonenumber,
                    style: GoogleFonts.aboreto(
                        fontSize: 15, fontWeight: FontWeight.bold),
                  )
                ],
              ),
            ),
            Buttonsubmit(Textt: "Adopt Now", onTap: _sendadoptionemail),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: GestureDetector(
                onTap: _launchwhatsapp,
                child: Container(
                  decoration: BoxDecoration(
                      borderRadius: BorderRadius.circular(20),
                      color: const Color.fromARGB(255, 39, 39, 39)),
                  height: 50,
                  width: MediaQuery.sizeOf(context).width,
                  child: Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                      Image.asset(
                        "assets/whatsapp.png",
                      ),
                      SizedBox(
                        width: 10,
                      ),
                      Text(
                        "Contact On WhatsApp",
                        style: GoogleFonts.abel(color: Colors.white),
                      )
                    ],
                  ),
                ),
              ),
            ),
            SizedBox(
              height: 40,
            )
          ],
        ),
      ),
    );
  }
}
```

12.2.8 Edit Dog Profile Page Code:

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:google_fonts/google_fonts.dart';
import 'Login&SIgnup/ReqWidgets.dart';

class EditDogProfilePage extends StatefulWidget {
  final String docId;
  final String initialDogName;
  final String initialDogAge;
  final String initialDogBreed;
  final String initialDogCityLocation;
  final String initialDogDescription;

  EditDogProfilePage({
    required this.docId,
    required this.initialDogName,
    required this.initialDogAge,
    required this.initialDogBreed,
    required this.initialDogCityLocation,
    required this.initialDogDescription,
  });

  @override
  _EditDogProfilePageState createState() => _EditDogProfilePageState();
}

class _EditDogProfilePageState extends State<EditDogProfilePage> {
  late TextEditingController _dogNameController;
  late TextEditingController _dogAgeController;
  late TextEditingController _dogBreedController;
  late TextEditingController _dogCityLocationController;
  late TextEditingController _dogDescriptionController;

  @override
  void initState() {
    super.initState();
    _dogNameController = TextEditingController(text:
widget.initialDogName);
    _dogAgeController = TextEditingController(text: widget.initialDogAge);
    _dogBreedController = TextEditingController(text:
widget.initialDogBreed);
    _dogCityLocationController =
        TextEditingController(text: widget.initialDogCityLocation);
    _dogDescriptionController =
        TextEditingController(text: widget.initialDogDescription);
  }

  void _deleteDogProfile() {
    FirebaseFirestore.instance
        .collection('dogsprofiles')
        .doc(widget.docId)
        .delete()
        .then((_) {
      Navigator.pop(context);
```

```
    }).catchError((error) {
      print('Failed to delete dog profile: $error');
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.white,
        title: Text('Edit Dog Profile'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: SingleChildScrollView(
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              TextFormField(
                controller: _dogNameController,
                decoration: InputDecoration(
                    labelText: 'Dog Name', labelStyle:
GoogleFonts.oswald()),
              ),
              TextFormField(
                controller: _dogAgeController,
                decoration: InputDecoration(
                    labelText: 'Dog Age', labelStyle:
GoogleFonts.oswald()),
              ),
              TextFormField(
                controller: _dogBreedController,
                decoration: InputDecoration(
                    labelText: 'Dog Breed', labelStyle:
GoogleFonts.oswald()),
              ),
              TextFormField(
                controller: _dogCityLocationController,
                decoration: InputDecoration(
                    labelText: 'City Location',
                    labelStyle: GoogleFonts.oswald()),
              ),
              TextFormField(
                controller: _dogDescriptionController,
                decoration: InputDecoration(
                    labelText: 'Description', labelStyle:
GoogleFonts.oswald()),
              ),
              SizedBox(height: 16),
              Buttonsubmit(
                  Textt: "Save Changes",
                  onTap: () {
                    _updateDogProfile();
                  }),
              Buttonsubmit(
                  Textt: "Delete Dog Profile",
                  onTap: () {
```

```
                    _deleteDogProfile();
                  }),
              ],
            ),
          ),
        ),
      );
  }

  void _updateDogProfile() {
    FirebaseFirestore.instance
        .collection('dogsprofiles')
        .doc(widget.docId)
        .update({
      'dogname': _dogNameController.text,
      'dogage': _dogAgeController.text,
      'dogbreed': _dogBreedController.text,
      'dogcitylocation': _dogCityLocationController.text,
      'dogdescription': _dogDescriptionController.text,
    }).then((_) {
      Navigator.pop(context);
    }).catchError((error) {
      print('Failed to update dog profile: $error');
    });
  }

  @override
  void dispose() {
    _dogNameController.dispose();
    _dogAgeController.dispose();
    _dogBreedController.dispose();
    _dogCityLocationController.dispose();
    _dogDescriptionController.dispose();
    super.dispose();
  }
}
```

12.2.9 Nearby Adoption Center Page Code:

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'dart:math';

class Nearbyadoption extends StatefulWidget {
  Nearbyadoption({super.key});

  @override
  State<Nearbyadoption> createState() => _NearbyadoptionState();
}

class _NearbyadoptionState extends State<Nearbyadoption> {
  // Function to generate a random number between 1 and 50
  int generateRandomNumber() {
    Random random = Random();
```

```
    return random.nextInt(50) + 1;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: SingleChildScrollView(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                '''
                Nearby Adoption Center
                ''',
                style: GoogleFonts.aboreto(
                    fontWeight: FontWeight.bold, fontSize: 25),
              ),
              SizedBox(
                height: 12,
              ),
              Container(
                child: StreamBuilder(
                  stream: FirebaseFirestore.instance
                      .collection('users')
                      .where('userType', isEqualTo: 'Adoption Center')
                      .snapshots(),
                  builder: (context, AsyncSnapshot<QuerySnapshot>
snapshot) {
                    if (snapshot.connectionState ==
ConnectionState.waiting) {
                      return CircularProgressIndicator();
                    }
                    if (snapshot.hasError) {
                      return Text('Error: ${snapshot.error}');
                    }
                    final List<QueryDocumentSnapshot> documents =
                        snapshot.data!.docs;
                    return ListView.builder(
                      shrinkWrap: true,
                      physics: NeverScrollableScrollPhysics(),
                      itemCount: documents.length,
                      itemBuilder: (context, index) {
                        final doc = documents[index];
                        // Generate a random number
                        int randomNumber = generateRandomNumber();
                        return GestureDetector(
                          onTap: () {},
                          child: Padding(
                            padding: const EdgeInsets.all(8.0),
                            child: ListTile(
                              leading: Icon(
                                Icons.location_on_sharp,
                                color: Color.fromARGB(255, 255, 174, 35),
                              ),
                              tileColor: Colors.grey[200],
                              title: Text(
```

```
                                doc['name'],
                                style: GoogleFonts.abel(),
                              ),
                              // Show the randomly generated number in the
subtitle
                              subtitle: Text(
                                '$randomNumber km away  Contact
Email:${doc["userEmail"]}',
                                style: TextStyle(
                                  fontStyle: FontStyle.italic,
                                  color: Colors.black87,
                                ),
                              ),
                            ),
                          ),
                        );
                      },
                    );
                  },
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

12.2.10 Location_Google Maps Page Code:

```
import 'dart:async';

import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';

class MapSample extends StatefulWidget {
  const MapSample({super.key});

  @override
  State<MapSample> createState() => _MapSampleState();
}

class _MapSampleState extends State<MapSample> {
  late GoogleMapController mapController;

  final LatLng _center = const LatLng(40.7128, -74.0060);

  void _onMapCreated(GoogleMapController controller) {
    mapController = controller;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
      body: GoogleMap(
        onMapCreated: _onMapCreated,
        initialCameraPosition: CameraPosition(
          target: _center,
          zoom: 10.0,
        ),
      ),
    );
  }
}
```

12.2.11 Nearest Foster Home Page Code:

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'dart:math';

class NearestAdoption extends StatefulWidget {
  const NearestAdoption({super.key});

  @override
  State<NearestAdoption> createState() => _NearestAdoptionState();
}

class _NearestAdoptionState extends State<NearestAdoption> {
  int generateRandomNumber() {
    Random random = Random();
    return random.nextInt(50) + 1;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: SingleChildScrollView(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                '''
                Nearby Foster Home
                ''',
                style: GoogleFonts.aboreto(
                    fontWeight: FontWeight.bold, fontSize: 25),
              ),
              SizedBox(
                height: 12,
              ),
              Container(
                child: StreamBuilder(
                  stream: FirebaseFirestore.instance
                      .collection('users')
                      .where('userType', isEqualTo: 'Foster Home')
                      .snapshots(),
```

```
                  builder: (context, AsyncSnapshot<QuerySnapshot>
snapshot) {
                    if (snapshot.connectionState ==
ConnectionState.waiting) {
                      return CircularProgressIndicator(); // Show a
loading indicator while data is being fetched
                    }
                    if (snapshot.hasError) {
                      return Text('Error: ${snapshot.error}');
                    }
                    final List<QueryDocumentSnapshot> documents =
                        snapshot.data!.docs;
                    return ListView.builder(
                      shrinkWrap: true,
                      physics: NeverScrollableScrollPhysics(),
                      itemCount: documents.length,
                      itemBuilder: (context, index) {
                        final doc = documents[index];
                        int randomNumber = generateRandomNumber();
                        return GestureDetector(
                            onTap: () {},
                            child: Padding(
                              padding: const EdgeInsets.all(8.0),
                              child: ListTile(
                                leading: Icon(
                                  Icons.location_on_sharp,
                                  color: Color.fromARGB(255, 255, 174,
35),
                                ),
                                tileColor: Colors.grey[200],
                                title: Text(
                                  doc['name'],
                                  style: GoogleFonts.abel(),
                                ),
                                subtitle: Text(
                                    "$randomNumber km away  Contact
Email:${doc["userEmail"]}"),
                              ),
                            ));
                      },
                    );
                  },
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

12.2.12 Favorite Page Code:

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:shared_preferences/shared_preferences.dart';

class FavouritePage extends StatefulWidget {
  const FavouritePage({Key? key}) : super(key: key);

  @override
  _FavouritePageState createState() => _FavouritePageState();
}

class _FavouritePageState extends State<FavouritePage> {
  late SharedPreferences _prefs;
  late List<String> _favoriteDogNames = [];

  @override
  void initState() {
    super.initState();
    initializeSharedPreferences();
  }

  Future<void> initializeSharedPreferences() async {
    _prefs = await SharedPreferences.getInstance();
    setState(() {
      _favoriteDogNames = _prefs
          .getKeys()
          .where((key) => _prefs.getBool(key) ?? false)
          .toList();
    });
  }

  Future<void> removeFromFavorites(String dogName) async {
    setState(() {
      _favoriteDogNames.remove(dogName);
      _prefs.remove(dogName); // Remove from SharedPreferences
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.white,
        title: Text('Favourite Dogs'),
      ),
      body: _favoriteDogNames.isEmpty
          ? Center(
              child: Text('No favorite dogs yet'),
            )
          : ListView.builder(
              itemCount: _favoriteDogNames.length,
              itemBuilder: (context, index) {
                final dogName = _favoriteDogNames[index];
                return ListTile(
```

```
                  tileColor: Colors.grey[200],
                  title: Text(
                    "Dogname : ${dogName}",
                    style: GoogleFonts.oswald(fontSize: 20),
                  ),
                  trailing: IconButton(
                    icon: Icon(
                      Icons.delete,
                      color: Colors.orange,
                    ),
                    onPressed: () {
                      removeFromFavorites(dogName);
                    },
                  ),
                  // You can add more details or actions here if needed
                );
              },
            ),
    );
  }
}
```

12.2.13 Bottom Navigation Bar Code:

```
import 'package:curved_labeled_navigation_bar/curved_navigation_bar.dart';
import
'package:curved_labeled_navigation_bar/curved_navigation_bar_item.dart';
import 'package:flutter/material.dart';
import 'package:rescuerovers/Home.dart';
import 'package:rescuerovers/Loaction_googlemaps.dart';
import 'package:rescuerovers/Login&Signup/NearestFosterHomes.dart';
import 'package:rescuerovers/favourite.dart';
import 'package:rescuerovers/fosterhomes.dart';
import 'package:rescuerovers/userdeatils.dart';

class BottomNavbarrr extends StatefulWidget {
  const BottomNavbarrr({super.key});

  @override
  State<BottomNavbarrr> createState() => _BottomNavbarState();
}

class _BottomNavbarState extends State<BottomNavbarrr> {
  final List screens = [
    HomePage(),
    FavouritePage(),
    NearestAdoption(),
    UserDetailsPage(),
  ];
  int _selectedindex = 0;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      bottomNavigationBar: CurvedNavigationBar(
        index: _selectedindex,
```

```
        backgroundColor: Colors.white,
        color: const Color.fromARGB(255, 32, 32, 32),
        items: const [
          CurvedNavigationBarItem(
            child: Icon(
              Icons.home_filled,
              color: Color.fromARGB(255, 255, 174, 0),
            ),
            label: 'Home',
            labelStyle: TextStyle(color: Color.fromARGB(255, 255, 174,
0)),
          ),
          CurvedNavigationBarItem(
            child: Icon(
              Icons.favorite,
              color: Color.fromARGB(255, 255, 174, 0),
            ),
            label: 'Favourites',
            labelStyle: TextStyle(color: Color.fromARGB(255, 255, 174,
0)),
          ),
          CurvedNavigationBarItem(
            child: Icon(Icons.pets_outlined,
                color: Color.fromRGBO(255, 174, 0, 1)),
            label: 'Locations',
            labelStyle: TextStyle(color: Color.fromARGB(255, 255, 174,
0)),
          ),
          CurvedNavigationBarItem(
            child:
                Icon(Icons.person_2, color: Color.fromARGB(255, 255, 174,
0)),
            label: 'User',
            labelStyle: TextStyle(color: Color.fromARGB(255, 255, 174,
0)),
          ),
        ],
        onTap: (index) {
          print(index);
          // Handle button tap
          setState(() {
            _selectedindex = index;
          });
        },
      ),
      body: screens[_selectedindex],
    );
  }
}
```

12.2.14 Foster Home Bottom Navigation Bar Code:

```
import 'package:curved_labeled_navigation_bar/curved_navigation_bar.dart';
import
'package:curved_labeled_navigation_bar/curved_navigation_bar_item.dart';
import 'package:flutter/material.dart';
import 'package:rescuerovers/DogsProfiles.dart';
import 'package:rescuerovers/Home.dart';
import 'package:rescuerovers/Loaction_googlemaps.dart';
import 'package:rescuerovers/Login&Signup/NearestFosterHomes.dart';
import 'package:rescuerovers/NearbyAdoptionCenter.dart';
import 'package:rescuerovers/favourite.dart';
import 'package:rescuerovers/fosterhomes.dart';
import 'package:rescuerovers/userdeatils.dart';

class BottomForFosters extends StatefulWidget {
  const BottomForFosters({super.key});

  @override
  State<BottomForFosters> createState() => _BottomNavbarState();
}

class _BottomNavbarState extends State<BottomForFosters> {
  final List screens = [
    DogCreated(),
    Nearbyadoption(),
    UserDetailsPage(),
  ];
  int _selectedindex = 0;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      bottomNavigationBar: CurvedNavigationBar(
        index: _selectedindex,
        backgroundColor: Colors.white,
        color: const Color.fromARGB(255, 32, 32, 32),
        items: const [
          CurvedNavigationBarItem(
            child: Icon(
              Icons.home_filled,
              color: Color.fromARGB(255, 255, 174, 0),
            ),
            label: 'Home',
            labelStyle: TextStyle(color: Color.fromARGB(255, 255, 174,
0)),
          ),
          CurvedNavigationBarItem(
            child: Icon(
              Icons.house,
              color: Color.fromARGB(255, 255, 174, 0),
            ),
            label: 'Adoption Center',
            labelStyle: TextStyle(color: Color.fromARGB(255, 255, 174,
0)),
          ),
          CurvedNavigationBarItem(
```

```
            child:
                Icon(Icons.person_2, color: Color.fromARGB(255, 255, 174,
0)),
            label: 'User',
            labelStyle: TextStyle(color: Color.fromARGB(255, 255, 174,
0)),
          ),
        ],
        onTap: (index) {
          print(index);
          // Handle button tap
          setState(() {
            _selectedindex = index;
          });
        },
      ),
      body: screens[_selectedindex],
    );
  }
}
```

12.2.15 Database:

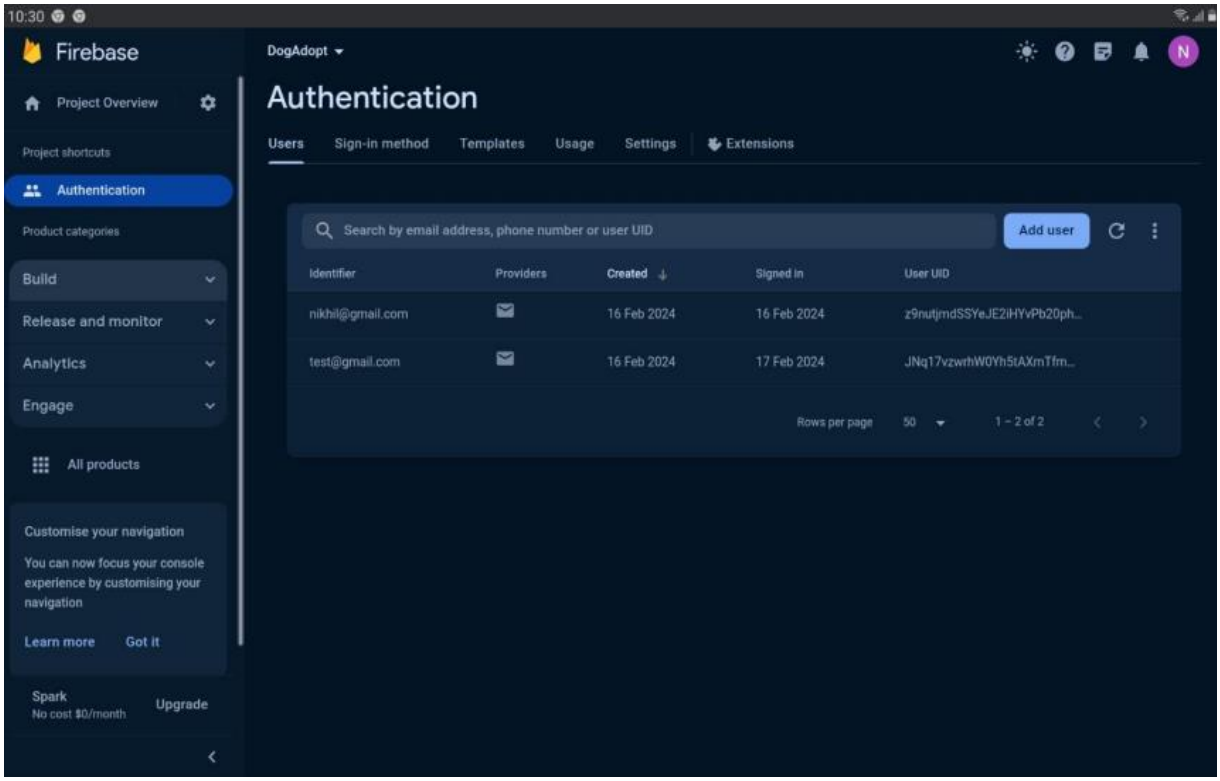

Figure 17(a): *Firebase Database authentication screen*

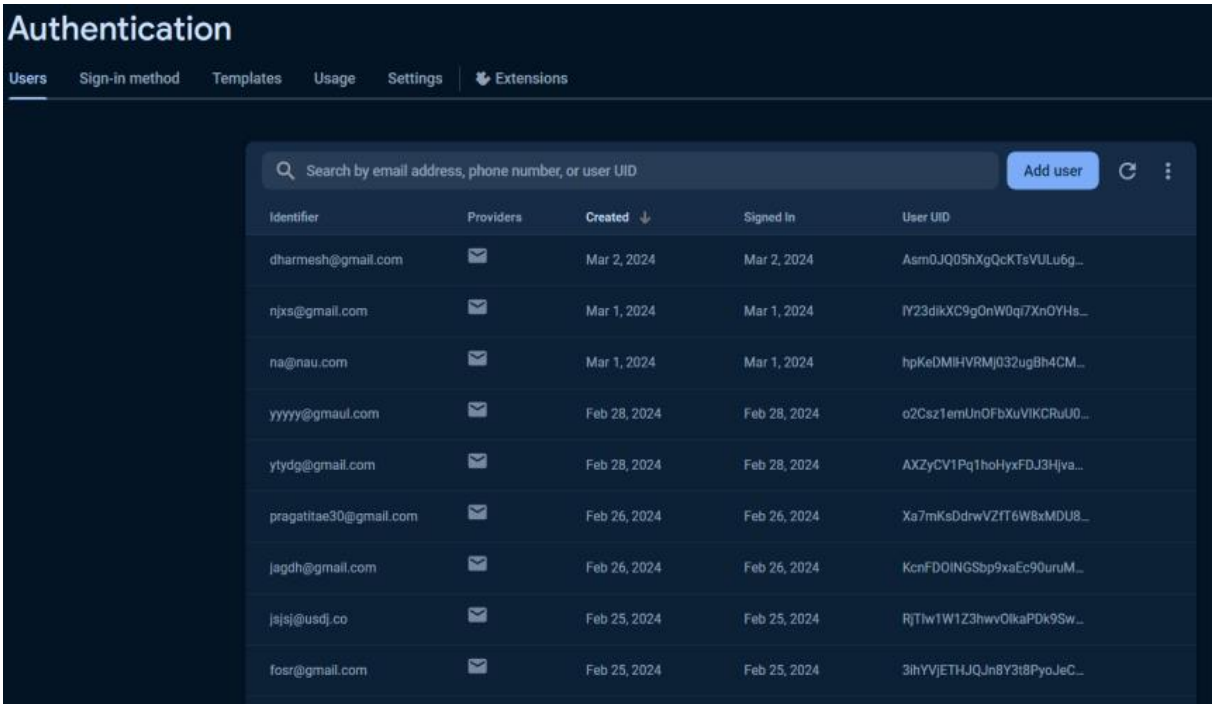

Figure 17(b): *Firebase Database authentication screen*

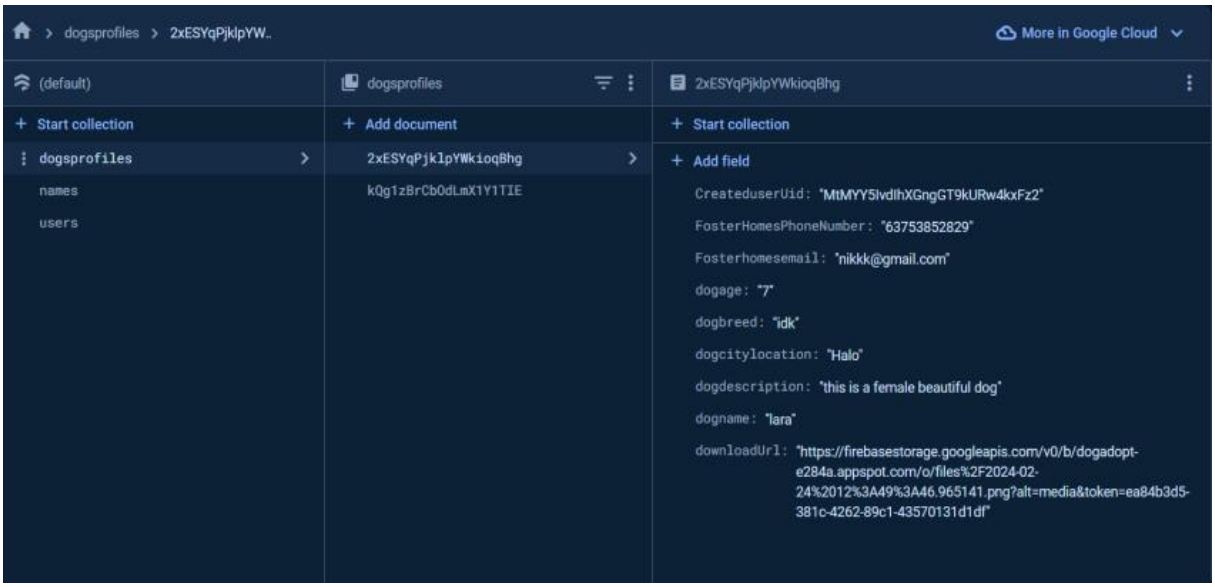

Figure 18: *Firebase Database Dog Profile screen*