Team Intercontinental
Duane Goodner
Traci Goreham
Erica May

Final Report

I. Introduction

Over the past eight weeks, our team has created a cross-platform (iOS and Android) mobile application for connecting animal lovers in search of a new pet with animals in need of loving homes. We used the Flutter SDK, the Dart programming language, and Firebase Firestore to create a functional and aesthetically pleasing application which allows shelter administrators to create accounts, login, add, edit, and delete animals that application administrators and users can view, sort, and filter. Additionally, we have gone above and beyond our client's requirements and created the ability for app users to create a favorite animal list, as well as, provided administrators with the ability to add, edit, and remove news items/PR blurbs in a news feed that is visible to both administrative and general users.

We have worked hard this quarter and are excited to share our Pet Matcher Application in greater detail in the sections below!

II. Description of Software from the User's Perspective

Our application supports two different types of users. The first is an administrator or employee of a pet adoption center who is trying to find homes for animals located within their pet adoption center. The second type of user is an individual from the public who is looking for a pet to bring home.

From the perspective of our first type of user, the administrator for the pet adoption center, our application will allow for the creation of a shelter account. Once an account is created, the adoption center administrator can create new "dating" profiles for each of their animal residents and has the ability to edit or remove existing animal profiles. These profiles include information about the animal's name, type (e.g., dog, cat, other), breed, adoption status, temperament/disposition, age, and an animal image. The administrative user also has the ability to create, edit, and delete news items or PR posts for their animal adoption center. Additionally, an administrator can see the current animal inventory and filter this inventory based on animal attributes.

From the perspective of our second type of user, an individual looking to adopt a pet, our application allows for the creation of a personal user account. Once an account is created, the user can log in and filter on a variety of preferences including date, animal type, breed, availability, and age. These preferences will yield a results list of prospective animal “matches”. The user can scroll through these matches, view additional details about each animal, choose favorites, and contact the pet adoption center where their chosen animal is located. The user can also view all their favorites on their favorites page. Additionally, a user has access to a news feed containing the latest pet news as posted by our pet adoption centers. The user can share these news posts via email or text message with their contacts.

III. Usage Instructions

Project Links

1. Link to Github repository for Pet Matcher application: https://github.com/duanegoodner/pet_searcher. Note that Dart source code files are in the lib folder and its sub-folders.
2. Link to Firebase Firestore database: https://console.firebase.google.com/u/0/project/pet-matcher-b534a/overview?pli=1

Setting Up a Flutter Development / Testing Environment

Testing the current version of our application needs to be done locally on a machine that has a Flutter development environment set up with Flutter version 2.0.4 or higher. The specific procedure you will need to follow will depend on your operating system and whether you want to test the application on an Android emulator, an iOS simulator, or both. Android emulators can run on any of the following operating systems: Windows, Linux, and MacOS. An iOS simulator can only run on MacOS. Detailed instructions for setting up Flutter can be found at https://flutter.dev/docs/get-started/install. The key steps in this process are:

1. Go to https://flutter.dev/docs/get-started/install, and select the link corresponding to your computer’s operating system. You will be directed to a page where you will need to:
   a. Follow the instructions to download and install the Flutter SDK on your computer.
   b. Follow the instructions to download and install Android studio.
   c. (optional) If you are running MacOS and want to test on an iOS simulator follow the “iOS setup” instructions to install Xcode and set up the iOS simulator.
   d. Follow the instructions to start Android studio and run the Android Studio Setup Wizard.
   e. Skip the instructions in the “Set up your Android device” section.
   f. Follow the instructions in the “Set up the Android emulator” section. We recommend setting up a Pixel 4 emulator with API 30. **Warning (**courtesy of CS 492 instructors)**:** Setting up an Android emulator will require you to enable VM acceleration on your computer. Doing this may break existing VirtualBox and VMWare installations unless you are using the latest versions of VirtualBox or VMWare.
2. Proceed to https://flutter.dev/docs/get-started/editor where you can find instructions on how to set up an editor. Choose the link for VS Code, and follow the enclosed instructions.

Downloading and Running the App

Once you have your Flutter development environment set up, you are ready to download and run our application. To do this:

1. Fork or clone a copy of Github repository https://github.com/duanegoodner/pet_searcher to your local machine.
2. Open the root level directory of the project with VS Code, and open file main.dart.
3. Go to https://flutter.dev/docs/get-started/test-drive?tab=vscode. Skip the instructions in the "Create the App" section, and follow the instructions in the "Run the app" section to start the app in either an Android emulator or an iOS simulator.

Usage Instructions for Testing and Navigating the App

Upon opening the app in an emulator or simulator, the user will see a landing page. From the landing page, the user has the option to log in or create a new account.

Test Case 1: Logging in and navigating as an admin user

1. From the Pet Matcher Landing Page, click on the "Log in" button to navigate to a screen containing input boxes for user email and password. Enter the following information:
    - Email: jane@gmail.com
    - Password: jane1234

2. Click the Login button. If the email and password are correct, the user will be directed to the admin homepage.

3. The admin homepage contains four tiles (News Feed, Add News Post, Inventory, and Add New Animal) which can be tapped on for quick navigation to the various main components of the administrator's side of the application. The descriptions of each feature are below:

    News Feed: This screen contains news and public relations articles.
    - The share icon allows the administrator to share this article with a contact via text message or email.
    - The pencil icon allows the administrator to edit the news item.
    - The trash icon allows the administrator to remove the news item from the news feed.

    Add News Post: This form allows the administrator to add a news item to the news feed. The administrator can include a title, post body, and image with each news post. The date will automatically be saved with the post.

    Inventory: A scrollable listview containing photos and summary information for each animal in the database.

- The magnifying glass icon on the app bar for this page pulls up an animal filter/search form. Animal criteria can be selected from the dropdown inputs of the form. The “breed” field does not become active until an “Animal type” has been selected. Pressing the “Search” button will filter the current animal inventory by the attributes selected.
- The three lines icon on the app bar for this page provides inventory sorting options.
- The plus icon on the app bar for this page takes the admin through the “Add a New Animal” process.
- The pencil icon and trash can icon on the animal images provide the admin with the ability to edit and remove animals from the inventory list.
- Clicking on a particular animal in the scrollable listview will take the user to a page with detailed information for that animal.

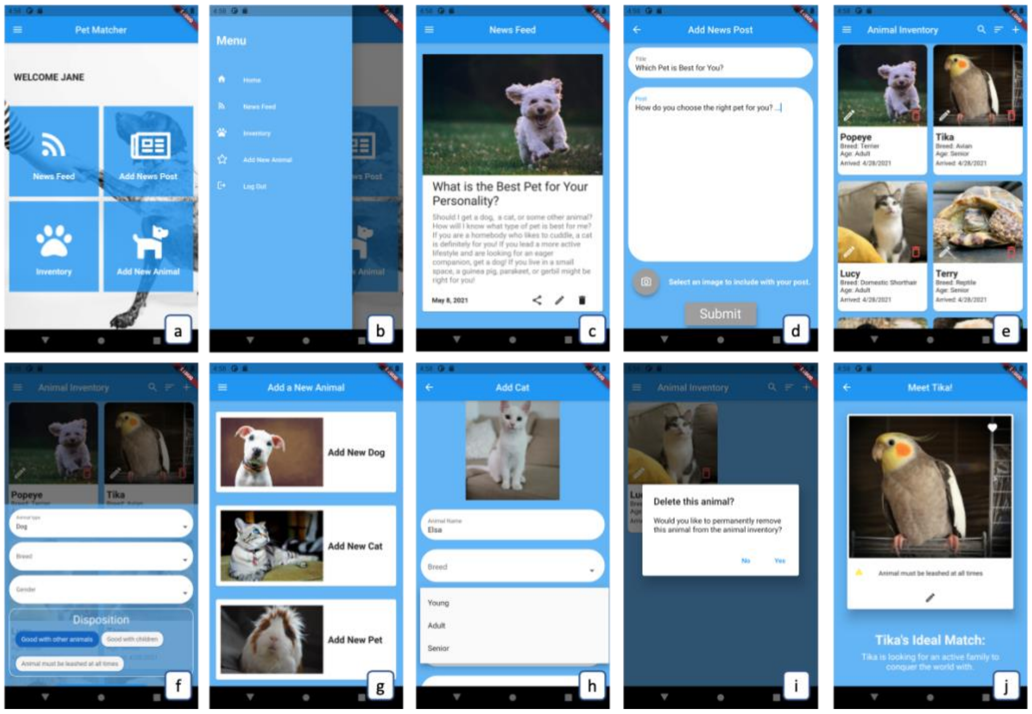


Figure 1. Sample of screen view for Admin User. (a) home screen, (b) home screen with menu drawer open, (c) news feed, (d) adding a news feed item, (e) animal inventory, (f) search menu for animal inventory, (g) selecting a new type of animal to add to inventory, (h) editing an existing animal, (i) dialog box confirming deletion of animal from inventory, and (j) detail screen for a specific animal.

Add a New Animal: This form allows the administrator to select a type of animal to add to the database.

- Upon selecting an animal type, a screen that allows an animal photo to be selected from the photos on your phone is displayed. If you are using an iPhone simulator, this screen should provide some default images, and you can choose any one of these images. If you are using an Android emulator, you will not have

    any default photos available, so you can just select the “photos” icon to navigate to an empty “Select a photo” screen, and then click on the back arrow.
  - After a photo (or the back arrow on Android) has been selected, the user is taken to a screen with a form where other animal data can be entered and uploaded to the database.

4. In the upper left corner of the admin homepage, an icon can also be tapped and a drawer menu will be displayed. This menu has options for the user to navigate to Home, News Feed, Inventory, Add New Animal, or Log Out. These options are the same as the admin home page tiles described above with the addition of the Home and Log Out features.

   Home: This is the same admin home page that appeared upon login.

   Log Out: The user is logged out, and the app returns to the Landing Page.

Test Case 2: Logging in and navigating as a general user

1. From the Pet Matcher Landing Page, click on the “Log in” button to navigate to a screen containing input boxes for user email and password. Enter the following information:
   - Email: tom@gmail.com
   - Password: tom1234

2. Click the Login button. If the email and password are correct, the user will be directed to the general user homepage.

3. The user homepage contains three tiles (News Feed, Search Animals, and Favorites) which can be tapped on for quick navigation to the various main components of the user’s side of the application. The descriptions of each feature are below:

   Search Animals: A scrollable listview containing photos and summary information for each animal in the database.

   - The magnifying glass icon on the app bar for this page pulls up an animal filter/search form. Animal criteria can be selected from the dropdown inputs of the form. The “breed” field does not become active until an “Animal type” has been selected. Pressing the “Search” button will filter the current animal inventory by the attributes selected.
   - The three lines icon on the app bar for this page provides inventory sorting options.
   - The heart icon on the animal photo allows the user to select, or favorite, an animal of interest. This animal will then be added to the user’s favorite list.

- Clicking on a particular animal in the scrollable listview will take the user to a page with detailed information for that animal.
    - From this animal detail screen, a user can click “Meet Me” to contact the shelter about the particular animal being viewed.

News Feed: This screen contains news and advice articles.
- The share icon allows the user to share this article with a contact via text message or email.

Favorites: This screen contains a scrollable listview of all the animals a user has favorited.

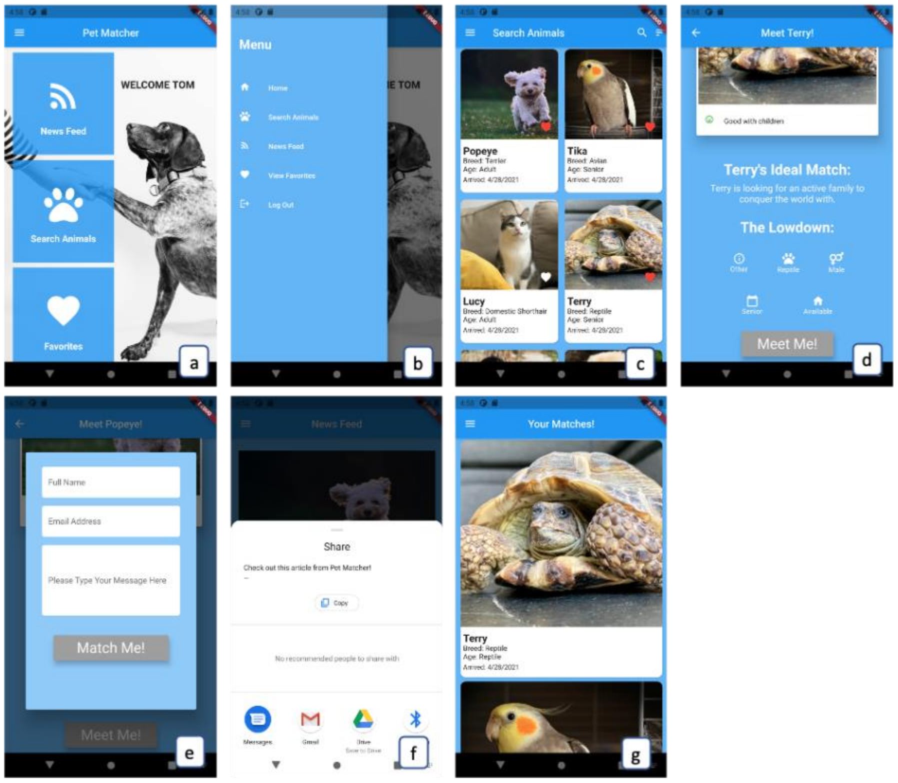


Figure 2. Sample of screen views for Public User. (a) home screen, (b) home screen with menu drawer open, (c) inventory screen, (d) animal detail screen, (e) dialog box to send information to animal shelter (f) bottom drawer for sharing news feed to other applications, and (g) screen with animals that user has marked as “favorites.”

4. In the upper left corner, a menu icon can be tapped and a drawer menu will be displayed. This menu has options for the user to navigate to the following pages:

   Home: This directs the user to the same admin home page that appeared upon login.
   - **Log out**: User is logged out, and the app returns to the Landing Page.

Test Case 3: Creating a new user

1. From the Pet Matcher Landing page, select the “Create Account” text to navigate to a page containing a form for new user information
2. Enter information for the new user’s first and last name, city, state, and postal code. Enter an email address and a password (must be at least 6 characters).
3. If you are creating an admin account, select the “I am a shelter admin” box. Otherwise, leave this field blank.
4. Click on Submit. If valid information has been entered in all fields, a new user will be added to the database, and the application will navigate to the Login Screen.
5. To verify that the newly created account is active, you can log in using the same email and password information provided in the previous step.

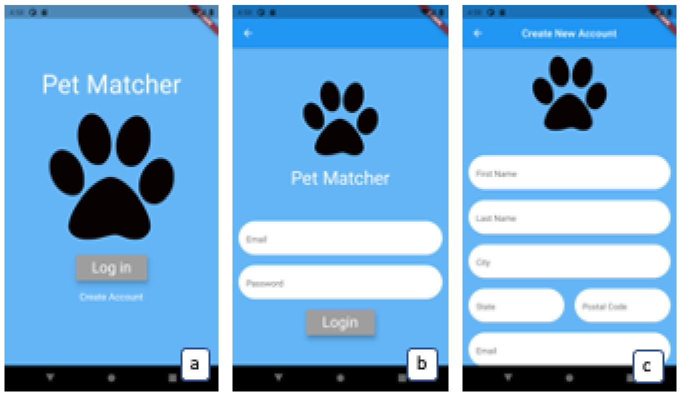


FIgure 3. (a) Main app landing page, (b) log-in screen, and (c) new account creation page.

Additional information:

If a user quits the application while logged in, the next time they start the application, they will be directed to their homepage (either admin or general user) instead of the application Landing Page.

IV. <u>Description of how software systems function together</u>

This application was written in the Dart programming language using the Flutter framework. Dart is a strongly typed, compiled, object-oriented language. Flutter can compile a single code-base to native iOS and Android code, allowing applications written with this framework to be cross-platform and extremely fast.

Google Cloud Firestore was used for a non-structured database, image storage, and user authorization. Various Flutter packages are available for seamless integration with Firestore. We also relied heavily on data streams from Firestore (rather than single-call queries) to easily keep the app state in sync with the database.

The application architecture primarily followed the “Model, View, Controller” paradigm. Many of the Service (aka Controller) and Model classes were managed using the Flutter GetIt package.

This package allows easy control over the class type (e.g. Singleton vs. Factory) and provides clean access to specific class instances without requiring that these business-logic classes be directly linked to any UI components.

In a Flutter application, all UI components and many non-UI components are nodes (aka “widgets”) in a tree data structure. Many of the details of passing state information from upper-level to lower-level widgets in our application were managed using the Provider package to minimize the amount of data passed through intermediate widgets. By using StreamProvider widgets that receive and store backend stream events, we were able to minimize the bandwidth needed to keep each UI view synchronized with backend data.

The diagram in Figure 3 shows how the process of logging in and displaying the appropriate user home screen is achieved by passing information between the user interface and database via an AppUserService class. Since basic authorization data for Firebase Users is stored separately from full user profile data, two calls must be made to the backend database to obtain data for a complete user profile. The initial call provides only email and password data to the backend which returns a basic Firebase User authorization profile (if the email and password correspond to a Firebase User). This information is sent as a stream that fires an event any time the user authorization state changes.

Upon receiving the Firebase authorization profile, the AppUserService class makes a second call to the database to obtain the user’s full profile information. The AppUserService class then converts this profile data into an AppUser object and provides it to the Flutter UI. Since the original authorization data is provided to the AppUserService class as a stream, AppUser object data is also sent to the UI as a stream that gets updated any time login / logout state changes. This stream is received by a StreamProvider widget near the top of the overall application widget tree. Lower level widgets can then access the most current AppUser data by referencing this StreamProvider instead of having to make a new call to the database. The curved green arrow in Figure 3 indicates how AppUser information is sent to the UserHomeScreen widget to determine which type of home screen to display (standard user or admin).

Provider widgets were also used for internal state management in other parts of the application. The green arrows in Figure 4 indicate how a top level StreamProvider of data type List<Animal> was combined with a ChangeNotifierProvider of data type <AnimalFilter> to display information for certain animals based on user-determined search criteria. The orange arrow in this figure shows how information about a user’s list of favorite animals was transferred to a “favorites” icon widget to determine the icon color (red if the animal is on the user’s list favorites, and white if it is not).

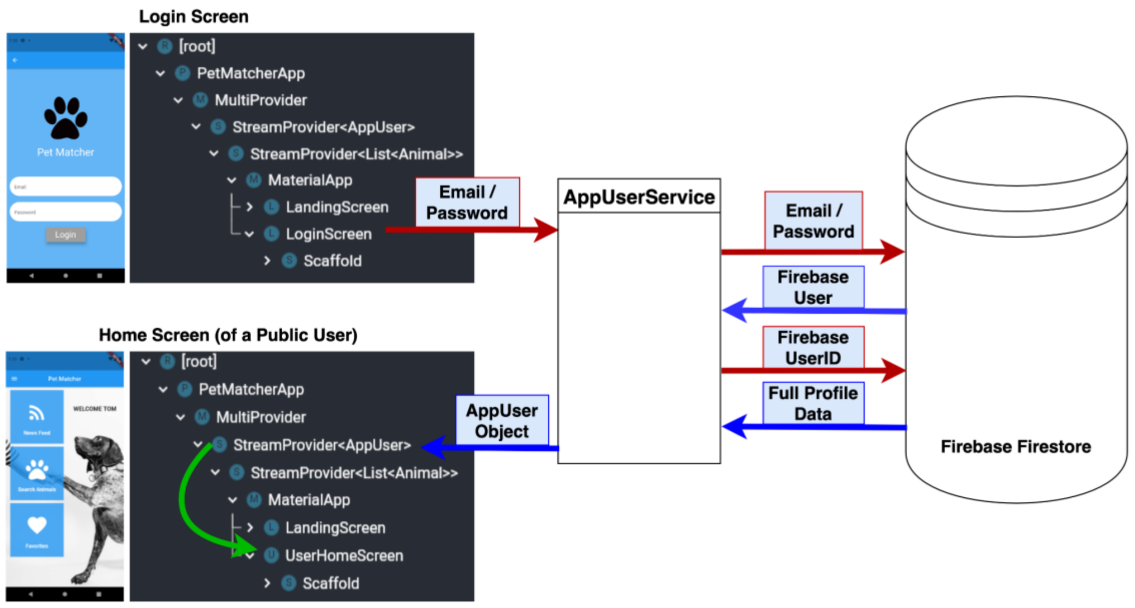
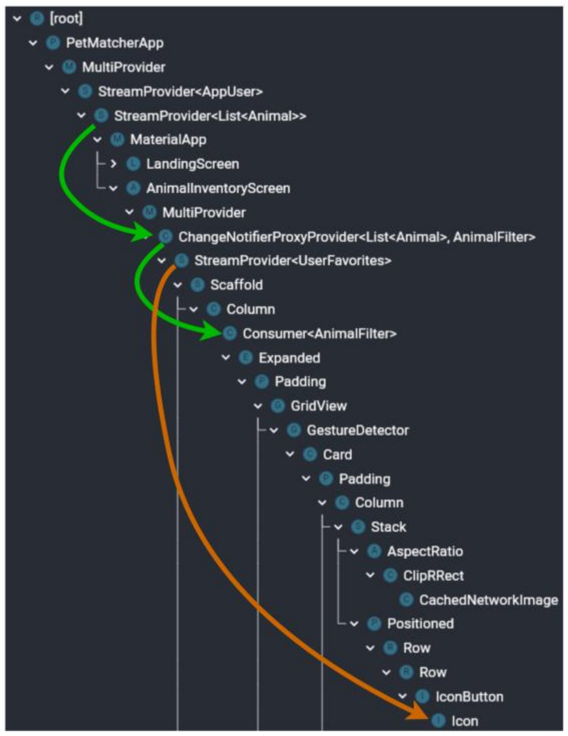


Figure 4. Communication between UI and database during the user sign-in process. Note that blue arrows indicate data that is sent as a stream (and is re-sent whenever information on the upstream side has changed).



```
[root]
  PetMatcherApp
    MultiProvider
      StreamProvider<AppUser>
        StreamProvider<List<Animal>>
          MaterialApp
            LandingScreen
            AnimalInventoryScreen
              MultiProvider
                ChangeNotifierProxyProvider<List<Animal>, AnimalFilter>
                  StreamProvider<UserFavorites>
                    Scaffold
                      Column
                        Consumer<AnimalFilter>
                          Expanded
                            Padding
                              GridView
                                GestureDetector
                                  Card
                                    Padding
                                      Column
                                        Stack
                                          AspectRatio
                                            ClipRRect
                                              CachedNetworkImage
                                          Positioned
                                            Row
                                              Row
                                                IconButton
                                                  Icon
```

Figure 5. Widget tree showing how Animal data (green arrows) and User Favorites data (orange arrow) is passed from upper-level to lower-level widgets.

Software Structure: Folder Structure

- android/ - Auto-generated Android project/build files. We modified some items here for Firebase setup and hardware permissions.
- ios/ - Auto-generated ios project/build files. We modified some items here for Firebase setup and hardware permissions.
- assets/ - Contains images for our application
- lib/ - All source code will go here
  - models/ - “Pure Dart” classes
  - navigation/ - Contains navigation routes and controller
  - screens/ - Contains one file for each of the screens listed above
  - services/ - Contains files/functions for reading/writing from database
  - widgets/ - Contains files with various custom widgets that get used in screens
  - main.dart - Entry point of application
  - app.dart - Establishes routing, application design, and providers
  - locator.dart - Service locator file that uses the GetIt package. Allows easy access to classes that are not part of the widget tree as well as control over the class type (e.g. Factory vs. Singleton)
  - styles.dart - Establishes color themes and font styles for our application
- test
  - unit/
  - widget/
  - integration/
- web/ - Auto-generated Desktop project/build files. Our team will only touch this folder if we work on a desktop version of the app (stretch goal).
- Pubspec.yaml - specifies Project dependencies / packages

Database Structure

- Users
  - First name
  - Last name
  - City
  - State
  - Zip code
  - Email
  - Password
  - Favorites (list of animal objects)
- Animals
  - Date added to the database
  - Name
  - Type (dog, cat, other)
  - Status
  - Breed
  - Disposition
  - Age (young, adult, senior)

    - Gender
    - Image url (images stored separately in file storage)
- NewsPost
    - Title
    - Body
    - Date
    - Image url (images stored separately in file storage)
- Message
    - Name
    - Email
    - Interest
    - Details

V. Software Libraries, Languages, APIs, Development Tools, Servers, and Systems

The following technologies were used in the development of our application:

- Flutter SDK: Flutter version 2.0.4 or higher
- Language: Dart (version 2.12.2 or higher)
- Database: Firebase firestore
- APIs/packages:
    - firebase_core
    - cloud_firestore
    - firebase_storage
    - firebase_analytics (user analytics)
    - firebase_auth (login/authentication)
    - font_awesome_flutter (additional icons)
    - provider (state management)
    - get_it (state management)
    - image_picker (camera/photos)
    - cached_network_images (images)
    - intl (date manipulation)
    - share (sharing)
    - flutter_form_builder (forms)
    - multi_select_flutter (forms)
    - rxdart (async)
    - stream_transform (streams)
    - flutter_test (testing)
- Development Environments: VSCode and Android Studio
- Development Tools: Emulators (Android AVD, iOS Simulator), VSCode debugging tools, Flutter Performance Tooler, Firebase Analytics
- Repository: Github

VI. Teammate Responsibilities

Duane Goodner: Duane spent the early weeks of our project setting up our Firestore database and implementing proper account creation and authorization using the firebase_auth package.

He was also our state management expert, extensively researching the provider package and implementing and integrating this package with our animal and user model classes. Through the use of this package and the way he organized and refactored code (for example, navigation components), he helped our application better adhere to the abstraction principle and keep our application organized and easy to maneuver by all group members. Additionally, Duane created our animal inventory screen and implemented much of the filtering and sorting features essential to our application. He also worked on the state management for adding favoriting functionality to the user-side of our application.

Traci Goreham: Initially, Traci focused her time on creating the initial routing of our application, developed various screens, and reusable UI components to get our application development jump-started. She created some essential reusable widgets such as the form text boxes, form dropdown boxes, and standard icon tiles, as well as, created a styles file to provide continuity throughout our application, She designed the landing page, login page, the final user and admin home screens, and the news feed. Additionally, she created the forms and functionality for adding, editing, and removing both animals and news items to/from our database.

Erica May: Erica's early work, including the account creation page, the admin and user drawers, and earlier versions of the admin and user home screens, provided the foundation for our navigation to all of the essential features of our application. She also designed our animal detail screen and created the form allowing users to contact the shelter about an animal match. Additionally, her extensive work creating the ability for users to favorite animals and view those animals in an animal matches screen, brought the "animal dating" theme, desired by our client, to life.

VII. <u>Weekly Breakdown of Tasks Completed</u>

Duane Goodner

Week	Task Description	Time (hours)
2	Initialize Flutter project and set up Github repository	1
	Research Github actions for auto-testing pull-requests	1
	Contribute to project plan write-up	3
	Create flow diagram showing app flow from admin perspective	3
	Create flow diagram showing app flow from user perspective	3
	Weekly Total	**11**
3	Research test methods	2
	Research state management tools	2
	Create Firebase Firestore project (with database and authorization selected)	3

	Build User Model class, User service class and User database fields	4
	Get user authorization working (with Firebase)	4
	Weekly Total	**15**
4	Begin integrating Provider package into our application	4
	Modify sign-in logic to accommodate users who are already logged in	3
	Incorporate GetIt to manage Service classes	4
	Animal inventory screen view	4
	Set up navigation for new views	1
	Weekly Total	**16**
5	Create Animal Inventory screen view	4
	Begin building animal search functions	3
	Integration tests for User account creation / log-in / log-out / re-log-in	4
	Create midpoint check document, prep repository, make video	3
	Weekly Total	**14**
6	Continue working on filtering, add filter logic for multiple options	10
	Testing for Animal searching / filtering / viewing	4
	Weekly Total	**14**
7	Add selecting multiple options for disposition, complete searching	5
	Add ability to select multiple options for animal disposition	4
	Refactor code for animal inventory/animal detail screen	2
	Assess for adaptive / responsive features	3
	Weekly Total	14
8	Redesign animal inventory screen	4
	Add ability to favorite animals and news items	8
	Extract business logic code into Services	3
	Weekly Total	15
9	Add ability to favorite animals and news items	7
	End-to-end testing of whole application	2
	Contribute to the creation of project poster (pdf)	4
	Contribute to final report write-up	4
	Weekly Total	17
10	Contribute to final report write-up	4
	Create application video	2

	App demonstrations	2
	Weekly Total	8
Total for Quarter		124

Traci Goreham

Week	Task Description	Time (hours)
2	Contribute to project plan write-up	6
	Create flow diagram showing app flow from admin perspective	0.25
	Create flow diagram showing app flow from user perspective	0.25
	Research various Flutter packages that may be useful for project	4
	Weekly Total	10.5
3	Research test methods	2
	Research state management tools	2
	Set-up file directory for app	1.5
	Landing/Welcome screen view	3
	Sign in screen view	3
	Set up navigation for Landing, Sign-in, and Account Setup views	1.5
	Unit / widget tests for account creation and sign-in components	3
	Weekly Total	16
4	Review / critique / adjust theme and styling from initial screens.	3
	Build Animal Model class, Animal database fields, and Animal CRUD functions	2.5
	Set-up camera/photo functionality for Add an animal form	2
	Integrate images and add animal form with the Firebase	2
	Add an animal form (available for admin only)	7
	Set up navigation for new views	1
	Weekly Total	17.5
5	Review / critique / adjust theme and styling from initial screens	1
	Build NewsItem Model classes, database fields and CRUD functions	1
	News item feed page (view and integrate with database), sharing	8
	Set-up navigation for new views	1.5
	Create midpoint check document, prep repository, make video	3
	Weekly Total	14.5

6	Create form for adding news item, integrate with database	7
	Integrate form with editing icon on newsfeed	3
	Testing for News Item creation / editing / viewing	3
	Weekly Total	13
7	Refactor code for animal inventory/animal detail screen	4
	Add functionality to edit/remove animal once adopted from admin inventory	6
	Assess for adaptive / responsive features	2
	Weekly Total	12
8	Redesign user and admin home screens and setup routing	7
	Create styles.dart file and/or theme widgets to help standardize styles across screens	4
	Remove unused code	2
	Tackle UI adjustments	3
	Weekly Total	16
9	UI adjustments	2
	End-to-end testing of whole application	4
	Contribute to the creation of project poster (pdf)	3
	Contribute to final report write-up	6
	Weekly Total	15
10	Contribute to final report write-up	5
	App demonstrations	2
	Weekly Total	7
Total for Quarter:		**121.5**

Erica May

Week	Task Description	Time (hours)
2	Contribute to project plan write-up	3
	Research prototyping options	2
	Create prototype images of important view layouts	4
	Research various Flutter packages that may be useful for project	2
	Weekly Total	11
3	Research test methods	2
	Research state management tools	2

	Get user authorization working (with Firebase)	2
	Account setup form view	6
	Unit / widget tests for account creation and sign-in components	2
	Weekly Total	14
	Review / critique / adjust theme and styling from initial screens.	2
	Create navigation drawer	5
	Add admin home screen view	7
	Add user home screen view	4
4	**Weekly Total**	18
	Review / critique / adjust theme and styling from initial screens	1
	Animal detail screen view	8
	Create midpoint check document, prep repository, make video	4
5	**Weekly Total**	13
	Refactor user/admin home pages to reflect actual news and new pets	12
6	**Weekly Total**	12
	Build InfoRequest Model classes, database fields, and CRUD functions	4
	"Request more information" / "Contact shelter" form view (available for general user only)	6
	Tests for InfoRequest	2
7	**Weekly Total**	12
	Add ability to favorite animals and news items	18
8	**Weekly Total**	18
	Add ability to favorite animals and news items	10
	End-to-end testing of whole application	2
	Contribute to the creation of project poster (pdf)	2
	Contribute to final report write-up	4
9	**Weekly Total**	18
	Contribute to final report write-up	2
	Create application video	4
	App demonstrations	2
10	**Weekly Total**	8
Total for Quarter:		**124**

VIII. <u>Conclusion</u>

Overall, we are very pleased with the work we have completed this quarter. We were able to deliver a multi-platform application that fits all of the original specifications of our client. Additionally, we were able to add two major stretch goals to our application in the completion of a news feed and the ability for a general user to favorite animals of interest and present those favorites on a matches page for easy viewing.

Although we have accomplished much, we recognize that there are definite areas where we could improve upon our application if we had more time. For example, we had initially wanted to create an extensive test suite including unit, widget, and integration tests; however, we gave the completion of required functionality priority, and never developed the test suite we originally desired. Additionally, there were user experience features and additional application functionality we wanted to include.

Despite these areas of want, we are extremely proud to present our Pet Matcher application to you! We have learned much throughout this development process that we intend to bring to our future software development jobs. We hope that you enjoy using our application!

IX. <u>Sources</u>

Code References:
https://levelup.gitconnected.com/using-firebase-in-flutter-web-4b99952180aa
https://www.filledstacks.com/post/flutter-dependency-injection-a-beginners-guide/
https://www.filledstacks.com/post/flutter-architecture-my-provider-implementation-guide/
https://fireship.io/lessons/advanced-flutter-firebase/
https://dev.to/rrtutors/upload-image-to-firebase-storage-flutter-android-ios-3f35
https://medium.com/firebase-developers/dive-into-firebase-auth-on-flutter-email-and-link-sign-in-e51603eb08f8
https://firebase.flutter.dev/docs/auth/usage/
https://pub.dev/packages/image_picker
https://pub.dev/packages/share
https://flutter.dev/docs/cookbook (This documentation provides many different code samples demonstrating how to use various Flutter widgets. It has been a helpful starting point for many of our UI components.)

Sources for Images
All animal images found for free at: https://unsplash.com/s/photos/pets
Paw Print Image Reference: https://wikiclipart.com/dog-paw-prints-clip-art_37264/