MAN'S BEST FRIEND

Software Design Document

Created by:

Andrew Wallace

Joey Dyer

Uday Cherukuri

Change Log

March 2nd: Added “Activities” functional requirement; edited details of “Profiles” functional requirement (SQLite); Updated Table of Contents; Expanded glossary (activity, reminder, informational reminder, bark)

March 7th: Fixed grammatical errors; Added subsections to introduction (Purpose and Scope); Added functional requirement “Dog Database”; Updated use case diagram; Updated Class Diagram; Added activity diagrams (Dog Database, Activities, Profiles, Barks);

March 8th: Updated UI Layouts (Login Screen, Profile Selection Screen, Dog Database Screen, Reminder Screen, Profile Screen, Barks Screen, Activities Screen); Added more detail to NFR 1,2,3; Updated the Table of Contents to correctly reflect the document;

March 22: Added descriptions to the use case diagram, high level class diagram, and the activity diagrams; Added priority to the functional requirements; Changed table of contents to only include one page number for each section; Changed “Design Models” to “Analysis Models”;

March 29: Added “Design Models” section; Added detailed class diagram; Added database diagram; Added sequence diagrams; Added descriptions for all of the design model diagrams.

March 31: Fixed detailed class diagram to include the proper format; Fixed formatting for entire document; Updated the pages listed in the table of contents;

Table of Contents

1. *Introduction*……………………………………………………………...…..4
2. *Glossary*…………………………………………....…………………4
3. *Functional Requirements*………………………………………………...5
4. *Non-Functional Requirements*…………………………....…………....7
5. *Hardware Requirements*………………………………………………...7
6. *Design Specifications*…………………………………………………....8
    a. *Components*……………………………………………………......8
    b. *System Operation*……………………………………………………9
7. *Analysis Models*……………………………………………………….....10
    a. *Use Case Diagram*……………………………………………………..10
    b. *Activity Diagrams*……………………………………………………...11
    c. *High Level Class Diagram*…………………………………..…………16
8. *UI Layout*…………………………………………………...……….…...17
9. *Design Models*…………………………………………………………….21
    a. *Detailed Class Diagram*………………………………………………..21
    b. *Database Diagram*………………………………………………………22
    c. *Sequence Diagrams*………………………………………………………23

1. Introduction:

a. Purpose

This document describes the requirement specifications of MansBestFriend. Included in this document are definitions of important terms, the application's functionalities, specifications about the design, an overall description on how the software should operate, and a few examples of potential user interface designs. This document also includes a basic class diagram, a use case diagram, and activity diagrams for each use case.

b. Scope

Man's Best Friend is an Android app designed to assist users with taking care of their dog. This includes anything from helpful reminders to location features that aim to create a dog owner community where users can mutually benefit from each other. The basic principle of Man's Best Friend is that your dog has a profile in which information for general health care is based on their age, weight, breed, and previous experience.

2. Glossary:

***Breed* -** a specific group of domestic animals having homogeneous appearance (phenotype), homogeneous behavior, and/or other characteristics that distinguish it from other organisms of the same species and that were arrived at through selective breeding.

***Profile* -** Data component that encapsulates all information associated with a specific dog. This includes picture, age, weight, breed, allergies, medical conditions, etc.

***Vaccine* -** Medication periodically given to dogs in order to keep them disease free and healthy.

***Bark* -** Community based message that will be viewable for everyone within a predefined radius of the sent location.

***Activity* -** A message displayed in a log format that tells the user what actions have been performed for their dog, along with the time and date of that action and accompanied by an optional custom comment

***Reminder* -** Uses Android calendar API to display messages to the user at set time intervals. These include predefined reminders (feed, bathe, medicate, walk, birthday) and users can add their own custom reminders as well

***Informational Reminder* -** A message that is displayed immediately after a reminder that displays relevant information based on the type of reminder as well as the breed of dog that the reminder is associated with

3. Functional Requirements:

a. FR1: Profiles

Priority: ***HIGH***

Description:

Users should be able to create a dog profile. A blank template will be presented for the user to manually enter information.

Details:

- Profile features include biography, pictures, general information (age, breed, sex, etc.)
- Profiles will be stored in an SQLite database located on each separate device
- Ability to support multiple profiles on a single device
- Dog breed will be selected from the predefined list of 50 top breeds. An option for "other" will be available, however the data will not be as accurate as a predefined breed.

b. FR2: Reminders

Priority: ***HIGH***

Description:

Users should be able to pick reminders. They will pick these from a list of available reminders, as well as be able to add custom reminders of their own

Details:

- Reminder frequency should be customizable from every n hours to every m years

- Reminders should have an optional message to display. Ex: “Medication Reminder: Spot needs the antibiotic today”
- Reminders should be tied to an existing dog profile

c. FR3: Dog Database

Priority: ***MEDIUM***

Description:

The app should contain basic healthcare information for the top 50 most popular dog breeds. This will act as a suggestion for each dog’s profile.

Details:

- General information should be offered at the time of certain preset reminders that is relevant to the individual reminder. Ex: when the feeding reminder activates for a chihuahua profile, also display the information “Professionals recommend feeding chihuahuas puppy food due to higher nutritional value”
- Stored in SQLite database on the device.
- Informational reminders should have a customizable frequency setting, including “Always”, “Frequent”, “Less Frequent”, and “Never”. A count of each reminder is kept to keep track of frequency of informational reminders.
    - *Always:* an informational reminder is always displayed after a reminder is displayed to the user
    - *Frequent:* an informational reminder is displayed after every three reminders
    - *Less Frequent:* an informational reminder is displayed after every seven reminders
    - *Never:* an informational reminder is never displayed

d. FR4: Barks

Priority: ***LOW***

Description:

Users should be able to create and receive "Barks" (customized messages) from other users in their local area. These are messages sent out to every user within range that has their bark setting turned on.

Details:

- Barks must be labeled under a certain category such as adoption, safety, fun, etc.
- Users should be able to customize the distance threshold for receiving barks around them at the intervals 5, 10, 25, and 50 miles
- Users will have the option to like a bark, and each bark will display the current number of likes it has received
- Users will be able to filter the types of barks they see based on category and/or importance (5+ likes on a bark makes it important)
- There will be a section of the application that displays all barks within the past week, which can be ordered by category, number of likes, or recency

e. FR5: Activities

Priority: ***HIGH***

Description: Users should be able to log activities based on what they have done for or with their dog. These include things such as logging when a user feeds, walks, medicates, and bathes their dog. These will be predefined activities with the ability to add a comment to each upon logging an activity.

Details:

- Each activity logged will follow this example structure (time and date is based on current device settings)
    - March 2, 1:52pm: Fed Spot (*"Optional Comment Here"*)
- The activity log will show all activities logged within the past month, based on the current time and date settings of the device
- Activities can be filtered based on the predefined activity types (e.g. only show walking activities)

4. Non-Functional Requirements:

NFR 1: Information for dog breeds should be accurate.

- Should contain correct scientific data.

NFR 2: The app should keep profile data isolated from other profile data.

- Profile data should not be viewable from other profiles.

NFR 3: The app should make efficient use of battery life.

- Should not be actively running to display notifications.

5. Hardware Requirements:

- Android device with Android 6.0 (Marshmellow) or greater
- Network communication for "Bark" feature

6. Design Specifications:

a. Components

Main Menu - Show list of profiles and option to create a new profile.

- New Profile Button - Open new profile menu with blank template
- Previous Profile Selection - Opens the main menu for a previously saved profile

New Profile Menu - Show list of fields to be entered . Contains back button.

- Text Fields - Name, Date of Birth, Breed, Gender, Weight
- Finish Button - Adds profile to list of available profiles.

Profile Main Menu - Show buttons that will lead to the different functionalities of the app. Also contain a back button to select a different profile.

- Add New Activity Button - Opens up a list of fields where users can input reminder title, reminder frequency, and reminder optional message. Appears at the bottom of the list of reminders
- View Reminders Button - Takes user to a view of all reminders displayed in a list format
- View Barks Button - Takes the user to a view of all Barks within range within the last week. At the top of this view there are three buttons: Category (drop down list of categories which user will check or uncheck), Most Liked (will sort all currently shown barks by

number of likes), and Most Recent (will sort all currently shown barks by time of creation)

- View Info for Dog Button - Takes the user to a screen where information about their dog and their dog's breed are displayed.

View Activities Menu - Shows previously logged activities for the current profile. Contains Back Button.

Back Button - Switches the current screen to the previously displayed screen.

b. <u>System Operation</u>

Application Initialization - Application state data is loaded from the Android device's hard drive and the login screen is displayed. This occurs on application launch

New Profile Button Pressed - New profile object is initialized. This object contains variables which will store the different information stored by the user. Once the user has completed the form, this profile object is added to the app's list of profiles.

New Profile Finish Button Pressed - The current new profile object is added to the user object's list of profiles. This data is to remain persistent.

User Selects Profile - Previous profile state data is loaded. After this, a main menu is launched which will be linked to this profile.

Add New Activity Button Pressed - A new activity object is initialized. Once the user has completed filling the form, this object is added to the profile object's list of activities.

View Reminders Button Pressed - Get profile object's list of reminders. Then, create new View Reminders object which will be populated by the profile's list of reminders.

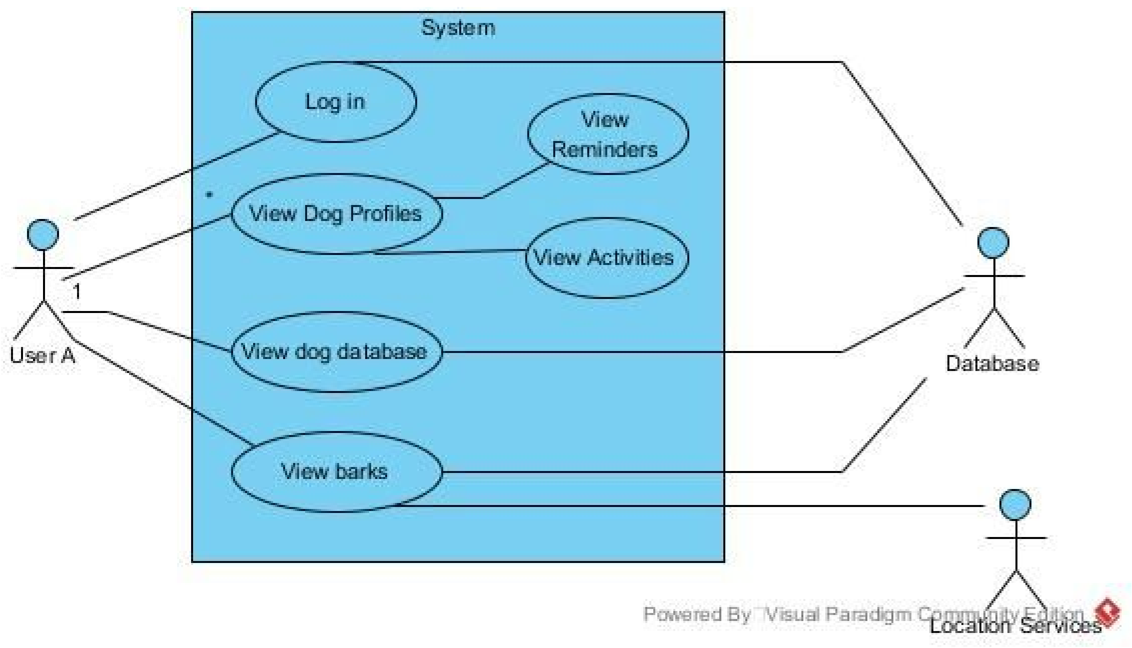
View Barks Button Pressed - Initialize a View Barks Object.

View Info for Dog Button Pressed - Initialize Info object. Then, get current profile's data that was entered by the user and add to the Info object. After this, if information is available for that specific breed, also add this data to the Info object.

Back Button Pressed - Close current screen and make previous screen active. Should be implemented as a stack.

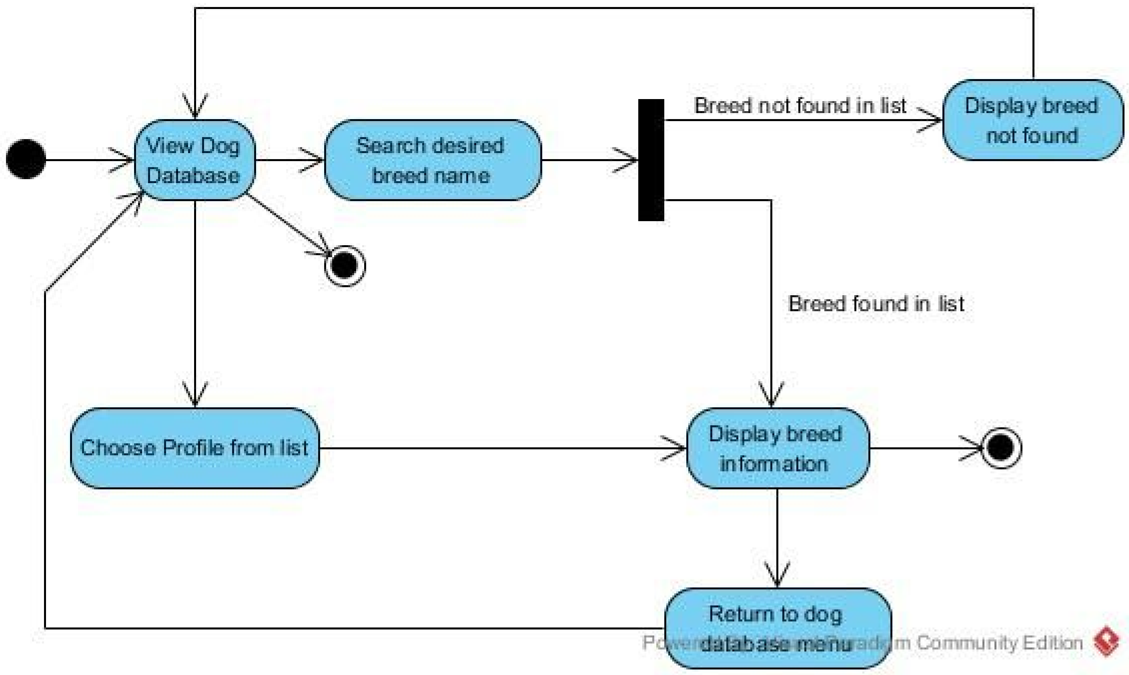
7. Analysis Models:

a. Use Case Diagram:



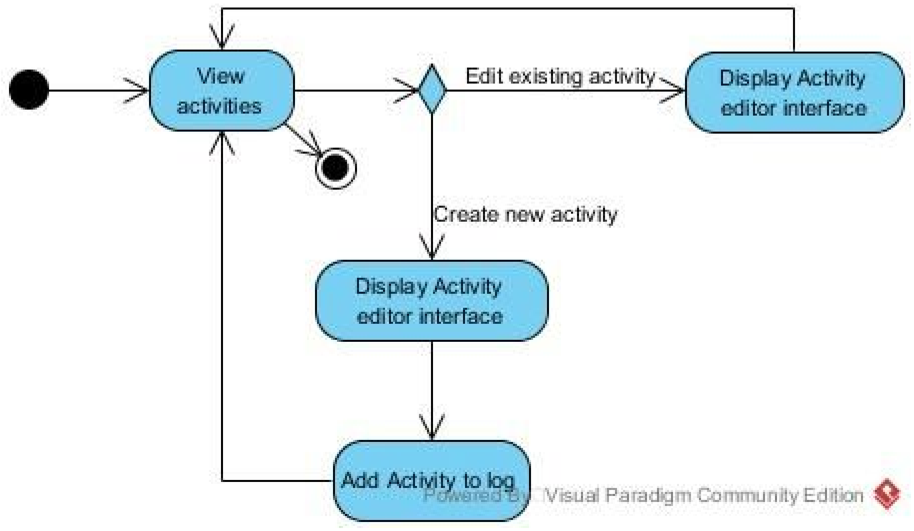
This diagram illustrates Man's Best Friend's use cases. From the main menu the user will be able to log in, view dog profiles, view the dog database or view local barks. Viewing dog profiles also includes two more use cases, view reminders and view activities. The main actor for these use cases is the user themselves, however the log in, view dog database, and view barks also interacts with the application's main database. Also, the view barks use case will interact with Google's location services in order to get the user's location and local barks.

b. Activity Diagram - Dog Database


View Dog Database
Search desired breed name
Breed not found in list
Display breed not found
Breed found in list
Choose Profile from list
Display breed information
Return to dog database menu
Powered By Visual Paradigm Community Edition

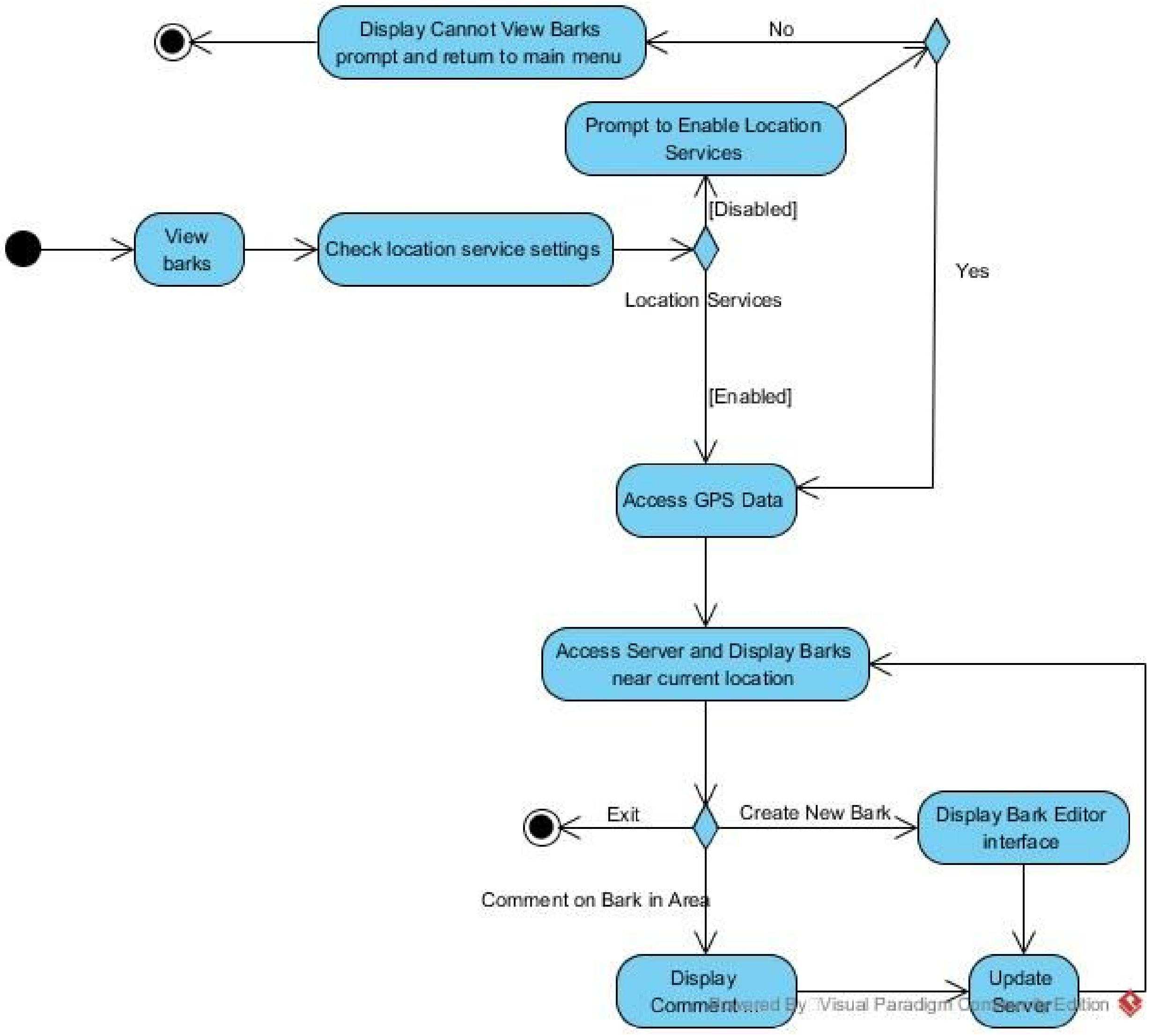
The Dog Database will be used to lookup information on a specific breed. The app will give the option to search for a specific breed. If this breed is not in the 50 supported breed list then the app will notify the user that the breed cannot be found. If the breed is found in the list the information for that breed will be displayed and the user will have the option to search for another breed.

c. Activity Diagram - Activities


View activities
Edit existing activity
Display Activity editor interface
Create new activity
Display Activity editor interface
Add Activity to log
Powered By Visual Paradigm Community Edition

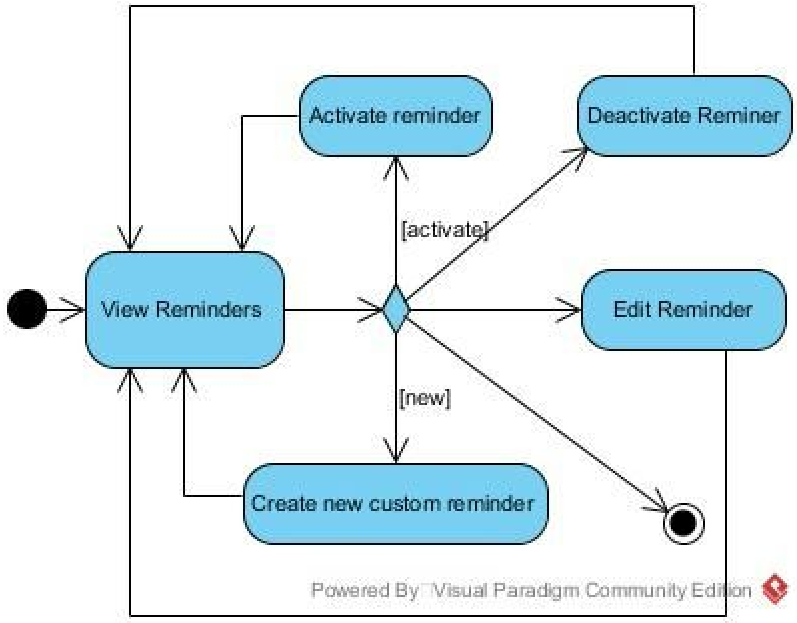
The user will be able to view activities. Once the activities are being viewed, the user will choose either to edit and existing activity or to create a new activity. The create new activity option will display the activity interface and then the activity will be added to the activity's log. Editing an existing activity will display the activity editor interface. Once the editing is completed the app will return to the view activities.

d. Activity Diagram - Barks


Display Cannot View Barks prompt and return to main menu
No
Prompt to Enable Location Services
[Disabled]
View barks
Check location service settings
Location Services
Yes
[Enabled]
Access GPS Data
Access Server and Display Barks near current location
Exit
Create New Bark
Display Bark Editor interface
Comment on Bark in Area
Display Comment...
Update Server
Powered By Visual Paradigm Community Edition

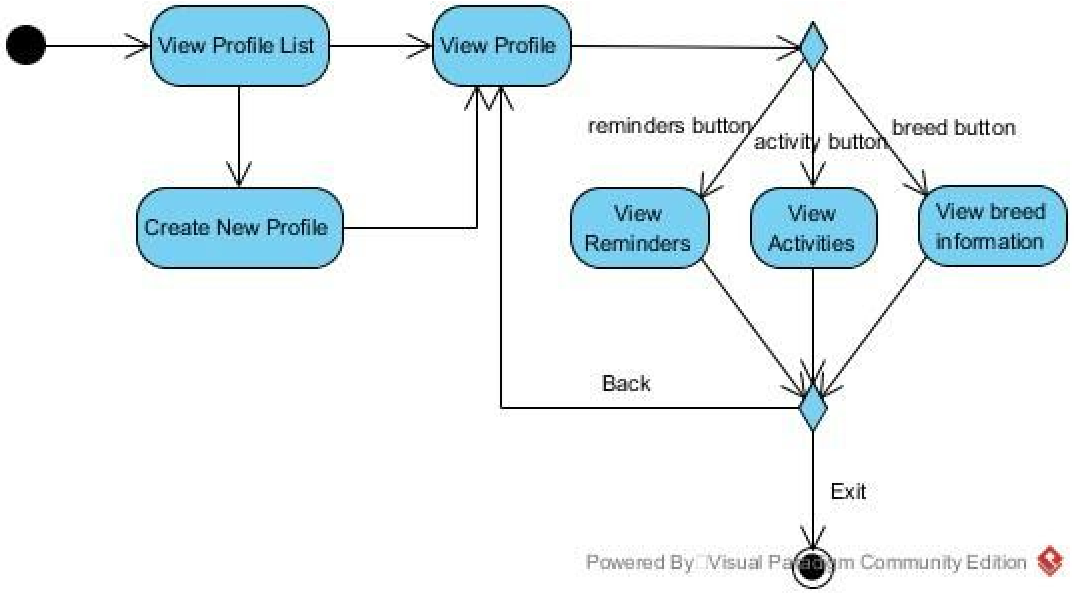
The user chooses to view barks from the main menu, at which point the app automatically requests the user's location (assuming it is enabled). Then, all barks with that location are retrieved from the database and displayed to the user. The user can then choose to create a new bark, which opens up the new bark interface (add text, type, etc.). Once the user is finished and submits the bark, it updates the database and is displayed to the user along with other preexisting barks.

e. Activity Diagram - Reminder:



On the view reminders screen the user will have the option to activate a reminder, deactivate a reminder, edit reminder, or create new reminder. Activating and creating new reminders will update the current active reminders list. Deactivating a reminder will remove it from the active reminders. Editing a reminder allows the user to edit the details of a specific reminder. If the user decides to not choose any of these options the app will return to the previous screen.

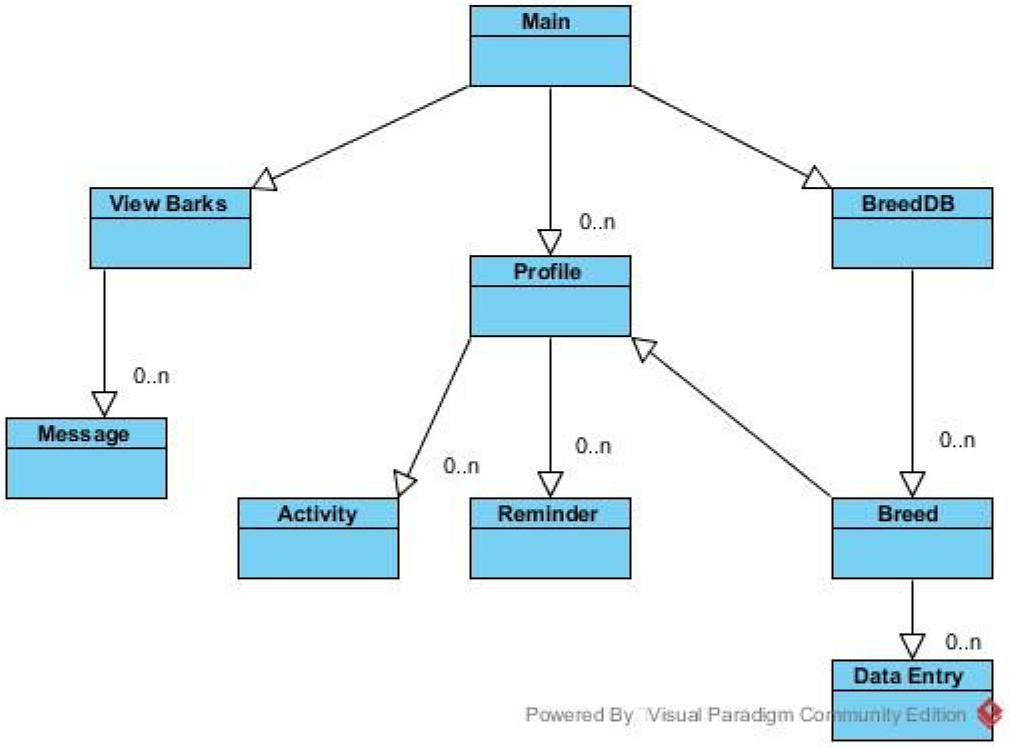
f. Activity Diagram - Profiles:



21

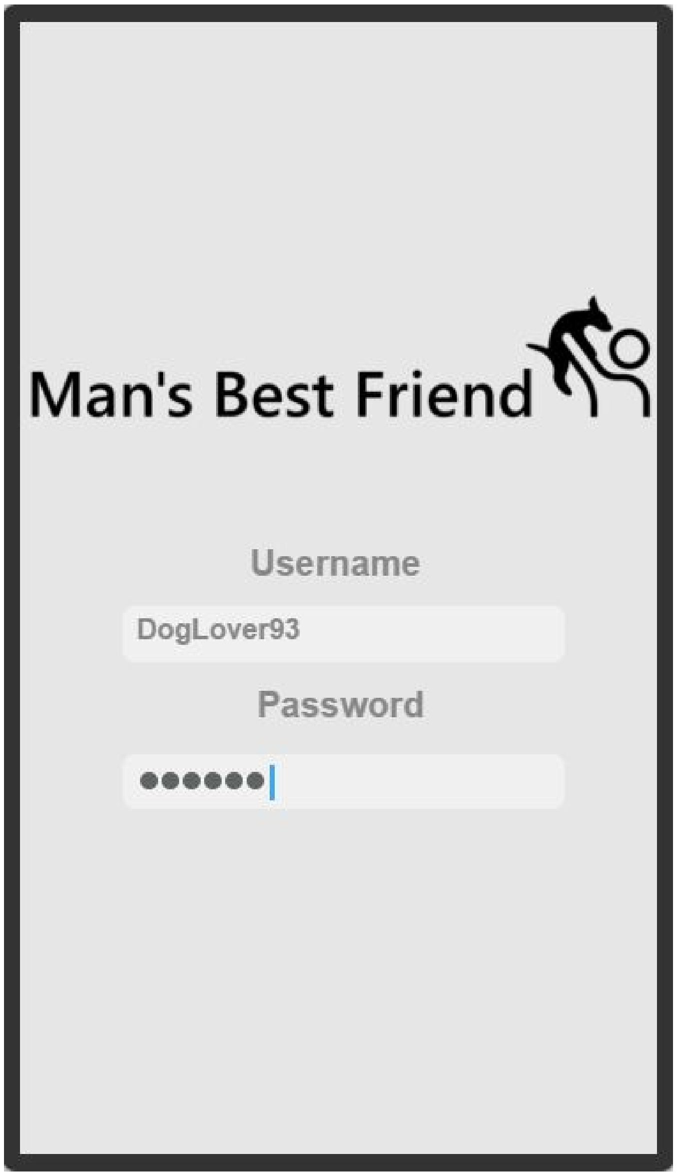
The view profile list will have the option of either viewing a profile or creating a new one. Creating a new profile will just initialize a Profile object with blank information and put the user at the view profile menu. At this menu the user can either view reminders, view activities, or view the information on that specific profile's breed. Each of those options will execute a different activity and will return to the main profile menu when finished.

g. High Level Class Diagram:

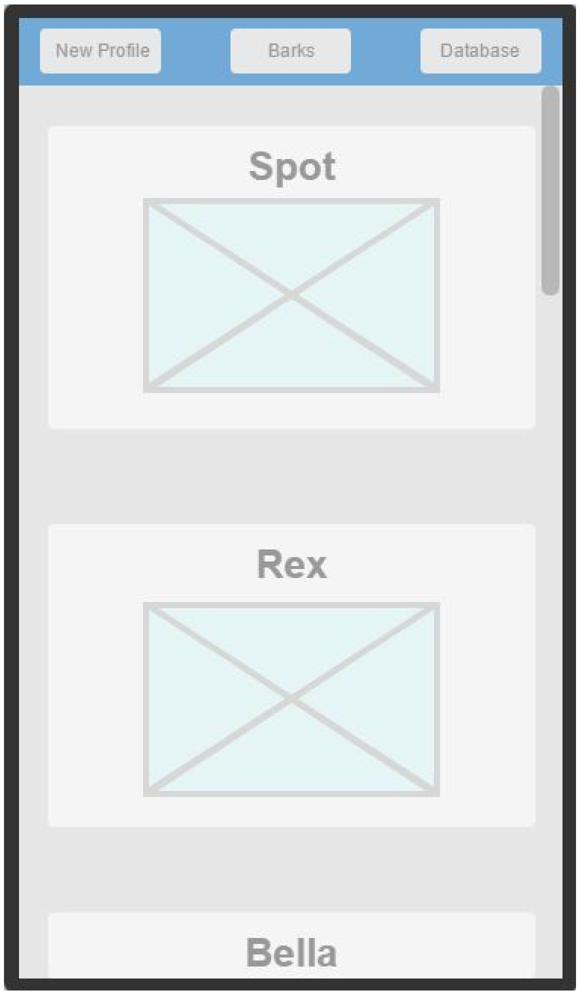


8. UI Layout:

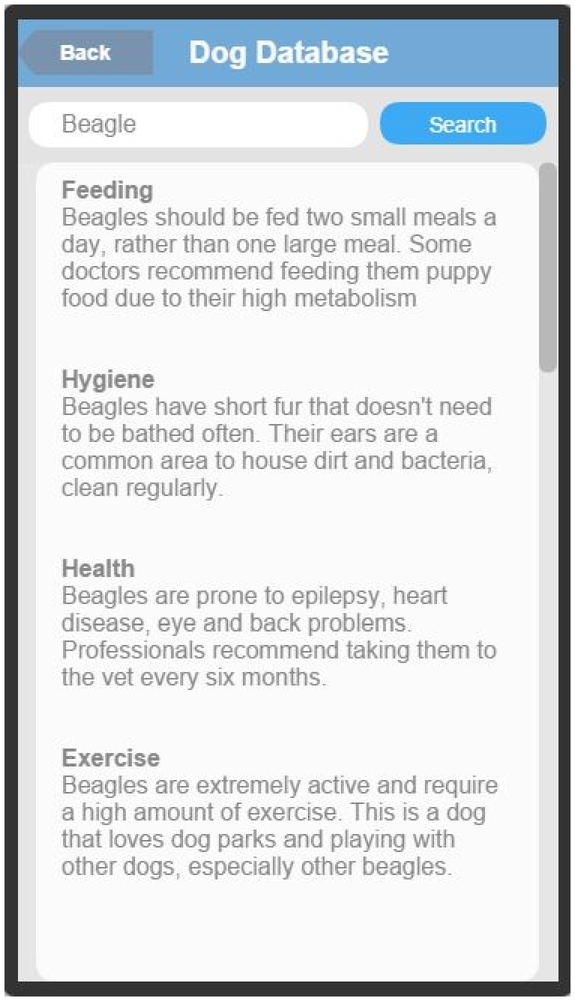
Login Screen:



Profile Selection Screen:



Dog Database Screen:



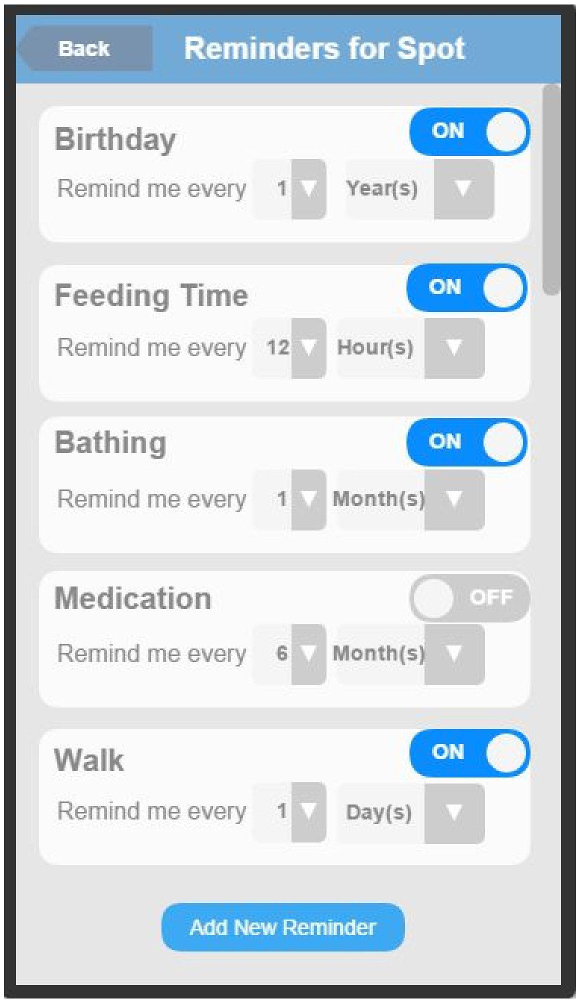
Feeding
Beagles should be fed two small meals a day, rather than one large meal. Some doctors recommend feeding them puppy food due to their high metabolism

Hygiene
Beagles have short fur that doesn't need to be bathed often. Their ears are a common area to house dirt and bacteria, clean regularly.

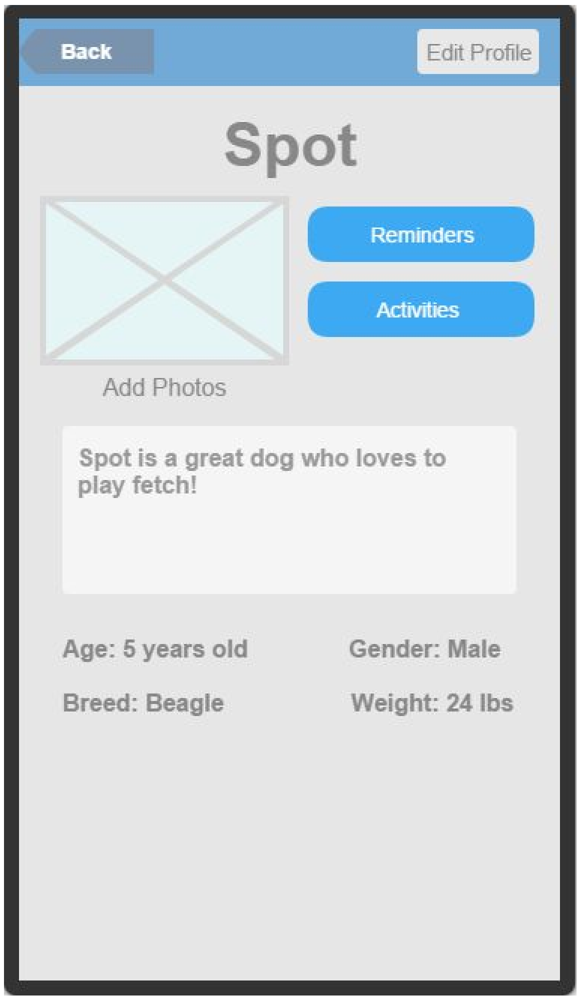
Health
Beagles are prone to epilepsy, heart disease, eye and back problems. Professionals recommend taking them to the vet every six months.

Exercise
Beagles are extremely active and require a high amount of exercise. This is a dog that loves dog parks and playing with other dogs, especially other beagles.

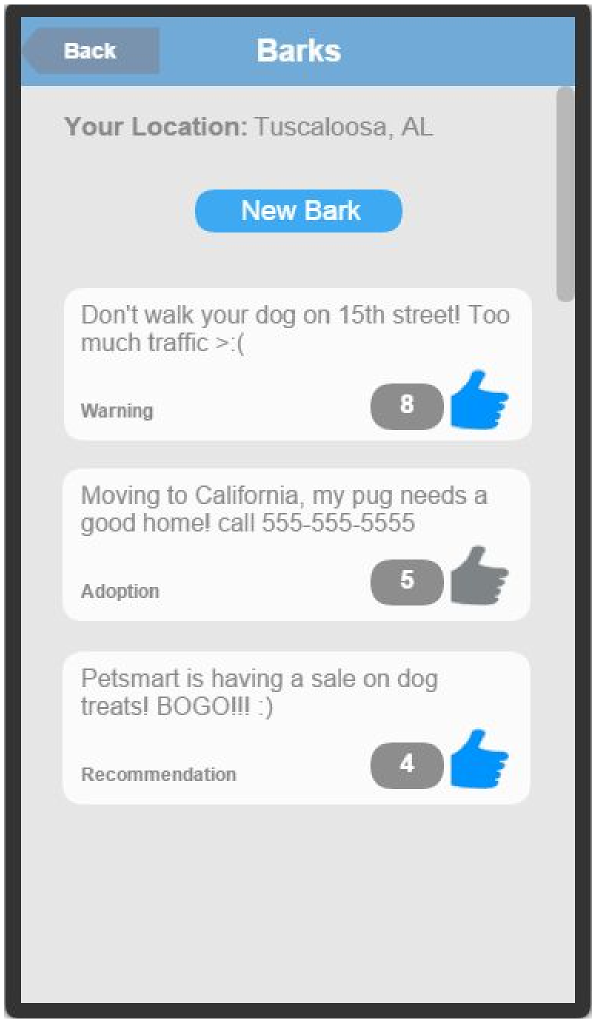
Reminder Screen:



Profile Screen:



Barks Screen:



Your Location: Tuscaloosa, AL

Don't walk your dog on 15th street! Too much traffic >:(

Warning 8

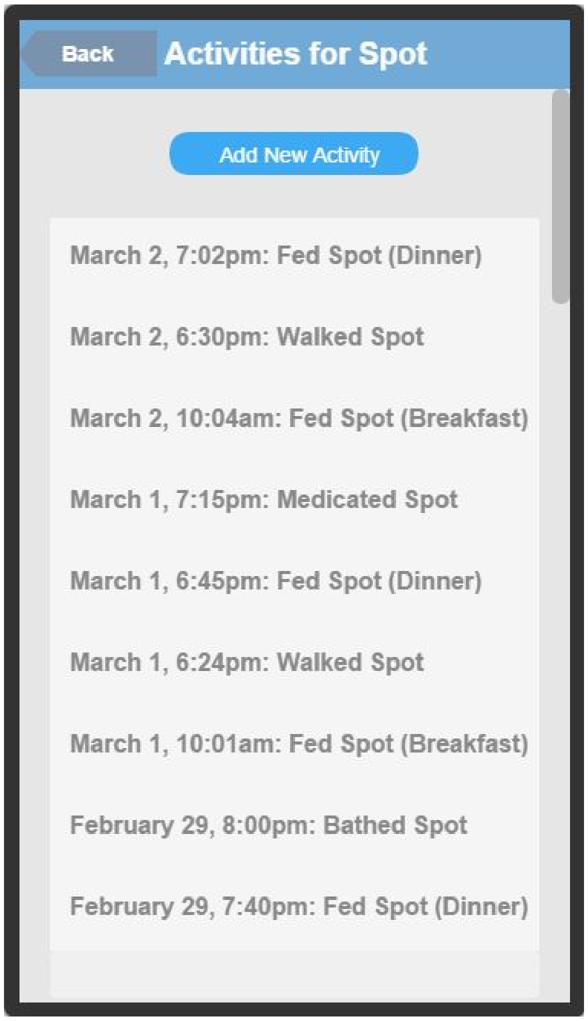
Moving to California, my pug needs a good home! call 555-555-5555

Adoption 5

Petsmart is having a sale on dog treats! BOGO!!! :)

Recommendation 4

Activities Screen



March 2, 7:02pm: Fed Spot (Dinner)

March 2, 6:30pm: Walked Spot

March 2, 10:04am: Fed Spot (Breakfast)

March 1, 7:15pm: Medicated Spot

March 1, 6:45pm: Fed Spot (Dinner)

March 1, 6:24pm: Walked Spot

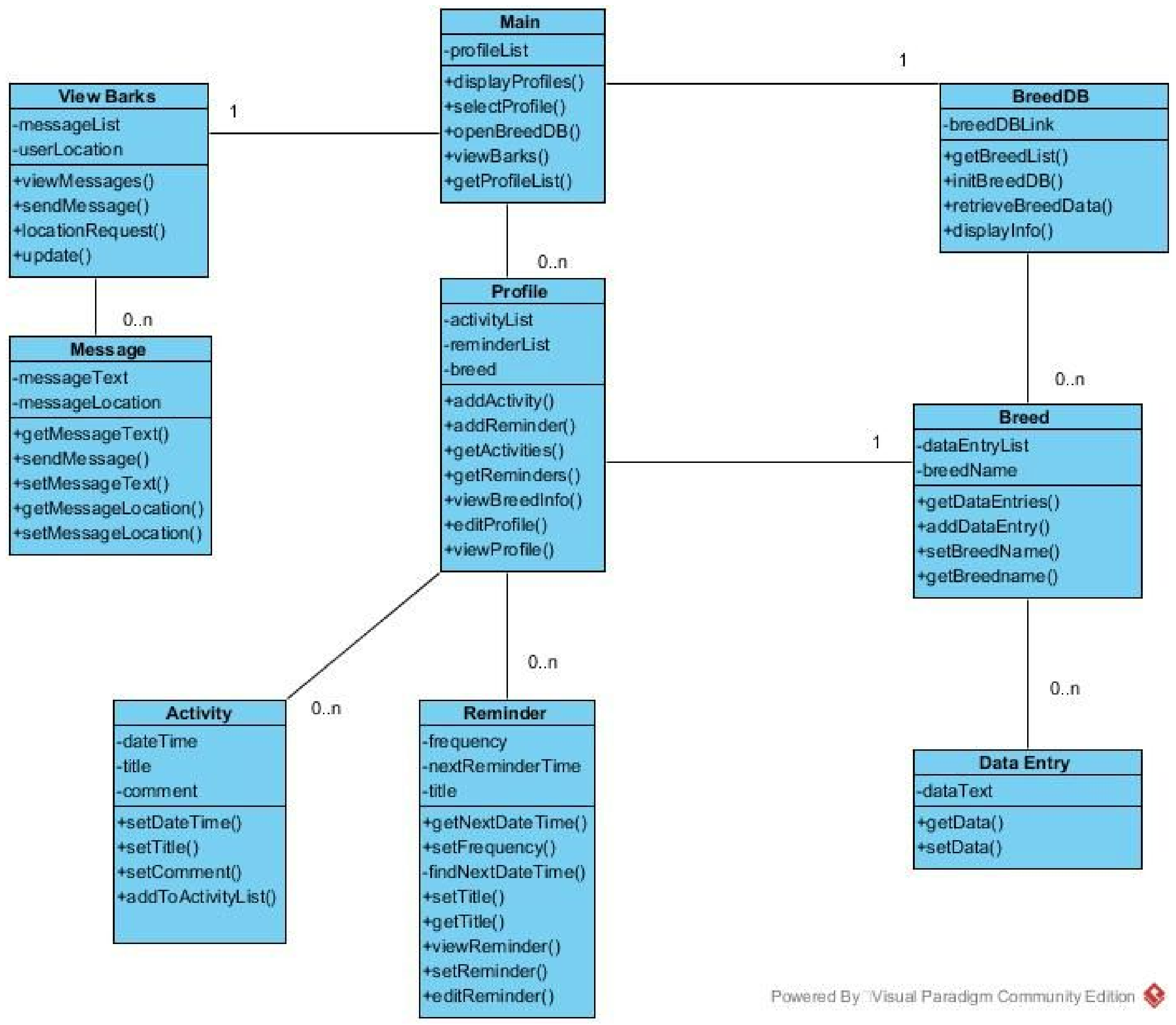
March 1, 10:01am: Fed Spot (Breakfast)

February 29, 8:00pm: Bathed Spot

February 29, 7:40pm: Fed Spot (Dinner)

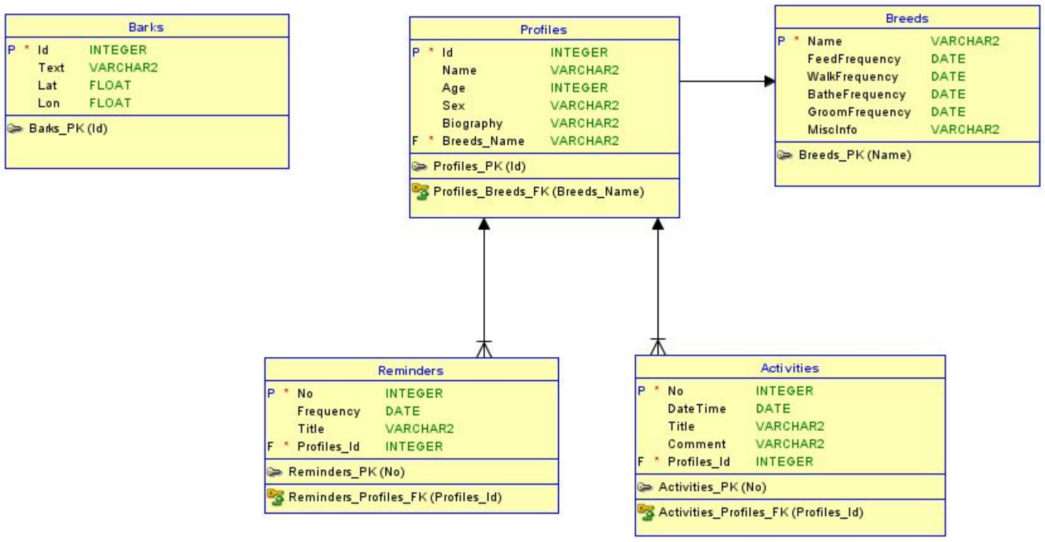
9. Design Models:

a. Detailed Class Diagram:



This diagram describes the detailed class diagram used to structure Man's Best Friend software. The main entry point for the application is the Main class, from which functions can be called to show all profiles, select a profile, view local barks, and interact with the dog database. Although the Main class can include multiple Profile classes, there is ever only one active Profile. Each Profile has functions for editing the profile, adding and Activity, adding a Reminder, and viewing the corresponding dog's breed information. Activities and Reminders can also be retrieved from the Profile class. Each Activity class has functions to set and get the title and date-time informations as well as comments. The reminder class is similar although the next date-time is not set from an external method call. Instead the next date-time for the reminder is found using the current date-time and the frequency set by setFrequency. Reminders can also be edited using the editReminder function. The BreedDB class will act as an interface with the SQLite database used to keep data persistent. From this class the initBreedDB and retrieveBreedData methods are used to interact with the database. Each specific breed in the database has its own corresponding class. This class, Breed, will contain the data specific for that breed. The Data Entry class represents any form of data to be entered in the Breed class. This data is manipulated using the setData and getData methods. Finally, the View Barks class is the main entry points for viewing local barks. The locationRequest method will get the current user's location which will be used to populate the barks list. Each bark is represented in the Message class, which contains the message's text and location. The functions setMessageText and setMessageLocation are used to set the text and location of the message respectively. The sendMessage function is used to send the message to the database of barks.

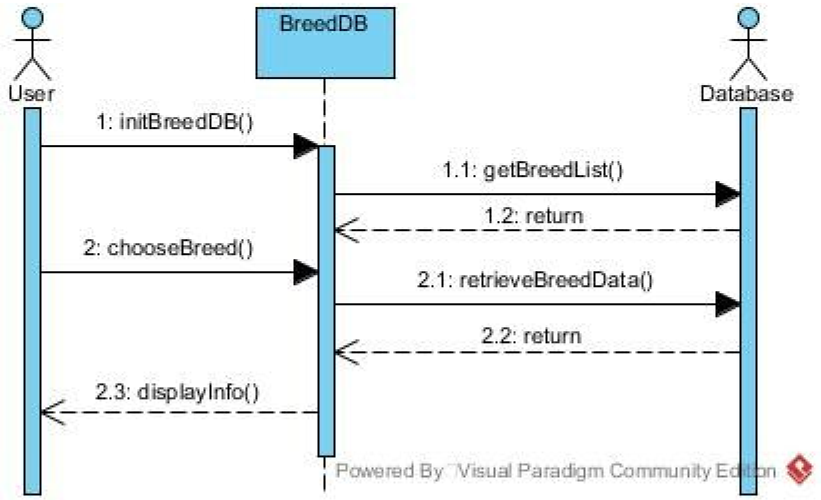
b. Database Diagram:



The information stored in the SQLite database will help keep user data persistent after the app has been closed. The profiles table will be used to map profile data to specific dogs. Within this table the dogs information is stored in the Name, Age, Sex, Biography and

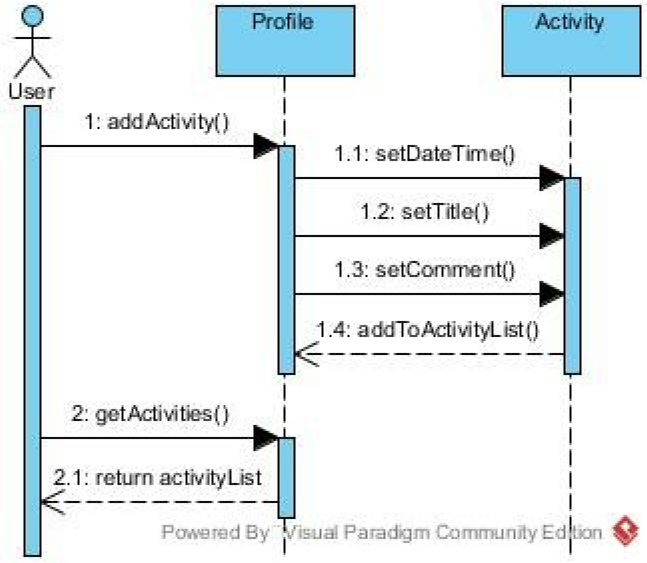
Breeds_Name. In this table Breeds_Name will be used to lookup a specific dog breeds data from the Breeds table. The Breeds table will contain data for each specific breed to act as a suggestion for certain reminders. These reminders include FeedFrequency, WalkFrequency, Bathe Frequency, etc… This table also allows miscellaneous information to be stored. The Reminders and Activities tables will contain the data for the main functionalities of this app. Reminders will have a Title and Frequency which will be used to calculate the next reminder time. Activities will act as a log for tracking progress. Both of these tables contain the Id of the profile for which each is associated with. This will be used to map Reminders and Activities to specific Profiles. Finally the Barks table is essentially used to keep track of Bark locations when they are sent. This table will need to be located on a server in order for any user to interact and view barks in their local area.

c. Sequence diagram - Dog Database



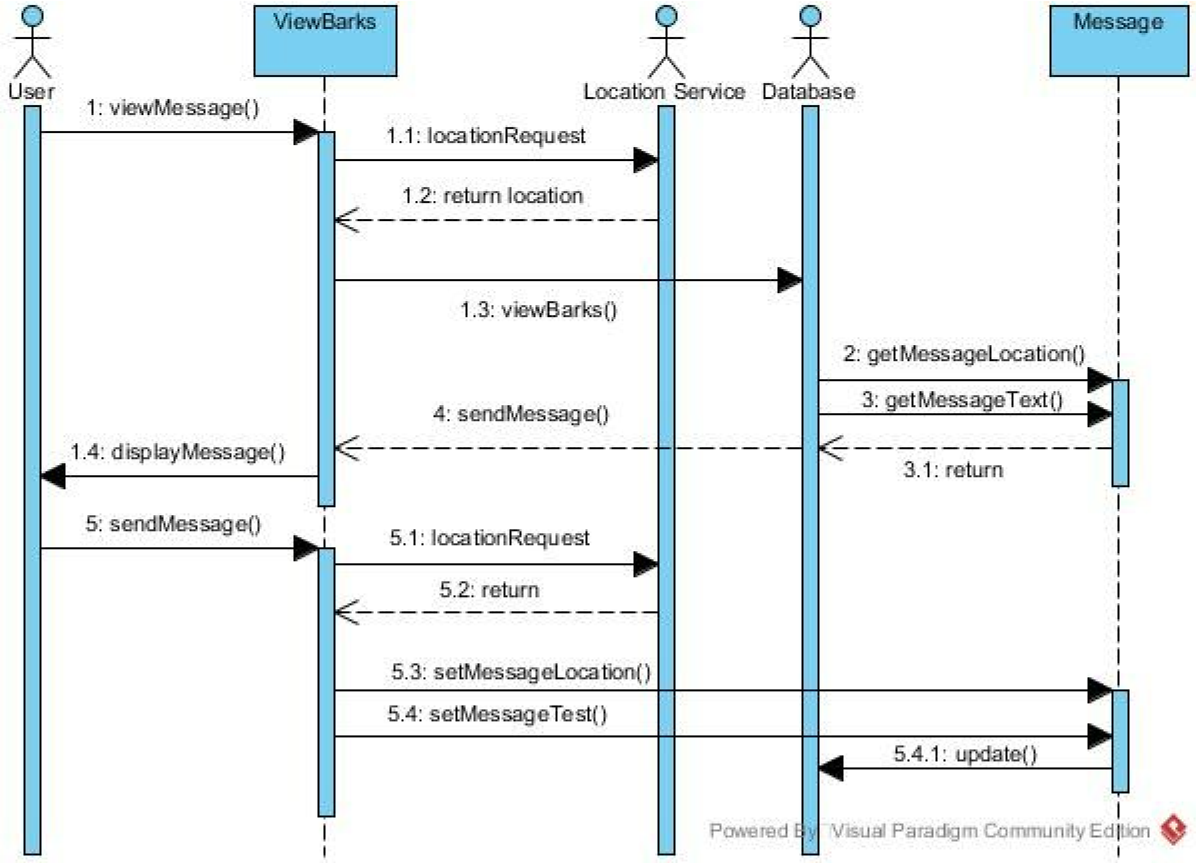
The user initializes the Dog Breed DB by selecting it from the main profile list menu. Upon selection, the user will be taken to a screen that will display a list of all available dog breeds, which will be retrieved from the database. Once the user chooses the breed they want to research, the data on that specific breed will be retrieved from the database and displayed to the user.

d. Sequence Diagram - Activity



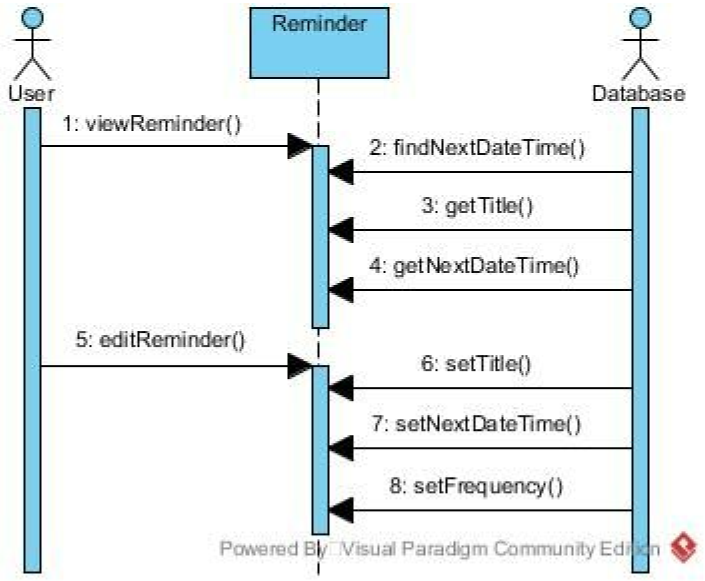
The activity screen consists of a list of previously performed activities and a button to add a new activity. The list is displayed by getting the activity list from the Profile class. When a user chooses to add a new activity, they set the date and time, title, and comment for the activity, which is then added to the list of all activities.

e. Sequence Diagram - Barks



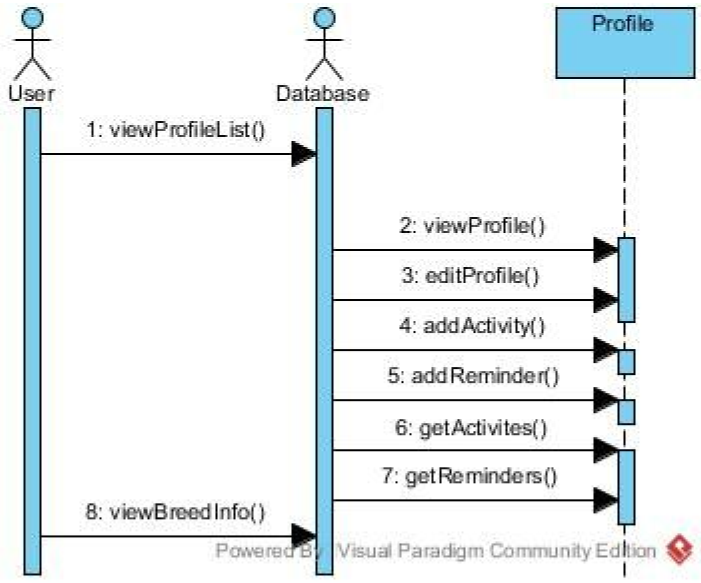
The ViewBarks class is initialized by calling the viewMessage() function. A location request is then automatically performed (after being pre-approved by the user) and returned to viewBarks. This location is used to retrieve all with that same location from the database. This is done by retrieving the location and text for each bark, returning it to viewBarks, and then displaying it to the user. The user can also add their own barks to the database. This is done by once again requesting their location, and then using that data along with text the user provides, and then updating the database by adding this new bark to the database.

f. Sequence Diagram - Reminder



After the reminder interface is prompted by the user, the viewReminder() method is called from Reminder and this then retrieves the next reminder with findNextDateTime() and then gets that reminders information with getTitle() and getNextDateTime(). To edit a reminder, the setTitle() is called with the parameter of title that the user provides. User also sets the DateTime and the frequency of the reminder with those respective functions.

g. Sequence Diagram - Profile



After the user opens the profile list, the data is then retrieved from the database and the user can then view individual profiles, edit them, add an activity, and add a reminder. The activities and reminders are gotten from the database when those requests are made. The user can also view breed info on the profile they have open.