PawPrints Pet Tracker

Hoban's Heroes

1

Glossary

- **PawPrints**
  - Pet tracker system including the **smartphone application**, **backend** and **database**.
- **Conan**
  - Pet tracker **hardware** that is to be **attached to a pet**.
- **LE**
  - Low Energy
- **Pet Owner**
  - Refers to the individuals responsible for caring for the pets and any assigned **users** that may be **associated** with a **pet's profile**.
- **Ping**
  - Pinging will refer to a user **device** receiving a **broadcasted hardware ID** from a **Conan device**

2

System Overview

A **pet tracker system** to track lost pets and reunite them with their owners

- Hardware:
  - A **device** attached to **pet's collar**
  - **Broadcasts** hardware ID every minute
- Software:
  - **Android application** listening for ID broadcasts
  - **Alerts** pet owners and user if pinged by a lost pet

Inexpensive, convenient and reliable tool promoting community involvement

3

Motivation

- Other pet recovery methods may be **costly**, **time consuming**, and **ineffective**
- A more **proactive** approach that promotes **community assistance**
- A **complementary** tool **not** a replacement

4

Implementation Plan

- Arduino 101 hardware with bluetooth low-energy module
  - Arduino IDE 1.8.7
  - AT commands
  - CurieBLE library
- Android Studio
  - Java 8 and XML
  - Firebase SDK
  - JavaScript

5

Implementation Plan - Testing

- Hardware Testing
  - Range of functionality in different environments
  - Timing of broadcast
  - Power consumption
- UI Testing Framework
  - Espresso
- Unit Testing Frameworks
  - JUnit
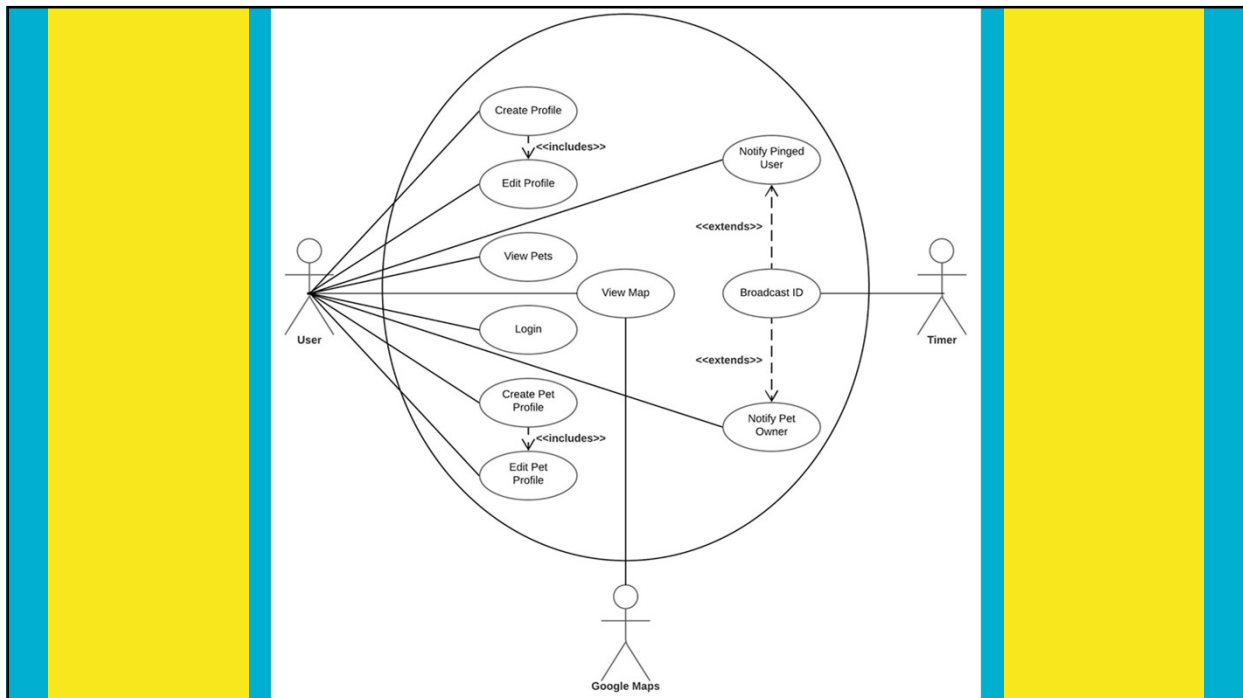 - Mocha for backend

6

Concerns

- Security
  - Pets can only be "tracked" by **owners**
  - **Firebase** authentication for **user profile** service
- Privacy
  - **Location** data from pings will **not** be posted with **user profile**
- Hardware
  - Power consumption of **Conan** and **device** are minimal with **BLE**
  - There is established hardware for other similar products

7

Use Case Diagram

8

9

Profile Management

Users will be able to

- Create an account

- Add pets to their account

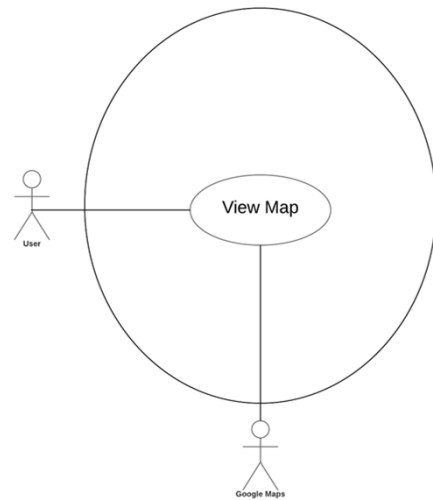- Manage their account information
- Manage their pet information

10

Tracking

Allows the **pet owner** to

- **View** real-time map with colored pins representing **alerts** created by **pinged users**

- **Select** a pet to **view alerts** associated with the pet's **Conan**
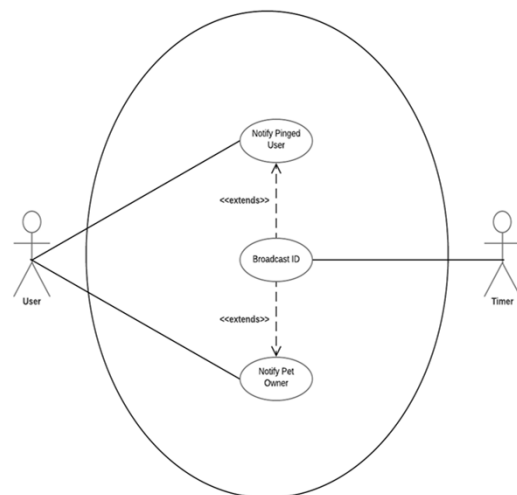
View Map

User

Google Maps

11

Alerts

If a pet status is **lost**

- The **application** will **notify** the **owner** of the pet

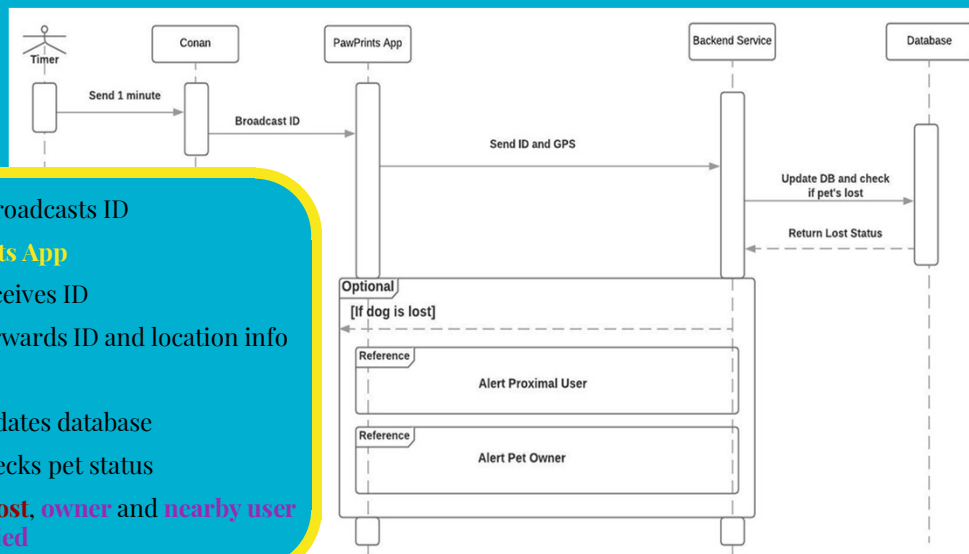- The **application** will also **notify** the **pinged user**

Notify Pinged User

<<extends>>

Broadcast ID

<<extends>>

Notify Pet Owner

User

Timer

12

Sequence Diagrams

13
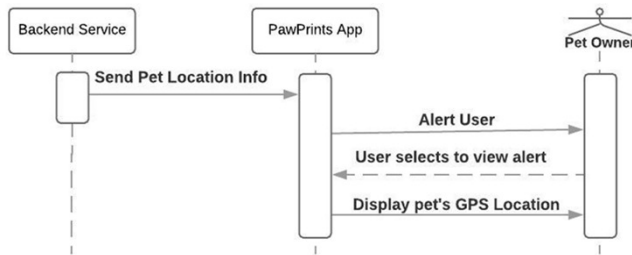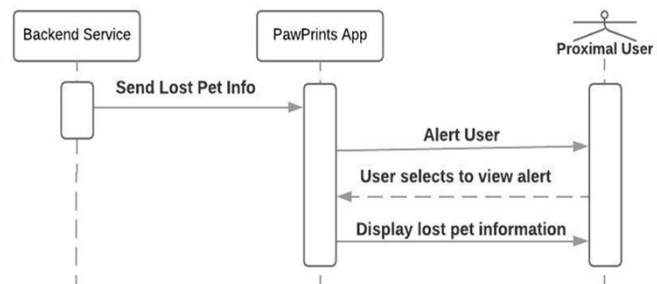
Sequence Diagram: Ping and Alert

- **Conan** broadcasts ID
- **PawPrints App**
  - Receives ID
  - Forwards ID and location info
- **Backend**
  - Updates database
  - Checks pet status
- If pet is **lost**, **owner** and **nearby user** are **notified**

14

Nearby User

- **Application** will send **lost** pet info to the **nearby user**
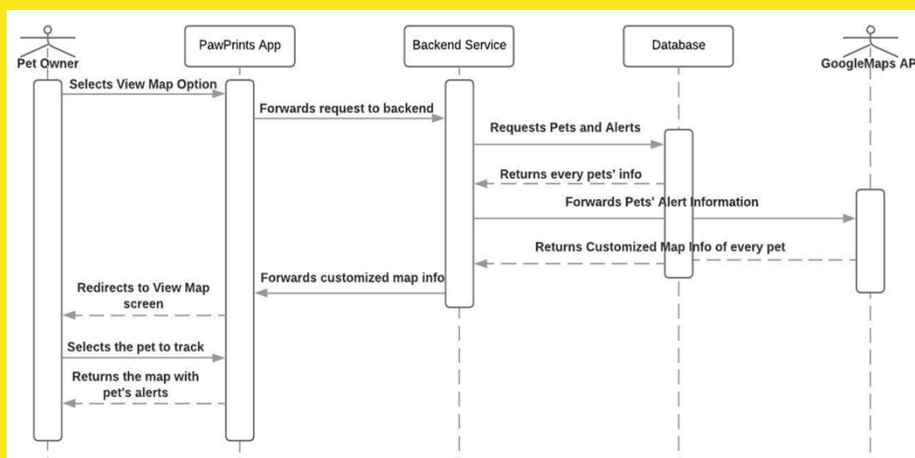- **User** will be able to **view** the **alert**

Pet Owner

- **Application** will send **coordinates** of where the pet was **pinged**
- **User** can select to **view** the **GPS location**

15

Sequence Diagram: View Map

The user selects to **view map** which calls **backend** services to determine pets associated with the user, gathers the **alert data** and accesses the **Google Maps API** to return a **customized map** for the user.
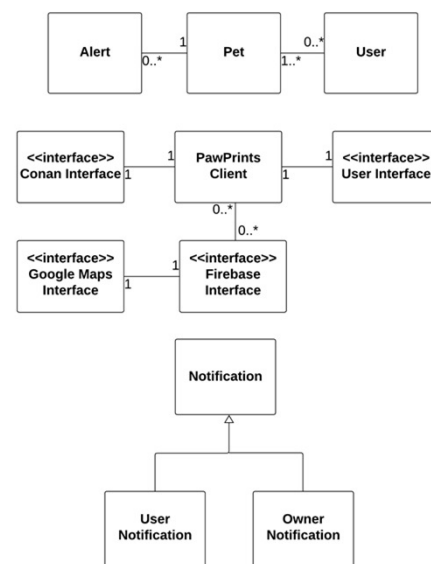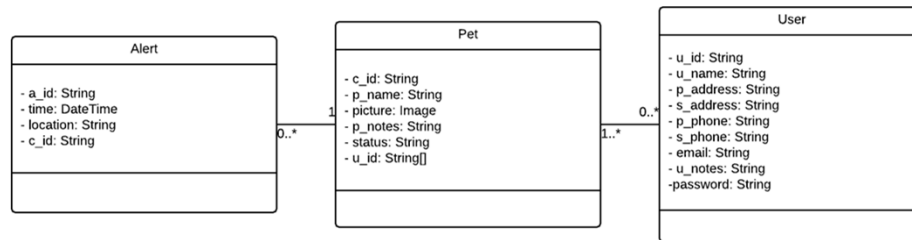
16

Class Diagram

17

Class Diagram Overview

- Three Main Parts:
  - Entity Classes
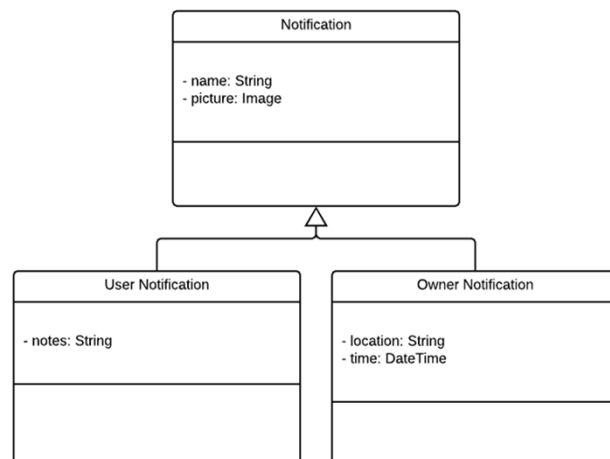 - Notifications
  - Interfaces

18

Entity Classes

- Alert – Location data of pet
- Pet – Information related to pet associated with Conan and owner ID
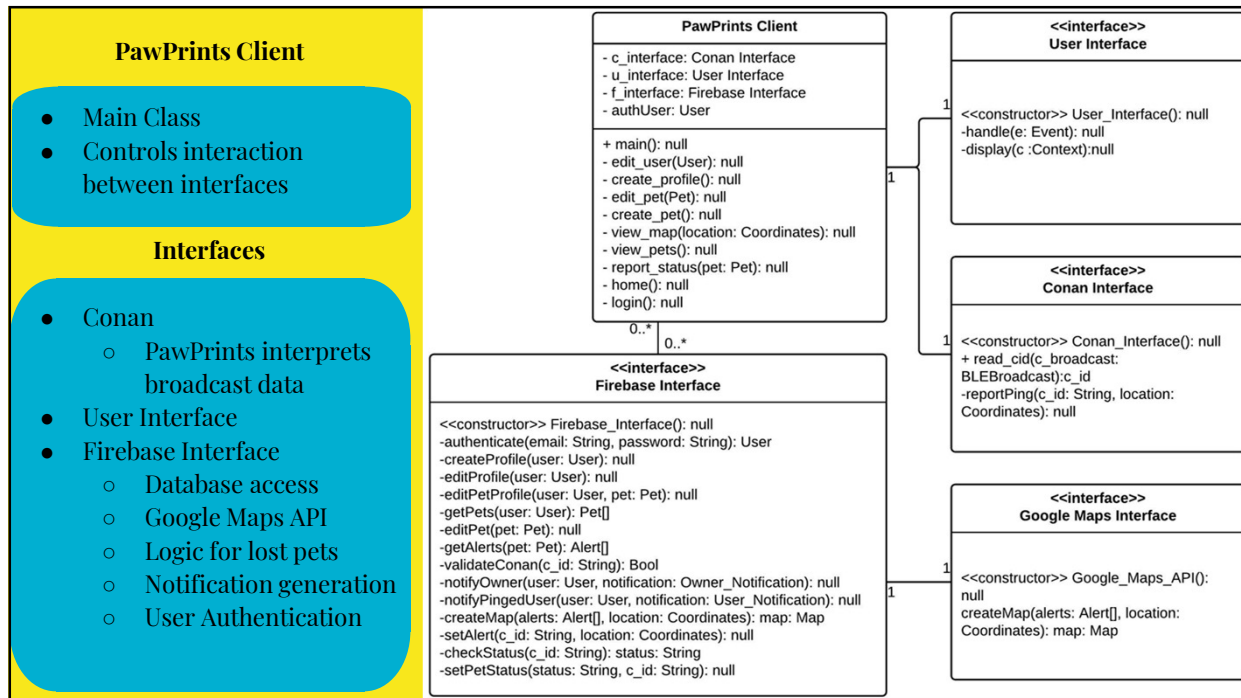- User – User profile information contains contact information

19

Notifications

Objects generated by the backend to provide appropriate information to users

- User Notification
  - Contains image of pet, pet's name, owner contact info, and notes to help with pet recovery
- Owner Notification
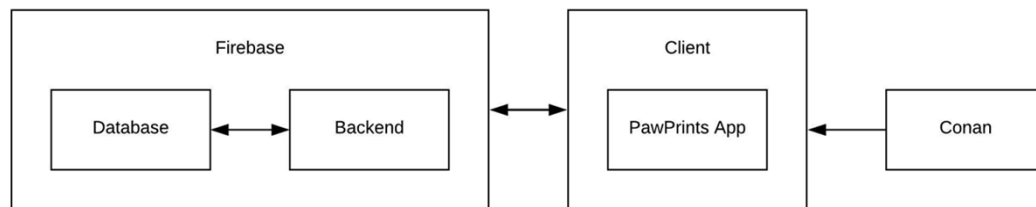 - Contains ID of pet, timestamp, and location of pet

20

PawPrints Client

- Main Class
- Controls interaction between interfaces

Interfaces

- Conan
  - PawPrints interprets broadcast data
- User Interface
- Firebase Interface
  - Database access
  - Google Maps API
  - Logic for lost pets
  - Notification generation
  - User Authentication

PawPrints Client

- c_interface: Conan Interface
- u_interface: User Interface
- f_interface: Firebase Interface
- authUser: User

+ main(): null
- edit_user(User): null
- create_profile(): null
- edit_pet(Pet): null
- create_pet(): null
- view_map(location: Coordinates): null
- view_pets(): null
- report_status(pet: Pet): null
- home(): null
- login(): null

**<<interface>>
User Interface**

<<constructor>> User_Interface(): null
-handle(e: Event): null
-display(c :Context):null

**<<interface>>
Conan Interface**

<<constructor>> Conan_Interface(): null
+ read_cid(c_broadcast: BLEBroadcast):c_id
-reportPing(c_id: String, location: Coordinates): null

**<<interface>>
Firebase Interface**

<<constructor>> Firebase_Interface(): null
-authenticate(email: String, password: String): User
-createProfile(user: User): null
-editProfile(user: User): null
-editPetProfile(user: User, pet: Pet): null
-getPets(user: User): Pet[]
-editPet(pet: Pet): null
-getAlerts(pet: Pet): Alert[]
-validateConan(c_id: String): Bool
-notifyOwner(user: User, notification: Owner_Notification): null
-notifyPingedUser(user: User, notification: User_Notification): null
-createMap(alerts: Alert[], location: Coordinates): map: Map
-setAlert(c_id: String, location: Coordinates): null
-checkStatus(c_id: String): status: String
-setPetStatus(status: String, c_id: String): null

**<<interface>>
Google Maps Interface**

<<constructor>> Google_Maps_API(): null
createMap(alerts: Alert[], location: Coordinates): map: Map
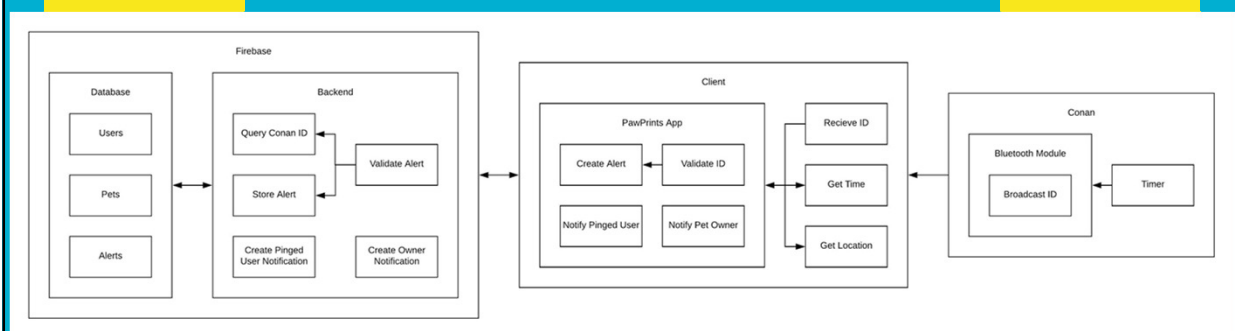
21

System Architecture

22

Overview

- **Conan communicates** with **PawPrints app** through client bluetooth module
- **Google Firebase** provides **platform** for both **database** and **backend** services
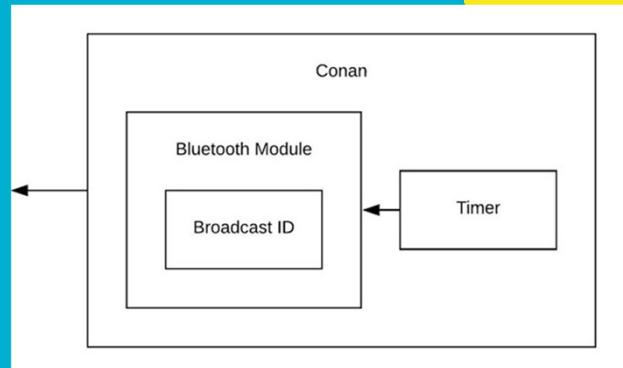
23

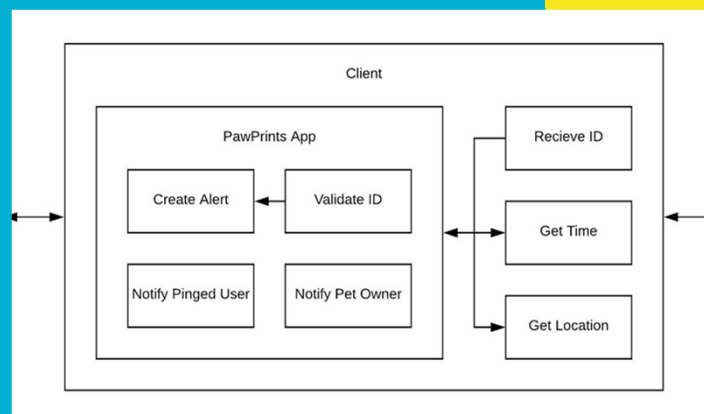# Ping and Alert

24

Ping and Alert Conan

- **Timer** on **Conan** arduino **triggers** broadcast ID use case
- **Bluetooth module** responsible for **broadcasting** hardcoded **Conan ID**


25

Ping and Alert Client

- Client **receives** ID
- **PawPrints app validates** and creates the **alert**
  - Retrieves **time** and **location** through **Android OS**
- **PawPrints app notifies** users when **prompted** by **backend**

26

Ping and Alert Firebase

- **Backend** validates **alerts received** from **PawPrints app**
- **Backend** stores **alert** and queries **pet info**
- If pet is marked **lost** the **backend** creates **notifications**
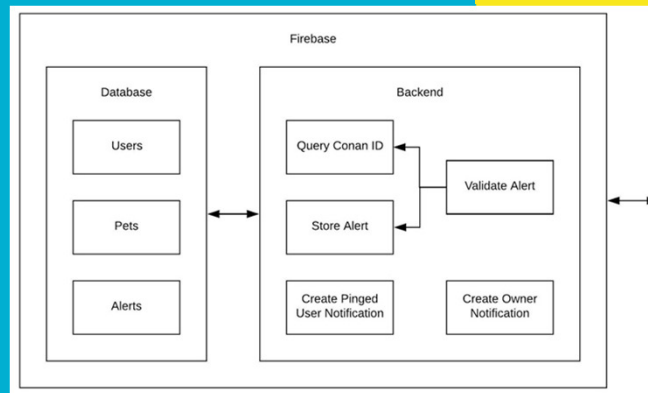
27

Lo-Fi Mock Up

28

Login

- User inputs username and password

- Password field hidden

- Click "Register" to start Create User Profile use case

29

Home

- Click profile picture
    - Edit profile
    - Log out
- Click "Pets" to view and edit pets

- Click "Map" for view map use case

- Click "Help" to display app tips

30

Pets

- Pet color backgrounds show status
  - Red indicates lost
  - Green indicates safe
- Warning button marks a pet as lost
- Click pet to edit pet profile
- Button to add new pets

31

Map
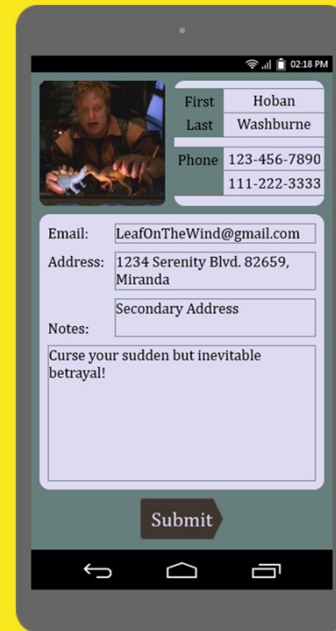
- Circle represents user location
- Alert symbol marks pet ping locations
  - Red for recent pings
  - Blue for old pings
- Drop down menu to switch between pets
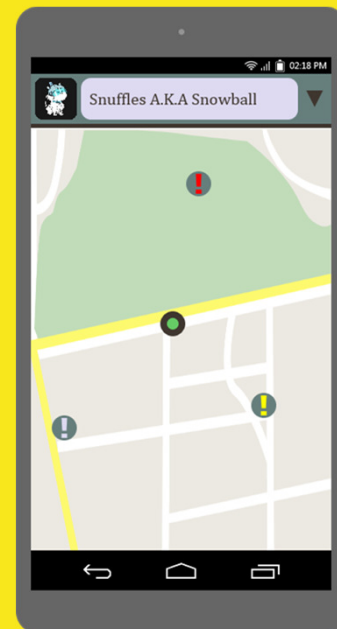
32

User Profile

- First Name
- Last Name
- Primary Phone Number
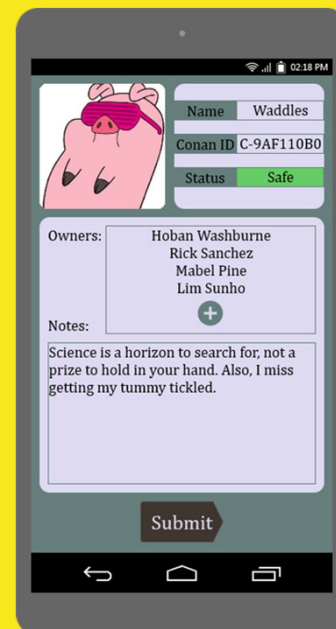- Secondary Phone Number
- Email
- Primary Address
- Secondary Address
- Notes

33

Pet Profile

- Name
- Associated Conan ID
- Status
- Owners
  - Other Paw Prints Users
- Notes
  - Info on pet for safe recovery

34

Questions

35

Pet Pictures

Send us cute pictures of your pets at

PawPrintsPetTracker@gmail.com

36