
# Sniffer

Missing Pet Tracker

**Prepared by:**

Lian Attily (20190133)

Qosay Al-Shatel (20190668)

**Supervised by:**

Dr. Firas Alghanim

Project Submitted in partial fulfillment for the degree of Bachelor of Science in Software Engineering Spring 2022-2023.

## Acknowledgement

We would like to express my deepest gratitude to everyone who has supported us throughout the journey of completing this graduation project.

First and foremost, we would like to thank our supervisor, Dr. Firas Al Ghanim, for his guidance, patience, and invaluable insights throughout the entire process.
We would also like to thank our family and friends (and pet owners) for their unwavering support and encouragement. Their belief in us has been a constant source of inspiration and motivation.

This project would not have been possible without the support and guidance of all of you. Thank you.

## List of Abbreviation

**GPS:** Global Positioning System
**Geofence:** Geographical Fence
**UML:** Unified Modeling Language
**SRS:** Software Requirements Specification
**WBS:** Work Breakdown Structure
**UREQ:** User Requirement
**FREQ:** Functional Requirement
**NF-REQ:** Non-Functional Requirement
**SDLC:** Software Development Life Cycle
**FCM:** Firebase Cloud Messaging
**API:** Application Programming Interface

# Table of Contents


Chapter 1: Introduction ..... 9
1.1. Overview ..... 10
1.2. Problem Statement ..... 10
1.3. Evaluation of Existing Systems ..... 10
1.3.1. Gap Analysis ..... 12
1.4. Commonly used methods ..... 13
1.4.1. Facebook groups and social media platforms ..... 13
1.4.2. Airtag ..... 13
1.4.3. Pet Microchips ..... 14
1.5. Objectives ..... 14
1.6. Stakeholder analysis ..... 14
Chapter 2: Project Management Plan ..... 16
2. Chapter 2: Project Management Plan ..... 17
2.1. Project Charter ..... 17
2.2. Software Development Approach ..... 18
2.3. Scope Statement ..... 18
2.4. Work Breakdown Structure (WBS) ..... 19
2.5. Project Tasks ..... 21
2.6. Gantt Chart ..... 1
2.7. RACI Matrix ..... 1
2.8. Risk Management ..... 1
2.8.1. Risk Source ..... 1
2.8.2. Risk Identification ..... 2
2.8.3. Risk Metric ..... 3
2.8.4. Risk Analysis ..... 4
2.8.5. Risk Response ..... 5
2.9. Software Development Lifecycle ..... 6
Chapter 3: Software Requirements Specifications ..... 8
3.1. Stakeholders ..... 9
3.2. Power-Interest Matrix ..... 9
3.3. Requirements Elicitation ..... 10

3.3.1. Survey ..... 10
3.3.2. Survey Results ..... 10
3.3.3. Interview ..... 11
3.4. Requirements Validation ..... 11
3.4.1. User Storyboard ..... 12
3.5. User Requirements ..... 12
3.6. Functional Requirements ..... 13
3.7. Non-Functional Requirements ..... 16
3.8. Requirements Interdependency Matrix ..... 17
3.9. Requirements Priority ..... 18
3.10. Use Case Diagram ..... 20
3.11. Use Case Description ..... 22
Chapter 4: System Design ..... 25
4.1. Logical Model Design ..... 26
4.1.1. High-Level System Architecture ..... 26
4.1.2. Data Flow Diagram ..... 27
4.1.3. Class Diagram ..... 29
4.1.4. Sequence Diagrams ..... 30
4.2. Physical Model Design ..... 32
4.2.1. User Interface Flow ..... 32
4.2.2. User Interface Prototype ..... 34
Chapter 5: System Implementation ..... 35
5.1. Introduction ..... 36
5.2. Programming Languages, Technologies, & APIs ..... 36
5.3. Implementation Details ..... 39
5.3.1. Tracker ..... 39
5.3.2. Mobile Application ..... 43
Code Versioning ..... 48
Chapter 6: Test Plan ..... 49
6.1. Introduction ..... 50
6.2. Test Plan ..... 50
6.2.1 Introduction ..... 50
6.2.2. Test Items ..... 50

6.2.4 Out-of-Scope ..... 51
6.2.5 Test Strategy ..... 51
6.2.5. Entry Criteria ..... 51
6.2.8 Exit Criteria ..... 51
6.2.9 Test Deliverables ..... 51
6.2.10 Environmental Needs ..... 51
6.3 Test Execution Document ..... 52
6.3.1 Introduction ..... 52
6.3.2 Functional Testing ..... 52
Chapter 7: Conclusion & Future Works ..... 60
Appendix A: Test Cases ..... 62
Appendix B: Records of Change ..... 68
Appendix C: ..... 75
Appendix D: User Manual ..... 81
References ..... 92

# List of Figures

Figure 1: Gantt Chart of Sniffer ..... 1
Figure 2: Waterfall SDLC ..... 7
Figure 3: Power-Interest matrix of the stakeholders of Sniffer ..... 10
Figure 4: Storyboard to contextualize how the pet tracker and the app will be used ..... 12
Figure 5: Use case diagram of the Sniffer app main functionalities ..... 20
Figure 6: High-level system architecture diagram ..... 26
Figure 7: Context Diagram of Sniffer: Pet Tracking App ..... 27
Figure 8: Level-1 Data Flow Diagram showing the main subprocesses of the mobile application ..... 28
Figure 9: Class diagram ..... 29
Figure 10: Sequence diagram for the account registration activity ..... 30
Figure 11: Sequence diagram for the Login activity ..... 30
Figure 12: Sequence diagram for the pet tracking activity ..... 31
Figure 13: User Interflow Diagram ..... 32
Figure 14: Arduino UNO ..... 38
Figure 15: DFROBOT SIM7000E Shield ..... 38
Figure 16: Sample successful POST request to the Google Cloud Function through Postman ..... 55
Figure 17: State-Transition Diagram of Sniffer ..... 56
Figure 18: Snapshot of Dart Code Metrics Analysis results of the mobile application ..... 59
**Figure 19: Class diagram before editing** ..... 70
**Figure 20: Class diagram after editing** ..... 71
**Figure 21: Initial use case diagram** ..... 73
**Figure 22: Use case diagram after editing** ..... 74

# List of Tables

Table 1: Sniffer Pet Tracking System Features ..... 10
Table 2: Comparing the capabilities of our proposed pet tracking app to existing systems in the market ..... 11
Table 3: Main stakeholders of the system ..... 14
Table 4: Breakdown of project tasks ..... 21
Table 5: Defining roles and responsibilities across the project team ..... 1
Table 6: Assessing and categorizing potential risks associated with Sniffer ..... 2
Table 7: Risk Probability-Impact Metric ..... 3
Table 8: Risk Impact/Probability ..... 4
Table 9: Risk Probability-Impact analysis ..... 4
Table 10: Prioritizing and planning response strategies for identified risks ..... 5
Table 11: Stakeholders of Sniffer Pet Tracking App ..... 9
Table 12: Textual Requirements Labels represented as Numbered Labels ..... 15
Table 13: Requirements Interdependency Matrix ..... 17
Table 14: Functional Requirements Prioritization ..... 18
Table 15: Programming Languages, Technologies & APIs for the Flutter mobile application ..... 36
Table 16: Programming Languages, Technologies & APIs for the GPS tracker ..... 37
Table 17: Boundary Equivalence Partitioning ..... 52
Table 18: Input Data Testing for Equivalence Partition ..... 53
Table 19: Test cases derived from state-transition diagram ..... 56
Table 20: Description of the state-transition test cases and their expected results vs actual results ..... 56
Table 21: Traceability matrix between functional requirements and test cases ..... 58
Table 22: Test Case 6 ..... 62
Table 23: Test Case 7 ..... 62
Table 24: Test Case 8 ..... 62
Table 25: Test Case 9 ..... 63
Table 26: Test Case 10 ..... 63
Table 27: Test Case 11 ..... 63
Table 28: Test Case 12 ..... 64
Table 29: Test Case 13 ..... 64
Table 30: Test Case 14 ..... 64
Table 31: Test Case 15 ..... 65
Table 32: Test Case 16 ..... 65
Table 33: Test Case 17 ..... 65
Table 34: Test Case 18 ..... 66
Table 35: Test Case 19 ..... 66
Table 36: Test Case 20 ..... 66
Table 37: Test Case 22 ..... 67
Table 38: All changes and modifications in the functional requirements ..... 68

# Chapter 1: Introduction

## 1.1.Overview

Statistically, 1 in 3 pets goes missing at some point in their lifetime. Of those that do, less than 23% are reunited with their owners (Basappa, 2021). Sniffer is a mobile application and a GPS tracker that aims to provide pet owners with a solution to be able to track their pets' location and receive alerts if they leave their home.

Using geofencing technology, users can define perimeters for a specific geographical area and receive alerts when their pet crosses the virtual perimeter. They will also be able to track the pet's location live once they leave the assigned premises. This way, pet owners will not have to worry about their pets ever escaping and not being able to find their way back home.

Sniffer was founded to solve the 'lost pets' problem in Jordan, as there is a lack of services or resources to look for missing pets. A piece of hardware will be attached to the pet (through their collar), and the pet owner can assign a virtual fence and know when their pet has exited that boundary.

## 1.2.Problem Statement

A lost dog or cat scenario is every pet owner's worst nightmare. When a beloved pet strays from its home, it can be a traumatic experience for both the pet and its owner. Over the course of their lifespan, one-third of all dogs and cats in the US are reported missing, more than 80% are never located, and between 9,450,000 and 9,632,000 pets end up in shelters, where they are put to death (Hamilton, n.d.).

Our goal is to create a mobile application that connects to an Arduino device with a GPS module, which can be attached to a pet's collar. This will allow pet owners to set virtual fences and receive notifications when their pet exits the designated boundary. Additionally, the pet owner can track the pet's location in real-time, providing peace of mind and a sense of security. The proposed solution aims to alleviate the stress and uncertainty of pet ownership by providing a reliable and convenient way to monitor a pet's location and ensure their safety.

## 1.3. Evaluation of Existing Systems

Listed below are some important features provided by our software, in comparison with the existing systems and methods discussed above:

*Table 1: Sniffer Pet Tracking System Features*

| Feature ID | Name | Brief description |
|---|---|---|
| 1 | Virtual boundary | Permits the user to define a radius area on the map |

| | | |
|---|---|---|
| 2 | Out of premises alerts | Real-time alerts sent to the user when their pet has exited or entered the set-up boundaries. |
| 3 | Live location tracking | Ability to track the location of the pet live |
| 4 | Pet profile | Customize the pet's profile |
| 5 | Multiple pet profiles | Ability to add multiple pets to be tracked by a single user |

*Table 2: Comparing the capabilities of our proposed pet tracking app to existing systems in the market*

| Name | Description | Cons | Link | Available in Jordan? |
|---|---|---|---|---|
| **Whistle**<br><br>2012 | A GPS-enabled device that attaches to a pet's collar and allows pet owners to track their pet's location in real-time using a mobile app. It only allows for one virtual fence and one alert type, and it is a bit bulky. | - Only available in the US<br>- only allows for one virtual fence,<br>- Expensive compared to other pet tracking devices ($99/year) | Product website | No |
| **Tractive**<br><br>2012 | A GPS pet tracking device that allows pet owners to track their pets in real-time, monitor activity levels, and see the pet's location anywhere in the world. | - Only available in the UK<br>- Expensive compared to other pet tracking devices (49€ plus 13€/monthly subscription)<br>- Does not have a geofencing feature<br>- Users report connectivity issues | Product Website | No |
| **Airtag**<br><br>2021 | AirTag is a small tracking device that uses Apple's Find My network to locate lost items, it has a built-in speaker that can play a sound to help locate a lost item, and it can be paired with an iPhone and can be located through the Find My app. As a pet tracker, it is not durable enough to | - It is not specifically designed for pet tracking, it is intended for tracking everyday items.<br>- Not durable enough to withstand outdoor conditions, it's not water or dust resistant. | Product website | Yes |

| | | | | |
|---|---|---|---|---|
| | withstand outdoor conditions; it's not water or dust resistant, it does not have advanced geofencing or pet-specific features, and it's only compatible with Apple devices. | - Does not have advanced geofencing or pet-specific features.<br>- Only compatible with Apple devices | | |
| **Tabcat**<br><br>2015 | A pet tracking device designed specifically for cats. It uses radio-frequency technology to send a signal between two devices, one worn on the cat's collar and the other held by the owner, with a range of 400 feet. It makes noise to locate lost cats and is easy to set up but it only works with iOS devices; not suitable for outdoor use and can only track 2 cats at a time. | - Designed for cats only<br>- Only works with iOS devices<br>- Not designed for outdoor use<br>- Doesn't offer additional features like geofencing, real-time location tracking<br>- Only uses Bluetooth to track the cat | Product website | No |

### 1.3.1. Gap Analysis

Whistle, Tractive, and Tabcat seem to have the following weaknesses in common:

- Poor marketing and branding
- Poor GPS tracking accuracy and constant signal issues and false ‘lost pet’ alerts
- Poor battery life

Below is a detailed analysis comparing the listed existing systems with Sniffer:

- Whistle: Our pet tracking app offers more options for creating and managing pet profiles, such as the ability to add multiple profiles and edit profile information, while Whistle only allows for one pet profile. Additionally, our app allows for creating, naming, and deleting multiple geographical fences, while Whistle only offers one virtual fence. Moreover, Whistle is only available in the US, and does not serve Jordan nor the MENA region.

- Tractive: Sniffer has a more robust geofencing feature with the ability to assign a specific geofence to a pet, and receives alerts every 60 seconds if a pet leaves the defined area, while Tractive does not have this feature. Additionally, our app allows for live monitoring of the pet’s live location, while Tractive only offers real-time location updates.

- Tabcat: Sniffer has more robust user account management features, such as email verification, and the ability to keep the user logged in, while Tabcat does not have these features. Additionally, our app allows for creating multiple pet profiles, while Tabcat only allows for two. Tabcat is also designed to track cats only, while Sniffer can be placalso be placed on dogs

- AirTag: Sniffer is specifically designed for pet tracking, while AirTag is intended for tracking everyday items. Additionally, our app offers more advanced geofencing and pet monitoring features, such as the ability to assign a specific geofence to a pet and receive alerts when the pet leaves the defined area, while AirTag does not have these features.

## 1.4. Commonly used methods

### 1.4.1. Facebook groups and social media platforms

Facebook and social media platforms can be used as a method to find lost pets or track pet location. These platforms allow pet owners to post information about their lost pet, including photos and a description, and share them with a wide audience. Pet owners can also join local pet-related groups or communities to post about their lost pet and reach a larger audience. Additionally, social media platforms can be used to share information about lost pets that have been found, which can help reunite pets with their owners. However, the effectiveness of this method depends on how quickly the information is shared and the number of people who see it. Additionally, using social media as a method to track pet location is not accurate and real-time, as it depends on the pet owner to post and share the information about the pet's location. Moreover, the method relies on the pet owner's network and the size of the community, so it might not be as efficient as using a dedicated pet tracking device

Facebook is the most common method for locating missing pets in Jordan. As soon as owners realize that their pet is missing, they post a lost pet query on various Facebook groups (neighborhood groups, specialized lost pets groups...etc) with the pet's photo, contact information, and their last known location. This method, while unreliable, is widely used as there is no other service in Jordan that can help track lost pets.

### 1.4.2. Airtag

An AirTag is a small tracking device that is designed to help users keep track of commonly lost or stolen items. It utilizes the network of iPhone users to locate lost items via Bluetooth communication. The device sends out low-power radio waves on a frequency band between 2.400 GHz and 2.483.5 GHz and is equipped with Bluetooth chips and tiny speakers. The paired device obtains these signals, which contain a unique encrypted ID, and identifies the tracker's location if the tag is nearby. The device can be located by playing a sound via the app or asking Siri to locate it with a sound alert. While AirTags are an innovative solution for tracking everyday items, it's not recommended to use them for pet tracking. AirTags are not designed to withstand outdoor conditions, and they may not be able to withstand the wear and tear of being attached to a pet's collar. Additionally, the range of AirTag is limited, and it's not as accurate as other pet tracking devices, so it's not reliable to use them for pets tracking (Basappa, 2021).

When questioned about using AirTags to track pets, Apple was quick to stress that the company designed the AirTag to track items, not people or pets. Through their tests, Apple's

Airtag proved to be unreliable when detecting/locating the missing pet if they get lost in a residential area as the Bluetooth signal is less likely to be picked up by a nearby phone. In large and busy cities, however, there is a higher chance of the Airtag signal being located (Newman, 2021).

### 1.4.3. Pet Microchips

Microchips are small radio-frequency identification transponders that are implanted under a pet's skin, typically between the shoulder blades. They are about the size of a grain of rice and contain a unique identification number. The microchip does not require a battery or any power source, and it has no moving parts. When the pet is scanned by a veterinarian or shelter, the ID number is transmitted, allowing for identification of the pet. The procedure for implanting the microchip is similar to a vaccination and is performed by a veterinarian. This method of pet tracking is passive and does not require any action from the pet owner to track the pet, it is a permanent solution for lost pets, and it can be easily scanned by shelters or vets (Reisen, 2021). This method, however, is not available in Jordan.

## 1.5. Objectives

The project's ultimate goal is to provide pet owners with a reliable, affordable, and user-friendly system which they can use to keep their pets safe. We aim to have a system where the pet owner can easily define a safe zone through the application, get notified whenever the pet leaves the area, and have the ability to track the pet's location in real time. Having this system will decrease the possibility of a pet getting lost with no hope of being found.

## 1.6. Stakeholder analysis

*Table 3: Main stakeholders of the system*

| Stakeholder | Role |
|---|---|
| **Pet owner** | Define the safe zone,<br>Customize the pet profile,<br>Track the pet’s location |
| **Pet care provider** | Track the pet’s location |
| **Project supervisor** | Supervise the team through the whole project |

| | |
|---|---|
| **Project manager** | Monitor the project’s progress and lead the team |
| **Test manager** | Making sure the product is working as expected by testing its functionalities |

# **Chapter 2:** Project Management Plan

# 2. Chapter 2: Project Management Plan

## 2.1.Project Charter

**Project Title:** Sniffer: Missing Pet Tracker
**Date of Authorization:** Oct 24th, 2022
**Project Start Date:** Oct 24th, 2022 **Project Finish Date:** June 2023

**Key Schedule Milestones:**

- Complete the first version of the software requirements specifications (SRS) by January 2023
- Complete the first version of the high-level design document of the software by January 2023
- Complete the first functioning prototype of the software by April 2023
- Complete a fully functioning prototype of the software by June 2023

**Budget Information:** The budget allocated for this project is 150JDs, used to fund the costs associated with the system's hardware components. Mobile application development will be done internally by the team members.

**Project Manager:** Lian Attily
lia20190133@std.psut.edu.jo
(079) 951-3313

**Project Objectives:** The main objective of the Sniffer project is to develop a reliable and user-friendly system for pet owners to track their pets' location and be alerted when they run away. We aim to have a system where the pet owner can easily define a safe zone through the application, get notified whenever the pet leaves the area, and have the ability to track the pet's location in real-time. Having this system will decrease the possibility of a pet getting lost with no hope of being found. The final version of the documentation and a fully functioning and tested version of the software are to be submitted by June 2023.

**Main Project Success Criteria:** To submit a functioning version of a mobile application integrated with an embedded Arduino device that fully meets all the requirements set in the SRS and does not exceed the budget by June 2023.

**Approach:**

- Develop a proof of concept for the system (Integrating an embedded GPS tracking Arduino device with a mobile application)
- Learn Arduino and the mobile application development language necessary to implement the project
- Deliver the first version of the SRS and high-level design document by January 2023
- Use an incremental development approach to building the system and test after every

increment
- Use GitHub for version control and continuous integration of the system
- Use Google Workspace to manage documentation and communication between the team and supervisor
- Schedule Bi-weekly meetings with the team supervisor to monitor project progress

**Roles and Responsibilities:**

| Name | Role | Position | Contact Information |
|---|---|---|---|
| Dr. Firas Al-Ghanim | Supervisor | Supervise | f.ghanim@psut.edu.jo |
| Lian Attily | Project Manager, Development Team Member | Developer, UX Designer | lia20190133@std.psut.edu.jo |
| Qosay Shatel | Development Team Member, Testing Team member | Developer, Tester | qos20190668@std.psut.edu.jo |

## 2.2. Software Development Approach

This project will follow a waterfall software development technique, further justification and details in the SDLC section.

## 2.3. Scope Statement

**Project Title:** Sniffer: Pet Tracker
**Date of project approval:** Oct. 24th 2022
**Date prepared:** Nov. 23rd, 2022 **Prepared by:** Lian Attily
lia20190133@std.psut.edu.jo

**Project Justification:** Currently, there is no implemented software here in Jordan that alerts pet owners when pets leave their homes and allows them to track their location live through a mobile application. Pet owners turn to posting on local Facebook groups and asking about their pets through word-of-mouth, which is both uncertain and inefficient. Sniffer is a pet GPS tracker and a mobile application that alerts the user if their pet leaves the assigned perimeter and allows the pet owner to track the live location of their pet(s) to eliminate the missing pets issue.

**Project Deliverables:**
*Product deliverables:*
1. Cross-Platform Mobile Application for pet owners

2. Embedded Arduino Device that will go on the pet's collar, used to alert the pet owner if their pet leaves the premises and track the pet's location live.

*Project Management deliverables:*

1. Software Project Management Plan (PMP)
    a. Project Charter
    b. Scope Statement
    c. Work Breakdown Structure
    d. Gantt Chart
    e. Risk Analysis
2. Software Requirements Specification Document (SRS)
3. System Design
4. System Test Plan and Document

**Project Success Criteria:** Submitting a fully functioning version of a mobile application and embedded Arduino device by June 2023 that fully meets all requirements set in the SRS and does not exceed the allocated budget.

**Project Scope Description:**
A user-friendly mobile application will be developed and connected to an embedded device that allows the user to define a safe radius for their pet, alerts them when their pet has left the area and allows them to track their location live.

**Constraints:**

- Time (The final version of the software must be submitted within 8 months)
- Developers lack experience in embedded hardware software
- Resources (Only two developers will be working on this project)

## 2.4. Work Breakdown Structure (WBS)

| Name | Begin date | End date | Duration |
|---|---|---|---|
| | | | |
| **Research & Initialization Phase** | 10/11/22 | 10/21/22 | 9 |
| • Brainstorming solutions | 10/11/22 | 10/14/22 | 4 |
| • Research existing systems | 10/17/22 | 10/21/22 | 5 |
| | | | |
| **Planning Phase** | 10/24/22 | 11/16/22 | 18 |
| • Develop project charter | 10/24/22 | 10/26/22 | 3 |
| • Develop scope statement | 10/27/22 | 10/31/22 | 3 |
| • Develop WBS | 11/01/22 | 11/07/22 | 5 |

| | | | |
|---|---|---|---|
| • Develop risk management plan | 11/08/22 | 11/16/22 | 7 |
| | | | |
| **Analysis Phase** | 11/17/22 | 12/26/22 | 28 |
| • SRS (Software Requirements Specification) | 11/17/22 | 12/26/22 | 28 |
| o Identify Stakeholders | 11/17/22 | 11/17/22 | 1 |
| o Requirements elicitation | 11/18/22 | 11/28/22 | 7 |
| o Requirements specification | 11/29/22 | 12/07/22 | 7 |
| o Requirements management | 12/08/22 | 12/26/22 | 13 |
| ▪ Requirements interdependency | 12/08/22 | 12/13/22 | 4 |
| ▪ Requirements traceability matrix | 12/08/22 | 12/13/22 | 4 |
| ▪ Requirements prioritization | 12/14/22 | 12/15/22 | 2 |
| ▪ Requirements validation | 12/14/22 | 12/26/22 | 9 |
| • UI prototypes | 12/14/22 | 12/20/22 | 5 |
| • Storyboards | 12/21/22 | 12/26/22 | 4 |
| o Use case diagram | 12/21/22 | 12/26/22 | 4 |
| • Power-Interest matrix | 11/18/22 | 11/18/22 | 1 |
| | | | |
| **Design Phase** | 12/27/22 | 02/08/23 | 25 |
| • SDD (Software Design Description) | 12/27/22 | 01/25/23 | 22 |
| o High-Level system architecture | 12/27/22 | 12/30/22 | 4 |
| o User interface flow diagram | 12/27/22 | 12/30/22 | 4 |
| o UML design | 01/02/23 | 01/25/23 | 18 |
| ▪ Class diagram | 01/02/23 | 01/05/23 | 4 |
| ▪ Sequence diagrams | 01/06/23 | 01/16/23 | 7 |
| ▪ State transition diagrams | 01/17/23 | 01/26/23 | 7 |
| ▪ Identify constraints | 01/27/23 | 02/02/23 | 5 |
| • Identify design patterns | 02/03/23 | 02/08/23 | 5 |
| | | | |
| **Implementation Phase** | 02/10/23 | 06/06/23 | 116 |
| • Flutter mobile application development | 02/10/23 | 06/06/23 | 116 |
| • Arduino & GPS module | 04/14/23 | 05/15/23 | 29 |
| | | | |
| | | | |
| **Testing & Integration Phase** | 05/15/23 | 06/06/23 | 23 |
| • Integrate Arduino with mobile application | 04/20/23 | 05/05/23 | 15 |
| • STD (Software Test Document) | 05/07/23 | 06/01/23 | 24 |
| o Test plan | 05/07/23 | 05/15/23 | 8 |
| ▪ Identify test scope | 05/16/23 | 05/17/23 | 1 |

| | | | |
|---|---|---|---|
| ▪ Identify test approach | 05/18/23 | 05/20/23 | 2 |
| ▪ Identify test resources | 05/21/23 | 05/23/23 | 2 |
| ▪ Test scheduling | 05/21/23 | 05/23/23 | 2 |
| ▪ Identify acceptance criteria | 05/21/23 | 05/23/23 | 2 |
| ▪ Identify test cases | 05/24/23 | 05/27/23 | 3 |
| ▪ Identify test risks | 05/24/23 | 05/29/23 | 5 |
| • Identify responsibilities | 05/30/23 | 06/01/23 | 2 |
| • Identify deliverables | 06/02/23 | 06/04/23 | 2 |
| • Black-box testing | 05/30/23 | 06/06/23 | 7 |
| • White-box testing | 05/30/23 | 07/06/23 | 7 |
| • Usability testing | 05/30/23 | 06/06/23 | 7 |

## 2.5.Project Tasks

*Table 4: Breakdown of project tasks*

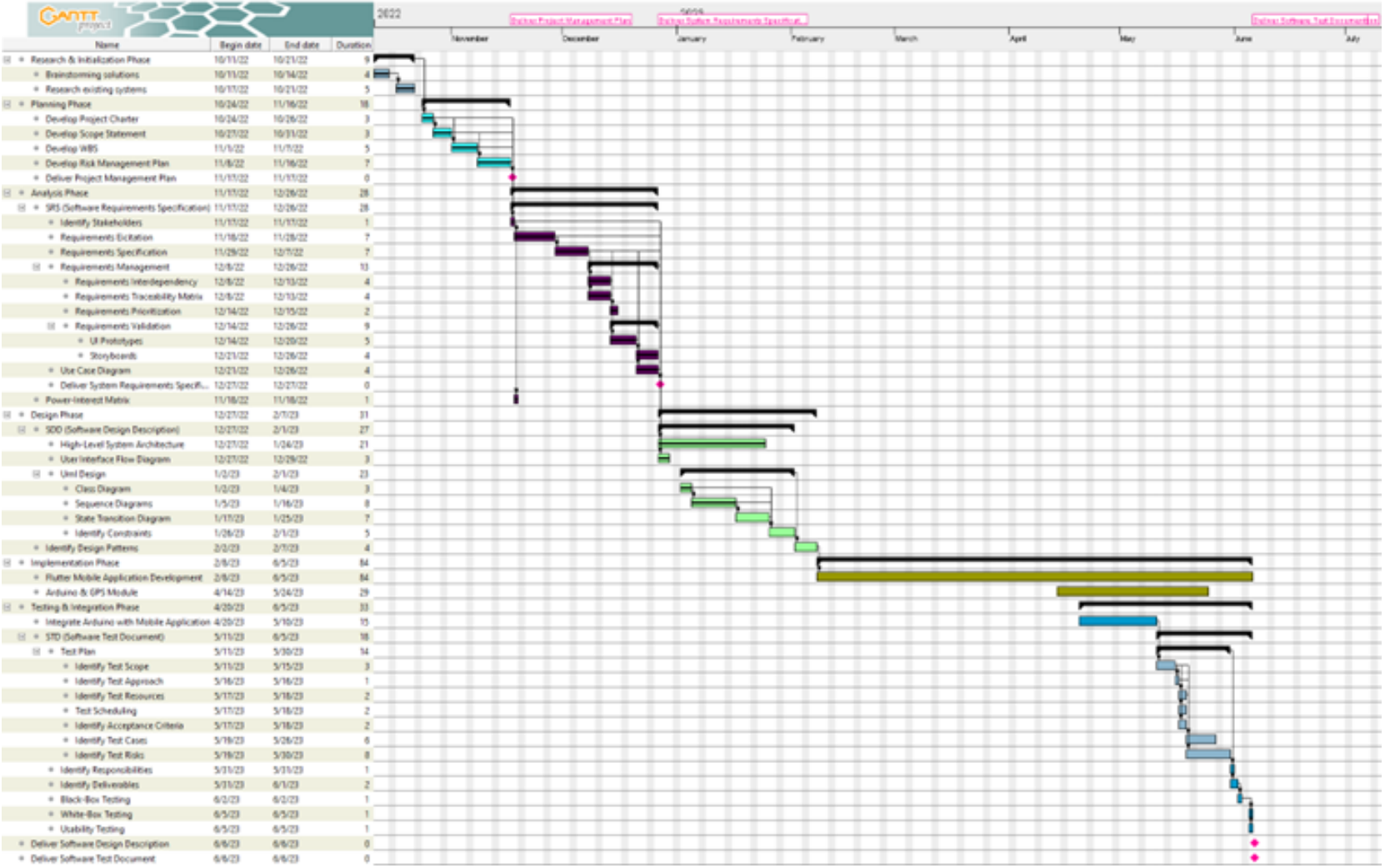
| Phase | ID | Task Name | Description | Pred | Task Allocation |
|---|---|---|---|---|---|
| **Research and Initialization Phase** | A | Brainstorming solutions to the problem | Brainstorm and fine tune solutions for the proposal of the project; How can we solve the issue of missing pets in Jordan? | - | Qosay Al Shatel |
| | B | Research existing systems | Research and study other pet tracking applications and systems and analyze their pros and cons, and what makes Sniffer stand out in the market | A | Lian Attily |
| | C | Develop Project Charter | Develop the Project Charter | A, B | Lian Attily |

| | | | | | |
|---|---|---|---|---|---|
| **Planning Phase** | D | Develop Scope Statement | Develop the Scope Statement for the project | C | Lian Attily |
| | E | Develop Work Breakdown Structure | Develop the Work Breakdown Structure for the activities of the system | C, D | Qosay Al Shatel |
| | F | Develop Risk Management Plan | Identify and analyze the project risks, create a probability-impact matrix | D, E | Qosay Al Shatel |
| **Analysis Phase** | G | Identify Stakeholders | Identify everyone that affects the system or gets effected by it with their role | D, F | Qosay Al Shatel |
| | H | Requirements Elicitation | Conduct interviews with pet-owners and pet shop employees to understand what happens when a pet is lost and how they go about finding them to understand their needs and gain insight on what features we need to implement in our system. | G | Lian Attily |
| | I | Requirements Specification | Identify the functional and non-functional requirements and refine them. | H | Lian Attily |
| | J | Requirements Interdependency | Identify the relationships between the different requirements of the project and how they impact each other | I | Lian Attily |
| | K | Requirements Traceability Matrix | Trace the relationships between the user requirements and the functional/nonfunctional requirements that describe how will it be accomplished | I | Qosay Al Shatel |
| | L | Requirements Prioritization | Prioritizing requirements based on importance, urgency and dependencies | I, J | Qosay Al Shatel |
| | M | UI Prototypes | Create an initial representation of the application’s user interface | I, K | Lian Attily |

| | | | | | |
|---|---|---|---|---|---|
| | N | Storyboards | Develop a visual representation of the illustrating of the project's events | I, M | Qosay Al Shatel |
| | O | Use Case Diagram | Develop the use case diagram by presenting graphically the actors, the functional requirements, and the associations between them | I, M | Lian Attily |
| | P | Power-Interest Matrix | Identify the degree of control and the level of involvement or concern for each stakeholder to prioritize them accordingly | G | Qosay Al Shatel |
| **Design Phase** | Q | High-Level System Architecture | Identify the overall structural layout of the system including the modules, data stores, interfaces and networks using a block diagram | I, O | Qosay Al Shatel |
| | R | User Interface Flow Diagram | Create the user interface design for the mobile application, as well as the embedded GPS system | O | Lian Attily |
| | S | UML Design | Interpret the Software Requirements Specifications into UML diagrams and models (Class diagram, Sequence diagrams, State transition diagrams) & identify OCL constraints | R | Lian Attily |
| | T | Design Patterns | Identify the design patterns used | S | Qosay Al Shatel |
| | U | Flutter mobile application development | Implement the requirements from the SRS and design models into code | T | Lian Attily, Qosay Al Shatel |
| | V | Arduino & GPS Module | Implement the embedded system requirements into the Arduino GPS module | U | Lian Attily, Qosay Al Shatel |

| | | | | | |
|---|---|---|---|---|---|
| **Implementation Phase** | | | | | |
| **Testing & Integration** | W | Integrate Arduino with Mobile Application | Integrate the hardware components (Arduino system) with the Flutter mobile application and ensure they function as specified | U, V | Lian Attily, Qosay Al Shatel |
| | X | Identify Test Plan | Create the test plan by identifying the test scope, test approach, test resources, test scheduling, acceptance criteria, test cases, and test risks | W | Qosay Al Shatel |
| | Y | Identify Responsibilities | Assigning the tasks which makes up the Software Test Document to each of the team members | X | Qosay Al Shatel |
| | Z | Identify Deliverables | Identify the documents that need to be delivered at the end of the testing phase | X | Qosay Al Shatel |
| | A2 | Black Box Testing | Test the functionality of the system and ensure it functions as required without looking at the source code. | Z | Qosay Al Shatel |
| | B2 | White Box Testing | Test the functionality of the system and ensure the source code expected output matches the actual system behavior | A2 | Lian Attily |
| | C2 | Usability Testing | Test the system's reliability and performance against the non-functional requirements specifications | A2 | Lian Attily |

## 2.6.Gantt Chart




| Name | Begin date | End date | Duration |
|---|---|---|---|
| Research & Initialization Phase | 10/11/22 | 10/21/22 | 9 |
| Brainstorming solutions | 10/11/22 | 10/14/22 | 4 |
| Research existing systems | 10/17/22 | 10/21/22 | 5 |
| Planning Phase | 10/24/22 | 11/16/22 | 18 |
| Develop Project Charter | 10/24/22 | 10/26/22 | 3 |
| Develop Scope Statement | 10/27/22 | 10/31/22 | 3 |
| Develop WBS | 11/1/22 | 11/7/22 | 5 |
| Develop Risk Management Plan | 11/8/22 | 11/16/22 | 7 |
| Deliver Project Management Plan | 11/17/22 | 11/17/22 | 0 |
| Analysis Phase | 11/17/22 | 12/26/22 | 28 |
| SRS (Software Requirements Specification) | 11/17/22 | 12/26/22 | 28 |
| Identify Stakeholders | 11/17/22 | 11/17/22 | 1 |
| Requirements Ecitation | 11/18/22 | 11/28/22 | 7 |
| Requirements Specification | 11/29/22 | 12/7/22 | 7 |
| Requirements Management | 12/8/22 | 12/26/22 | 13 |
| Requirements Interdependency | 12/8/22 | 12/13/22 | 4 |
| Requirements Traceability Matrix | 12/8/22 | 12/13/22 | 4 |
| Requirements Prioritization | 12/14/22 | 12/15/22 | 2 |
| Requirements Validation | 12/14/22 | 12/26/22 | 9 |
| UI Prototypes | 12/14/22 | 12/20/22 | 5 |
| Storyboards | 12/21/22 | 12/26/22 | 4 |
| Use Case Diagram | 12/21/22 | 12/26/22 | 4 |
| Deliver System Requirements Specifi... | 12/27/22 | 12/27/22 | 0 |
| Power-Interest Matrix | 11/18/22 | 11/18/22 | 1 |
| Design Phase | 12/27/22 | 2/7/23 | 31 |
| SDD (Software Design Description) | 12/27/22 | 2/1/23 | 27 |
| High-Level System Architecture | 12/27/22 | 1/24/23 | 21 |
| User Interface Flow Diagram | 12/27/22 | 12/29/22 | 3 |
| Uml Design | 1/2/23 | 2/1/23 | 23 |
| Class Diagram | 1/2/23 | 1/4/23 | 3 |
| Sequence Diagrams | 1/5/23 | 1/16/23 | 8 |
| State Transition Diagram | 1/17/23 | 1/25/23 | 7 |
| Identify Constraints | 1/26/23 | 2/1/23 | 5 |
| Identify Design Patterns | 2/2/23 | 2/7/23 | 4 |
| Implementation Phase | 2/8/23 | 6/5/23 | 84 |
| Flutter Mobile Application Development | 2/8/23 | 6/5/23 | 84 |
| Arduino & GPS Module | 4/14/23 | 5/24/23 | 29 |
| Testing & Integration Phase | 4/20/23 | 6/5/23 | 33 |
| Integrate Arduino with Mobile Application | 4/20/23 | 5/10/23 | 15 |
| STD (Software Test Document) | 5/11/23 | 6/5/23 | 18 |
| Test Plan | 5/11/23 | 5/30/23 | 14 |
| Identify Test Scope | 5/11/23 | 5/15/23 | 3 |
| Identify Test Approach | 5/16/23 | 5/16/23 | 1 |
| Identify Test Resources | 5/17/23 | 5/18/23 | 2 |
| Test Scheduling | 5/17/23 | 5/18/23 | 2 |
| Identify Acceptance Criteria | 5/17/23 | 5/18/23 | 2 |
| Identify Test Cases | 5/19/23 | 5/26/23 | 6 |
| Identify Test Risks | 5/19/23 | 5/30/23 | 8 |
| Identify Responsibilities | 5/31/23 | 5/31/23 | 1 |
| Identify Deliverables | 5/31/23 | 6/1/23 | 2 |
| Black-Box Testing | 6/2/23 | 6/2/23 | 1 |
| White-Box Testing | 6/5/23 | 6/5/23 | 1 |
| Usability Testing | 6/5/23 | 6/5/23 | 1 |
| Deliver Software Design Description | 6/6/23 | 6/6/23 | 0 |
| Deliver Software Test Document | 6/6/23 | 6/6/23 | 0 |

*Figure 1: Gantt Chart of Sniffer*

## 2.7.RACI Matrix

*Table 5: Defining roles and responsibilities across the project team*

| Roles/Responsibility | Project Manager | Project Team | Supervisor |
|---|---|---|---|
| Research and Initiation Phase | R | R | C,A |
| Planning Phase | A,C | R | C |
| Analysis Phase | R,A | R,A | C |
| Design Phase | A | R | C |
| Implementation Phase | R,C | R,A | C, I |
| Testing & Integration | R,A | R,A | I |

- R – Responsible [Assigned to complete the task/deliverable]
- A – Accountable [Has the final decision-making authority & accountability for completion]
- C – Consulted [Consulted for approval or advice before making a decision or taking action]
- I – Informed [Must be informed after completing task/deliverable]

## 2.8.Risk Management

### 2.8.1. Risk Source

The team behind the Sniffer project focused mainly on brainstorming, analyzing stakeholders, looking into similar projects, and researching the field of application. There are a variety of resources available for identifying and assessing risks.

### 2.8.2. Risk Identification

*Table 6: Assessing and categorizing potential risks associated with Sniffer*

| Risk ID | Risk Description | Risk Category | Issue Date |
|---|---|---|---|
| **R1** | Failure to deliver the documentation and software in its final form before the deadline | **Business** | **11/08/2022** |
| **R2** | Insufficient communication leading to ambiguity and misunderstandings among team members. | **Organizational** | **11/08/2022** |
| **R3** | Team members do not have experience in mobile application development, specifically using Flutter | **Technical** | **11/08/2022** |
| **R4** | Team members do not have experience in Arduino and hardware embedded systems | **Technical** | **11/08/2022** |
| **R5** | The threat of surpassing the budgeted amount for hardware expenses | **Business** | **11/09/2022** |
| **R6** | Lack of adequate personnel to execute the project within the planned scope and duration | **Organizational** | **11/09/2022** |
| **R7** | Not having a clear understanding of the hardware needs to accomplish project targets | **Technical** | **11/09/2022** |
| **R8** | submitting documentation that is not up to standard or incomplete | **Technical** | **11/09/2022** |
| **R9** | Not meeting customer expectations | **Product** | **11/09/2022** |
| **R10** | The issue of not being able to scale the system to meet the needs of a large user base | **Business** | **11/09/2022** |
| **R11** | Not being able to guarantee the accuracy and reliability of the pet's location information | **Product** | **11/10/2022** |
| **R12** | Difficulty in integrating the Arduino with the Flutter application | **Technical** | **11/10/2022** |
| **R13** | Difficulty in ensuring the reliability of the GPS and SIM module | **Product** | **11/10/2022** |
| **R14** | The likelihood of falling short in achieving the requirement Pet.CreateGeographicalFence | **Product** | **12/10/2022** |
| **R15** | The likelihood of falling short in achieving the requirement Pet.AssignGeofenceToPet | **Product** | **12/10/2022** |
| **R16** | The likelihood of falling short in achieving the requirement Pet.MonitorLocation | **Product** | **12/10/2022** |

| | | | |
|---|---|---|---|
| **R17** | The likelihood of falling short in achieving the requirement Pet.ConfirmFound | **Product** | **12/10/2022** |

### 2.8.3. Risk Metric

*Table 7: Risk Probability-Impact Metric*

| Priority | Probability | Impact |
|---|---|---|
| *High* | *High chance of occurrence* | *Cease project activity if not solved right away* |
| *Medium* | *50% chance of occurrence* | *Cease project activity if not solved* |
| *Low* | *Low chance of occurrence* | *The project activity proceeds, may be solved at last or you can take other solution temporarily* |

### 2.8.4. Risk Analysis

*Table 8: Risk Impact/Probability*

| | | ***Probability*** | | |
|---|---|---|---|---|
| | | *Low* | *Medium* | *High* |
| | *Low* | | *R5, R10* | |
| | *Medium* | *R2* | | *R6* |
| | *High* | *R8* | *R9, R11, R13, R14, R15, R16, R17* | *R1, R3, R4, R7, R12* |

*Table 9: Risk Probability-Impact analysis*

| Risk ID | Probability | Impact | Priority |
|---|---|---|---|
| **R1** | **High** | **High** | **High** |
| **R2** | **Low** | **Medium** | **Low** |
| **R3** | **High** | **High** | **High** |
| **R4** | **High** | **High** | **High** |
| **R5** | **Medium** | **Low** | **Low** |
| **R6** | **High** | **Medium** | **High** |
| **R7** | **High** | **High** | **High** |
| **R8** | **Low** | **High** | **Medium** |
| **R9** | **Medium** | **High** | **High** |
| **R10** | **Medium** | **Low** | **Low** |
| **R11** | **Medium** | **High** | **High** |
| **R12** | **High** | **High** | **High** |
| **R13** | **Medium** | **High** | **High** |
| **R14** | **Medium** | **High** | **High** |
| **R15** | **Medium** | **High** | **High** |
| **R16** | **Medium** | **High** | **High** |
| **R17** | **Medium** | **High** | **High** |

### 2.8.5. Risk Response

The project team must have a response plan for each risk as part of the risk management plan in order to have a clear understanding of the steps that must be done to address those sorts of risks and to be ready for any consequences.

*Table 10: Prioritizing and planning response strategies for identified risks*

| Risk ID | Response Type | Response Description |
|---|---|---|
| R1 | **Mitigation** | Creating an efficient schedule that includes milestones, deadlines, and regular progress reviews. Increasing the working hours and distributing the tasks in a way that the team members time is utilized |
| R2 | **Avoid** | Establishing regular meetings and using tools like Google Docs to follow up on a shared document and to keep up with any updates from the team members to ensure that everyone is aware of the project's progress and any issues that arise. |
| R3 | **Mitigation** | Taking online Flutter courses and reaching out for help from experts |
| R4 | **Mitigation** | Following along with Arduino classes that is given on campus by competent students, taking online courses, and reaching out for help from experts. |
| R5 | **Accept** | Providing additional budget for the project |
| R6 | **Mitigation** | Prioritizing tasks to ensure the completion of the critical ones. Re-evaluating the schedule to be able to identify where can the project be completed more efficiently and in less time |
| R7 | **Mitigation** | Sufficient researching, looking into similar system and reaching out for help from experts |
| R8 | **Mitigation** | Establishing regular meetings with the supervisor to keep up with the team's progress to check their work and to direct them into the next step.<br>Looking over the top documented past graduation projects and following into their steps. |
| R9 | **Mitigation** | Handing out survey to targeted customers, and interviewing them in order to make sure that the product's requirements meet the customer needs.<br>Building design prototype which gives the ability for customers to have a good idea of how the system's final product will be and get their feedback on it. |
| R10 | **Avoid** | Using the Firebase development platform and it's services to handle the data and make the system scalable to accommodate large number of users |
| R11 | **Mitigation** | Testing the accuracy of the data (location) retrieved from the Arduino separately before integrating it with the application and reaching out for help from experts in testing it. |
| R12 | **Mitigation** | Getting the required resources, skills, and knowledge which is needed to integrate the application with the Arduino. |

| | | |
|---|---|---|
| | | Looking into similar systems where there is an application connected to an Arduino. |
| R13 | **Mitigation** | Provide training and resources to the team members to improve their skills and knowledge in ensuring the reliability of the GPS and SIM module. |
| R14 | **Mitigation** | Team members getting sufficient training and resources to improve their skills and knowledge, looking into similar systems that uses similar technology, and getting the help from domain experts |
| R15 | **Mitigation** | Team members getting sufficient training and resources to improve their skills and knowledge, looking into similar systems that uses similar technology, and getting the help from domain experts |
| R16 | **Mitigation** | Team members getting sufficient training and resources to improve their skills and knowledge, looking into similar systems that uses similar technology, and getting the help from domain experts |
| R17 | **Mitigation** | Team members getting sufficient training and resources to improve their skills and knowledge, looking into similar systems that uses similar technology, and getting the help from domain experts |

## 2.9. Software Development Lifecycle

A search and comparison of several software development life cycles was conducted to determine the best-fit methodology for our project. Different approaches, such as Agile, Iterative, Incremental, and Waterfall, were considered.

By looking at the nature of our project and our development team, we have a high level of defined sequential stages, predetermined deadlines, the need for detailed documentation of the different stages, and an inexperienced development team. The *waterfall* model is a good fit when considering those needs only; it’s a simple, deadline-driven, and sequential development process that relies on detailed documentation.

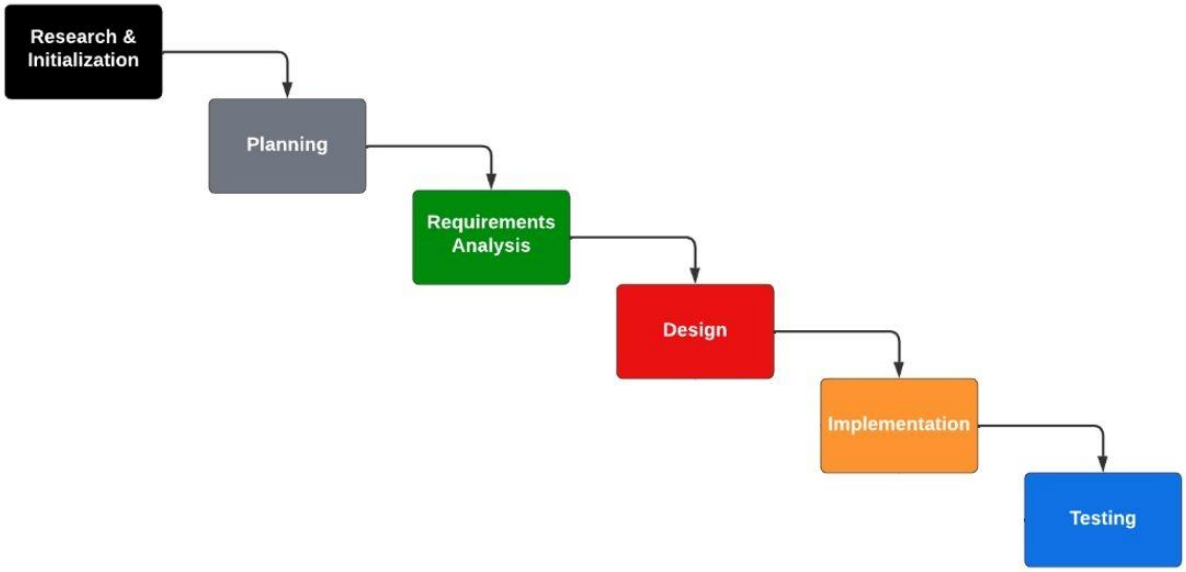



*Figure 2: Waterfall SDLC*

# Chapter 3: Software Requirements Specifications

# 3. Software Requirements Specifications

## 3.1. Stakeholders

*Table 11: Stakeholders of Sniffer Pet Tracking App*

| Stakeholder | Role |
|---|---|
| **Pet owner** | Define the safe zone<br>Customize the pet profile<br>Track the pet's location |
| **Pet Care Provider** | Track the pet's location |
| **Project Supervisor** | Supervise the team through the whole project |
| **Project manager** | Monitor the project's progress and lead the team |
| **Test manager** | Making sure the product is working as expected by testing its functionalities |

## 3.2. Power-Interest Matrix

Each stakeholder has his/her own role in the project, they differ in their power and interest in the product, and how likely are they going to influence or be influenced by the project, so we had to classify them in this matrix to manage them effectively.

- Supervisor
- Project manager
- Test manager
- Pet owner
- Pet care provider

This grid is divided into 4 categories:

- Low interest/low power: stakeholders who are don't have much interest nor impact.
- Low interest/high power: stakeholders who don't have much interest, but have high impact.
- high interest/low power: stakeholder who have much interest but not much impact.
- high interest/high power: stakeholders who have much interest and also have high impact

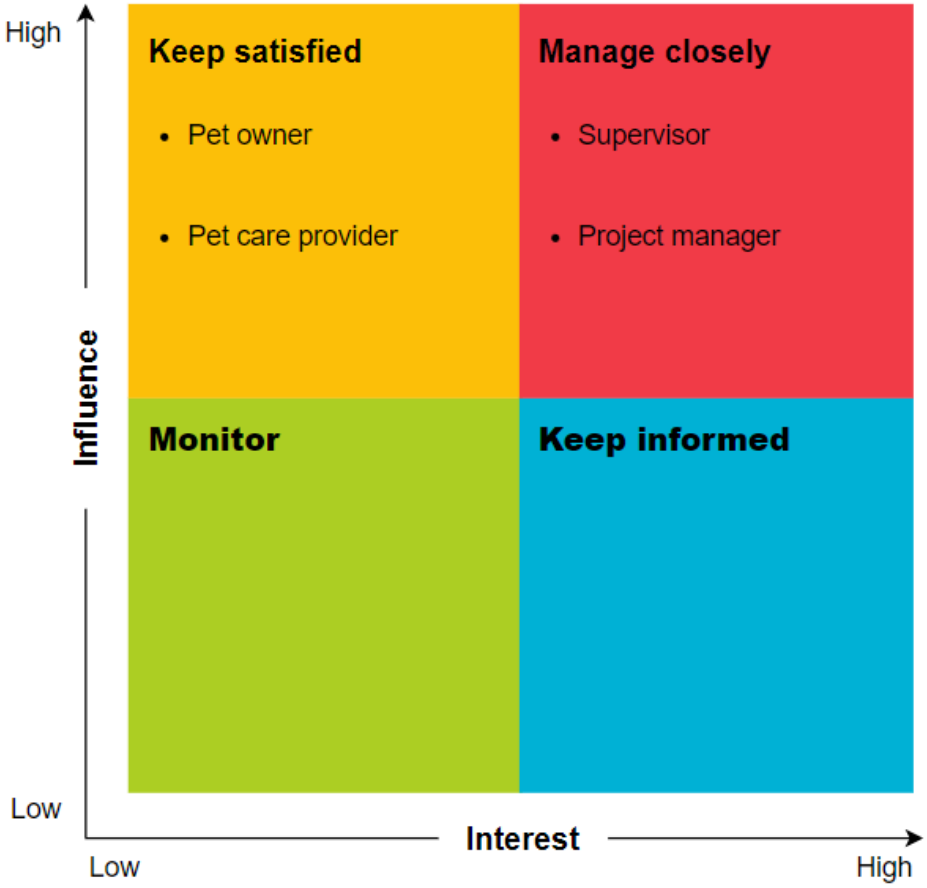



*Figure 3: Power-Interest matrix of the stakeholders of Sniffer*

As shown in figure 3, the stakeholders were classified into groups based on their interest in the project and their capability of impact on it.

## 3.3. Requirements Elicitation

### 3.3.1. Survey

The main requirement elicitation technique used was surveying and questionnaires. Surveys were used to gather quantitative data on pet behavior and the frequency pets going missing.

### 3.3.2. Survey Results

The survey was distributed in two languages (English and Arabic). The results listed are in English and can be found in Appendix C, but the Arabic results are available upon request.

The survey results proved the need for a reliable method to track pet location. The most common search technique for lost pets was posting on social media, and going to search for the pet on foot/in the car.

### 3.3.3. Interview

#### 3.3.3.1. Interview Questions

- Please introduce yourself
    - Name:
    - Age:
    - Background:
- What kind of pet do you own?
    - Name:
    - Kind:
- Is he or she an indoor pet, or an indoor/outdoor pet?
- Do you always keep him or her leashed when outside?
- How much time do you expect your pet to spend alone each day?
- Has your pet ever gotten lost or ran away before?
    - How did you handle it?
- If your pet runs away, how would you look for them?
- What are the chances of finding them, in your opinion? Why?
- What is something that you think would increase your chances of finding your pet if they ran away?

#### 3.3.3.2. Interview Results

After interviewing a sample of stakeholders of the system (dog and cat owners), we were able to gain useful insight about their needs and preferences when it comes to pet tracking. We were also able to understand the context in which the pet tracking app will be used. This can help identify the specific features and functionality that will be most valuable to pet owners. This helps us prioritize the requirements based on user needs, and conduct user-centered design of the system.

## 3.4. Requirements Validation

As a way of validating the requirements, storyboards was used as it’s an effective tool to improve users' comprehension of the system and to foster teamwork among the project team members. They also aid in comprehending the application's place in the real world.

### 3.4.1. User Storyboard

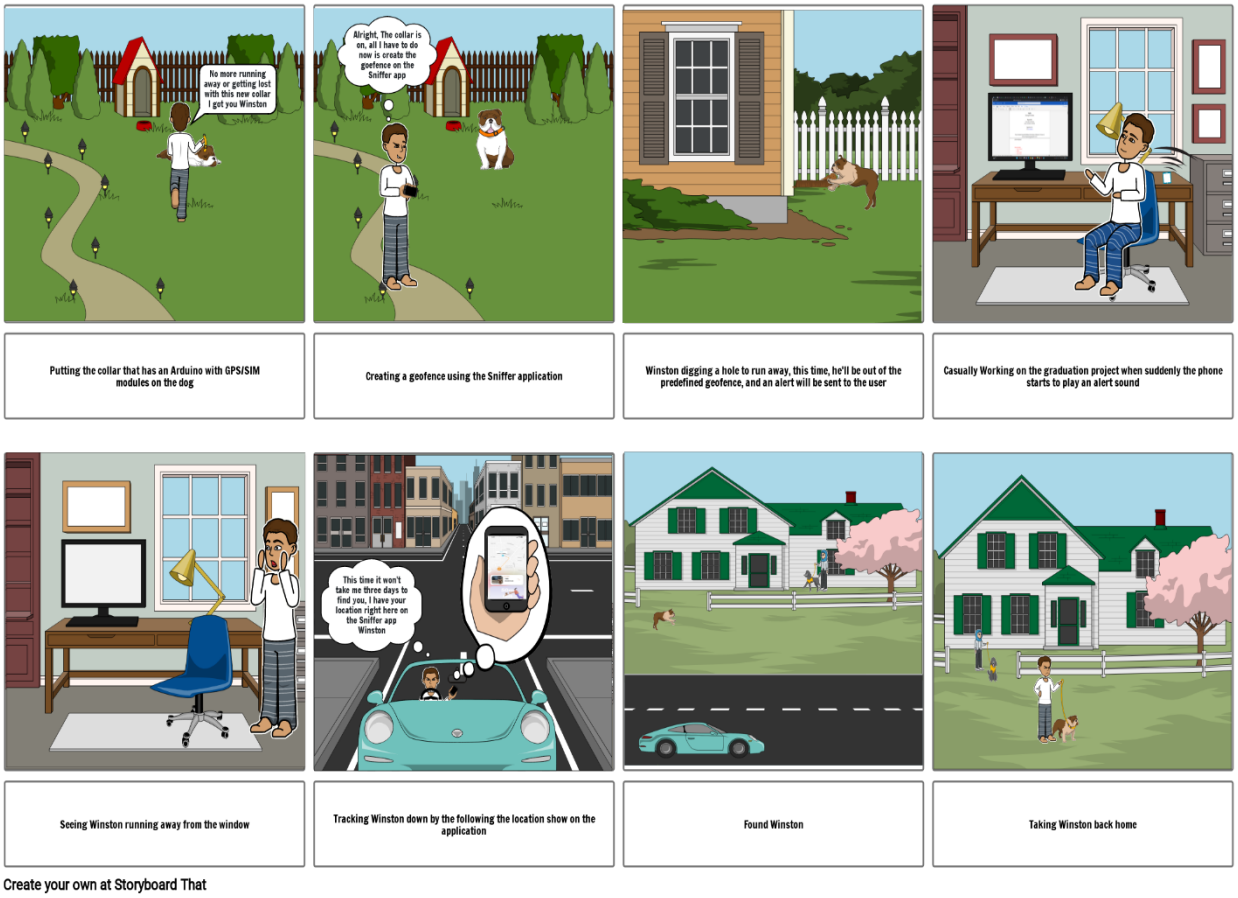



*Figure 4: Storyboard to contextualize how the pet tracker and the app will be used*

## 3.5. User Requirements

**UR-1:** The user shall be able to create an account
**UR-2:** The user shall be able to log into the application using their account
**UR-3:** The user shall be able to create add a pet and create a pet profile
**UR-4:** The user shall be able to edit any pet profile they create
**UR-5:** The user shall be able to delete any pet profile they create
**UR-6:** The user shall be able to create geographical fences
**UR-7:** The user shall be able to name any geographical fence they create
**UR-8:** The user shall be able to delete any geographical fence they create
**UR-9:** The user shall be able to assign a geographical fence to a pet
**UR-10:** The user shall receive notifications if their pet crosses the geographical fence assigned to it

**UR-11:** The user shall be able to track the location of any pet they added
**UR-12:** The user shall be able to confirm that their pet has been found if notified that they crossed the geographical fence assigned to them.

## 3.6. Functional Requirements

**UserAccount**

**.Registration:** The system shall allow the user to create an account by entering their username, email address, and password.

**.Username:** The system shall allow the user to choose a unique username with the following requirements:

a) Should be between 6-12 characters.
b) Should only consist of alphanumeric characters and special characters period (.) or underscore (_)
c) Should not have more than one special character in a row.

**.Password:** The system shall allow the user to choose a password with the following requirements:

a) Should be between 8-15 characters.
b) Should contain at least one upper-case letter.
c) Should contain at least one number.
d) Should contain at least one special character.

**.Verification:** The system shall validate the user's email address by sending them a confirmation link to their email

**.Login:** The system shall allow the user to log in to the account they created using their username and password in ***UserAccount.Registration***

**.KeepLoggedIn:** The system shall keep the user in the 'logged-in' state while they are not logged out.

**PetProfile**

**.Create:** The system shall allow the user to create a pet profile consisting of their:

a) Pet name
b) Photo

c) Type (i.e., Dog, Cat)
d) Birthday
e) Color
f) Weight
g) Special Note

**.MultipleProfiles:** The system shall allow the user to add up to 3 pet profiles

**.View:** The system shall allow the users to view any pet profile they create.

**.Edit:** The system shall allow the user to customize their pet's profile by editing any of the information listed in ***PetProfile.Create***

**.Delete:** The system shall allow the user to delete any pet profile

**Pet**

**.CreateGeographicalFence:** The system shall allow the user to define a safe parameter for each pet by selecting the geographical area in which the pet is allowed to be in and store it in the database

**.NameGeographicalFence:** The system shall allow the user to name the geographical fence they create

**.DeleteGeographicalFence:** The system shall allow the user to delete a previously created geographical fence

**.AssignGeofenceToPet:** The system shall allow the user to specify the pet that the geofence is assigned to

**.RecieveLostPetAlerts:** The system shall send the user alerts every 60 seconds if a pet leaves the defined geographical fence.

**.MonitorLocation**: The system shall allow the user to monitor the pet's location live

**.ConfirmFound:** The system shall allow the user to confirm that their pet has been found

**Tracker**

**.AddTracker:** The system shall allow the user to add a new tracker by entering a unique Tracker ID

**.AssignPet:** The system shall allow the user to assign a pet to a tracker.

**.Edit:** The system shall allow the user to edit the pet assigned to the tracker.

**.Delete**: The system shall allow the user to delete any added tracker.

*Table 12: Textual Requirements Labels represented as Numbered Labels*

| Requirement Label | ID |
|---|---|
| **UserAccount.Registration** | FR-1 |
| **UserAccount.Registration.Username** | FR-1.1 |
| **UserAccount.Registration.Password** | FR-1.2 |
| **UserAccount.Registration.Verification** | FR-1.3 |
| **UserAccount.Login** | FR-2 |
| **UserAccount.Login.KeepLoggedIn** | FR-2.1 |
| **PetProfile.Create** | FR-3 |
| **PetProfile.Create.MultipleProfiles** | FR-3.1 |
| **PetProfile.View** | FR-4 |
| **PetProfile.Edit** | FR-5 |
| **PetProfile.Delete** | FR-6 |
| **Pet.CreateGeographicalFence** | FR-7 |
| **Pet.NameGeographicalFence** | FR-8 |
| **Pet.DeleteGeographicalFence** | FR-9 |
| **Pet.AssignGeofenceToPet** | FR-10 |
| **Pet.RecieveLostPetAlerts** | FR-11 |

| | |
|---|---|
| **Pet.MonitorLocation** | FR-12 |
| **Pet.ConfirmFound** | FR-13 |
| **Tracker.AddTracker** | FR-14 |
| **Tracker.AssignPet** | FR-15 |
| **Tracker.Edit** | FR-16 |
| **Tracker.Delete** | FR-17 |

## 3.7. Non-Functional Requirements

**Usability**: The application should be easy to use, with a clear and intuitive interface.

**Performance**: The application should be able to track the location of pets in real-time, without significant delays or errors.

- The system shall have a maximum delay of 20 seconds

**Reliability**: The application shall be reliable, with a low rate of errors or failures.

- The system shall not fail more than 5 times per year.
- The mean time between failures shall be at least 21 days.

**Security**: The application shall be secure, with measures in place to protect against unauthorized access or data breaches.

- User passwords will be encrypted using hash and stored in the database as the hash value
- Users must be logged in using their username and password to use the application services (Prevent unauthorized access to the system)

**Scalability**: The application shall be able to handle many users and pets without experiencing performance degradation.

- The database architecture must be able to accommodate a 15% increase in users every year.

**Maintainability**: The application shall be easy to maintain, with a clear and well-documented codebase.

**Portability**: The application shall run on iOS and Android devices

## 3.8. Requirements Interdependency Matrix

*Table 13: Requirements Interdependency Matrix*

| Requirement ID | Refines to | Changes to | Similar to | Requires | Conflict With | Increase Cost Of | Increase Value Of | Decrease Cost Of | Decrease Value Of |
|---|---|---|---|---|---|---|---|---|---|
| **FR-1** | FR-1.1, FR-1.2 | - | - | - | - | - | FR-1.3 | - | - |
| **FR-1.1** | - | - | - | - | - | - | - | - | - |
| **FR-1.2** | - | - | - | - | - | - | - | - | - |
| **FR-1.3** | - | - | - | FR-1 | - | - | - | - | - |
| **FR-2** | - | - | - | FR-1 | - | - | - | - | - |
| **FR-2.1** | - | - | - | FR-1, FR-2 | - | - | - | - | FR-2 |
| **FR-3** | - | - | - | FR-1, FR-2 | - | - | FR-4, FR-5, FR-6 | - | - |
| **FR-3.1** | - | - | - | FR-3 | - | FR-3 | FR-3 | - | - |
| **FR-4** | - | - | - | FR-1, FR-2 | - | - | - | - | - |
| **FR-5** | - | - | - | FR-1, FR-2 | - | - | - | - | - |
| **FR-6** | - | - | - | FR-1, FR-2 | - | - | - | - | - |
| **FR-7** | - | - | - | FR-1, FR-2 | - | - | FR-8, FR-9, FR-10 | - | - |
| **FR-8** | - | - | - | FR-1, FR-2 | - | - | - | - | - |
| **FR-9** | - | - | - | FR-1, FR-2 | - | - | - | - | - |
| **FR-10** | - | - | - | FR-1, FR-2 | - | - | - | - | - |
| **FR-11** | - | - | - | FR-1, FR-2, FR-3, FR-7, FR-10, FR-14, FR-15 | - | - | FR-3, FR-7, FR-10, FR-13, FR-14, FR-15 | - | - |
| **FR-12** | - | - | - | FR-1, FR-2 | - | - | - | - | - |
| **FR-13** | - | - | - | FR-1, FR-2, FR-3, FR-7, FR-10, FR-11 | - | - | - | - | - |
| **FR-14** | - | - | - | FR-1, FR-2 | - | - | - | - | - |
| **FR-15** | - | - | - | FR-1, FR-2, FR-3, FR-14 | - | - | FR-3 | - | - |
| **FR-16** | - | - | - | FR-1, FR-2, FR-14 | - | - | - | - | - |
| **FR-17** | - | - | - | FR-1, FR-2, FR-14 | - | - | - | - | - |

## 3.9. Requirements Priority

*Table 14: Functional Requirements Prioritization*

| | Urgency | Importance | Priority |
|---|---|---|---|
| **UserAccount.Registration** | High | Essential | High Priority |
| **UserAccount.Login** | High | Essential | High Priority |
| **PetProfile.Create** | High | Essential | High Priority |
| **PetProfile.View** | Medium | Recommended | Medium Priority |
| **PetProfile.Edit** | High | Essential | High Priority |
| **PetProfile.Delete** | Medium | Recommended | Medium Priority |
| **Pet.CreateGeographicalFence** | High | Essential | High Priority |
| **Pet.NameGeographicalFence** | Medium | Recommended | Medium Priority |
| **Pet.DeleteGeographicalFence** | High | Essential | High Priority |
| **Pet.AssignGeofenceToPet** | High | Essential | High Priority |
| **Pet.RecieveLostPetAlerts** | High | Essential | High Priority |
| **Pet.MonitorLocation** | High | Essential | High Priority |
| **Pet.ConfirmFound** | High | Essential | High Priority |
| **Tracker.AddTracker** | High | Essential | High Priority |

| | | | |
|---|---|---|---|
| **Tracker.AssignPet** | High | Essential | High Priority |
| **Tracker.Edit** | High | Essential | High Priority |
| **Tracker.Delete** | High | Essential | High Priority |

## 3.10. Use Case Diagram

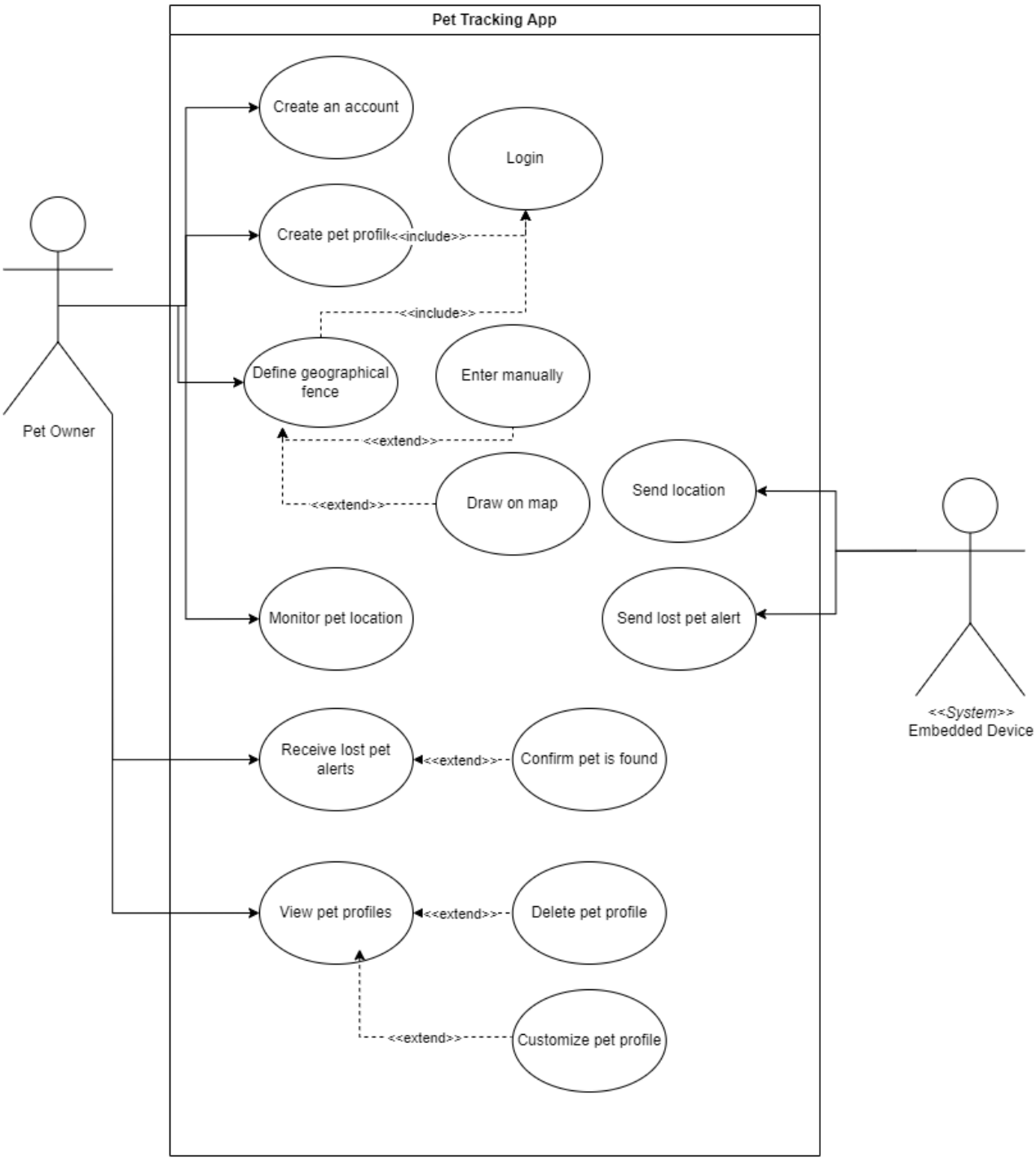



*Figure 5: Use case diagram of the Sniffer app main functionalities*

As a pet owner, the ability to track their pet's location and receive alerts if they leave a designated area is crucial for ensuring their pet's safety and being able to locate them if they go

missing. The pet tracking application on their smartphone will be used to define a geofence for their pet by drawing a boundary on a map within the app. The geofence size and notifications to be sent to the owner's phone if their pet exits the geofence will also be specified.

If the pet leaves the geofence, the owner will receive an alert on their phone with the option to view their current location on a map. They can then use the app to track their pet's location in real time and take any necessary action to locate and retrieve them.

In addition to receiving alerts when the pet leaves the geofence, the owner can also use the app to track their pet's location at any time, even when they are within the geofence. This will allow them to keep an eye on their pet's whereabouts and ensure they are safe and secure.

The hardware device, on the other hand, will send the pet’s location in real time and will alert the application once the set geographical fence is crossed.

## 3.11. Use Case Description

| | |
|---|---|
| **ID:** | **1** |
| **Title:** | Create Pet Profile |
| **Description:** | Create a profile for the pet by adding their basic information (name, type/breed, date of birth, color, photo, optional additional notes) |
| **Primary Actor:** | Application users (pet owners) |
| **Preconditions:** | 1. The user has successfully created an account<br>2. The user has logged into their account |
| **Postconditions:** | Pet profile created with name, type, birthday, color, photo, and notes (optional) |
| **Main Success Scenario:** | 1. User selects “Create pet profile” from menu<br>2. System displays form to fill the name, type, birthday, colour, and any additional notes<br>3. User can assign geographical fence to pet (pre-condition: Create Geographical Fence **[ID#2]**)<br>4. User clicks “Add Pet”<br>5. System stores pet profile to the database |
| **Frequency of Use:** | Once or more a year |
| **Owner:** | Lian Attily |
| **Priority:** | High |

| | |
|---|---|
| **ID:** | **2** |
| **Title:** | Define Geographical Fence |
| **Description:** | Define a geographical fence that a pet must stay in |
| **Primary Actor:** | Application users |
| **Preconditions:** | 1. The user has successfully created an account<br>2. The user has logged into their account<br>3. The user has created at least one pet profile |
| **Postconditions:** | Geographical fence created and stored to database |

| | |
|---|---|
| **Main Success Scenario:** | 1. User selects “Create” button from fences page<br>2. System displays p<br>3. User can tap on the map to add markers and create a fence<br>4. User can enter a name for the geofence defined in step 3<br>5. User clicks “Add Fence”<br>6. System extracts and stores geofence coordinates in the database |
| **Frequency of Use:** | Once or more a year |
| **Owner:** | Lian Attily |
| **Priority:** | High |

| | |
|---|---|
| **ID:** | **3** |
| **Title:** | Confirm Pet Found |
| **Description:** | Confirm that the lost pet is found |
| **Primary Actor:** | Application users |
| **Preconditions:** | 1. The user has successfully created an account<br>2. The user has logged into their account<br>3. The user has created at least one pet profile<br>4. The pet is wearing the device on their collar<br>5. The user received alerts that one of the pets crossed the assigned geofence |
| **Postconditions:** | Confirm pet is found and stop receiving lost pet alerts |
| **Main Success Scenario:** | 1. System sends lost pet alert with pet information<br>2. User can track the pet’s live location<br>3. User clicks “Confirm pet has been found”<br>4. System stops sending lost pet alerts |
| **Frequency of Use:** | Once or more a month |
| **Owner:** | Qosay Al Shatel |
| **Priority:** | High |

| | |
|---|---|
| **ID:** | **4** |
| **Title:** | Add Tracker |
| **Description:** | Register a tracker |
| **Primary Actor:** | Application users |
| **Preconditions:** | 1. The user has successfully created an account<br>2. The user has logged into their account |
| **Postconditions:** | User is logged in |
| **Main Success Scenario:** | 1. User clicks on “Create new” button on fence page<br>2. User enters tracker ID<br>3. System validates tracker ID<br>4. User enters title and assigns pet<br>5. User clicks save<br>6. System adds tracker to the database |
| **Alternative:** | 4. System displays “Invalid tracker ID” |
| **Frequency of Use:** | Once or more a month |
| **Owner:** | Lian Attily |
| **Priority:** | High |

# Chapter 4: System Design

# 4. System Design

## 4.1.Logical Model Design

### 4.1.1. High-Level System Architecture

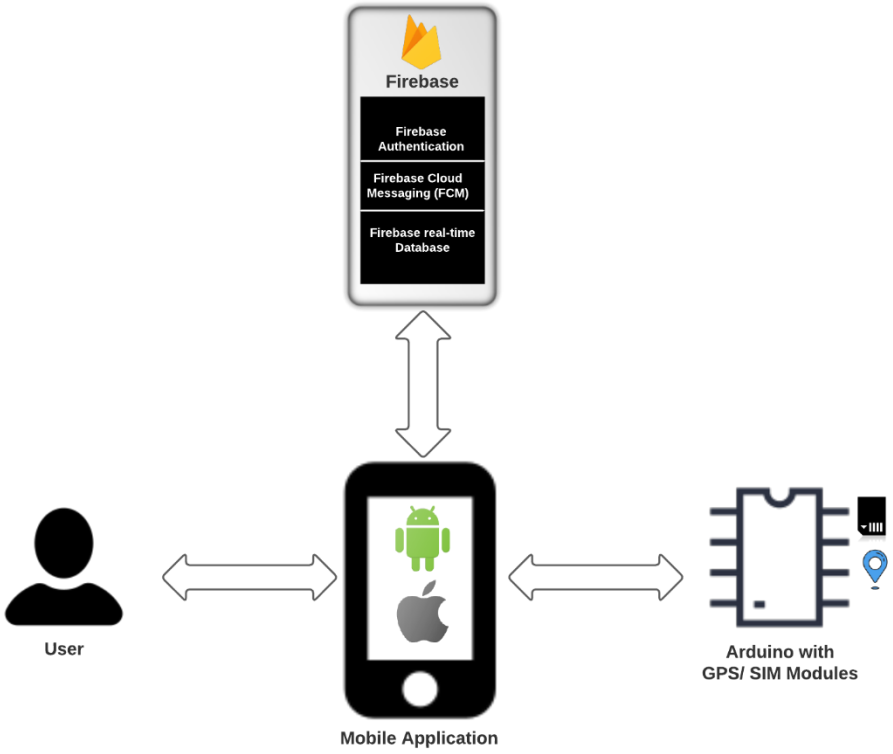



*Figure 6: High-level system architecture diagram*

The system is composed of three main components: the Arduino, the Application, and Firebase. The user deals with the system through the application which is considered the interface of the system. The Arduino, which is connected to both a GPS module and a SIM module, is used to determine the location of the pet and send that information to the Application. The Application, which is available on Android & IOS, allows the user to create geofences, track their pet's location, and receive notifications when the pet leaves the geofence. The application uses Firebase Authentication to handle user authentication and Firebase Cloud Messaging (FCM) to send notifications to the user. Firebase Real-time Database is used to store geofences created by the user and the location data of the pet received from the Arduino. If the pet moves out of the geofence created, the application will notify the user using FCM.

### 4.1.2. Data Flow Diagram

#### 4.1.2.1. Context Diagram

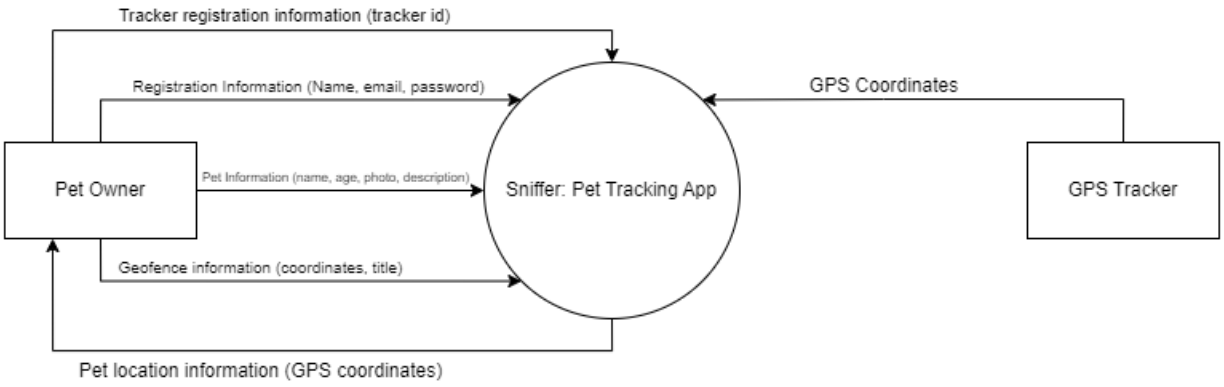



*Figure 7: Context Diagram of Sniffer: Pet Tracking App*

The data flow diagram illustrates the seamless exchange of information between the pet owner and the Arduino GPS tracker, with the mobile application serving as a critical intermediary.

### 4.1.2.2. Data Flow Diagram Level-1

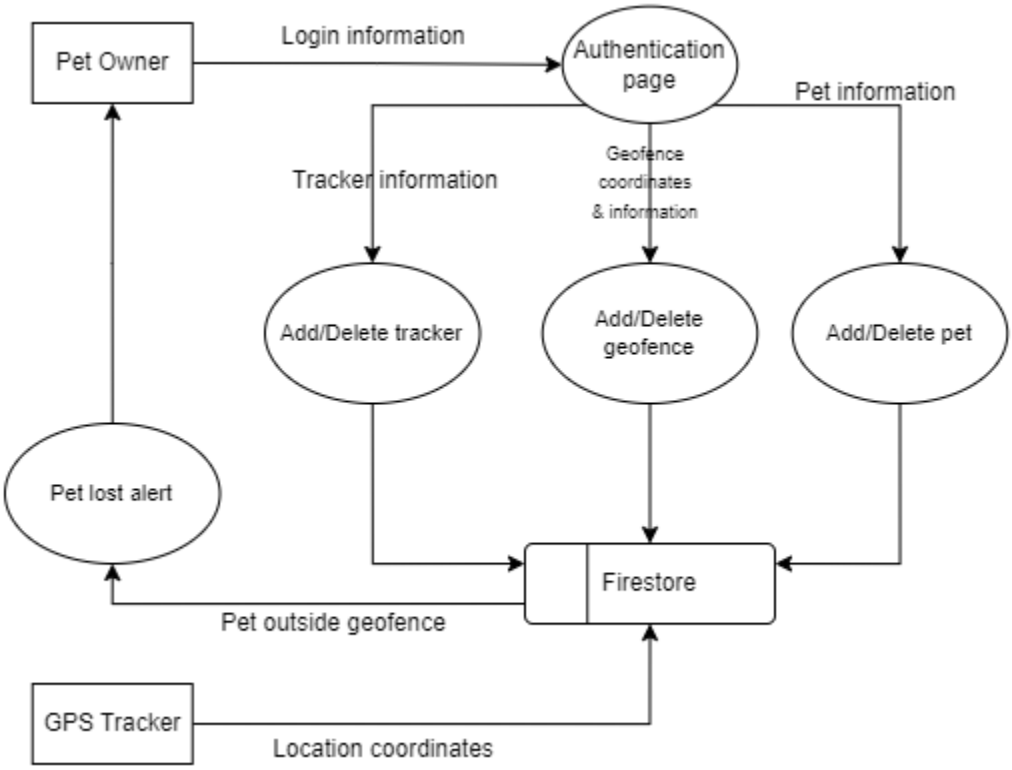



*Figure 8: Level-1 Data Flow Diagram showing the main subprocesses of the mobile application.*

The data flow diagram in the Sniffer demonstrates a comprehensive workflow between various components. The process begins with user authentication, ensuring secure access to the app's features. Once authenticated, users have the capability to add, edit, or remove pet details, geofence boundaries, and tracker configurations. This information is securely stored in the Firebase Firestore database, allowing for persistent data storage and retrieval.

Simultaneously, an Arduino-based GPS tracker attached to the pet's collar continually captures and transmits the pet's location coordinates to the Firebase Firestore database. These coordinates are associated with the respective pet's record, allowing for real-time tracking and analysis.

The Firebase Firestore, acting as the central repository for pet and geofence data, triggers a notification alert mechanism if the pet's current location falls outside the assigned geofence boundaries. This timely alert ensures that pet owners are promptly notified of any potential breaches in the designated safe zones.

### 4.1.3. Class Diagram

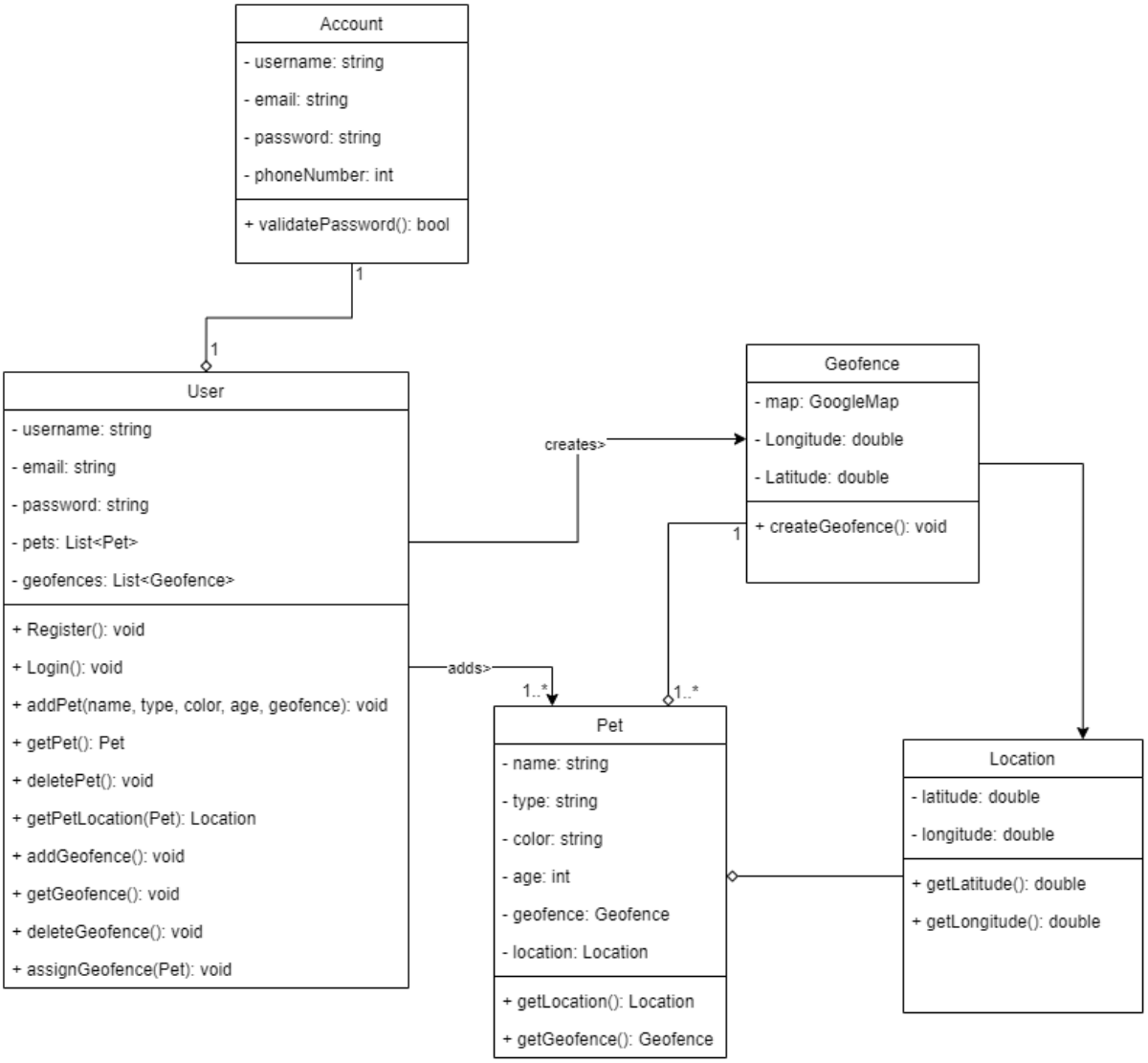



*Figure 9: Class diagram*

The class diagram provides a visual representation of the different classes, objects, and their relationships within the system. The main classes in the system are the User, Account, Pet, Geofence, and Location.

### 4.1.4. Sequence Diagrams

The sequence diagrams below depict a high-level representation of the system's main functionalities. Account registration, logging in, and tracking pet location.

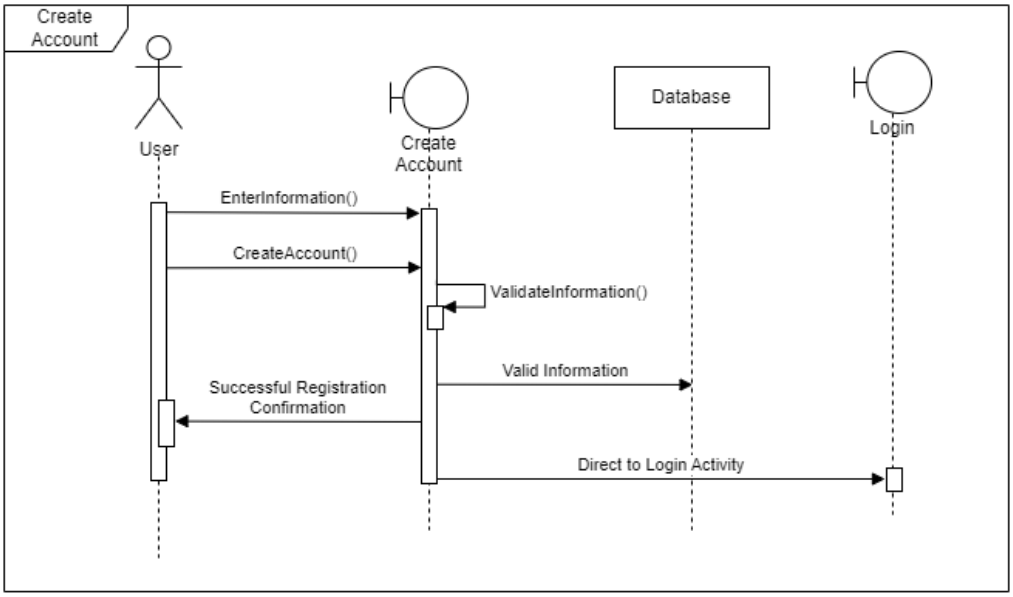



*Figure 10: Sequence diagram for the account registration activity*

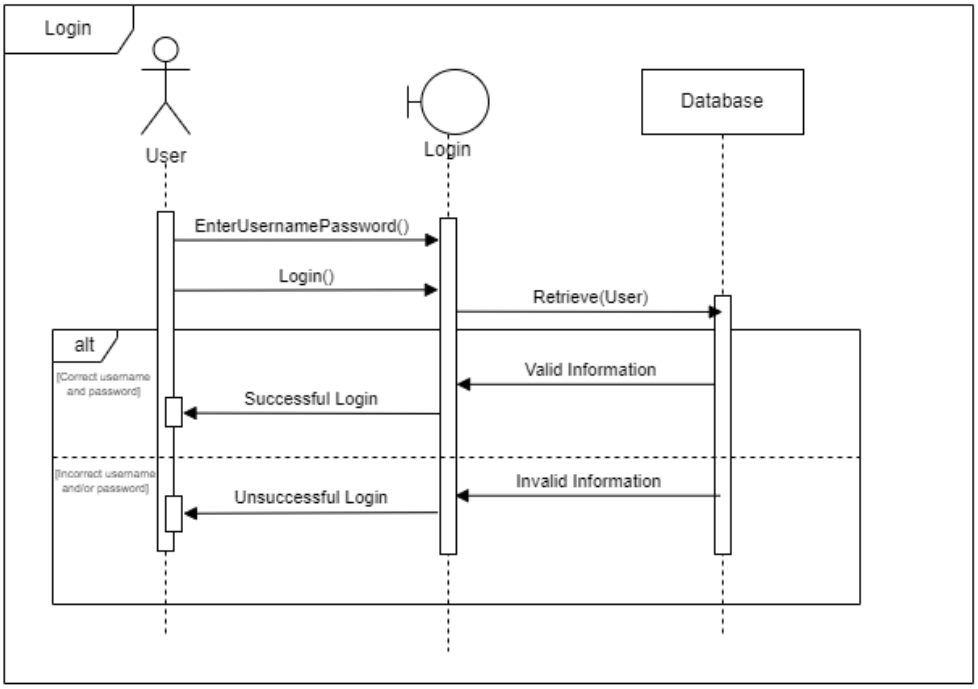



*Figure 11: Sequence diagram for the Login activity*

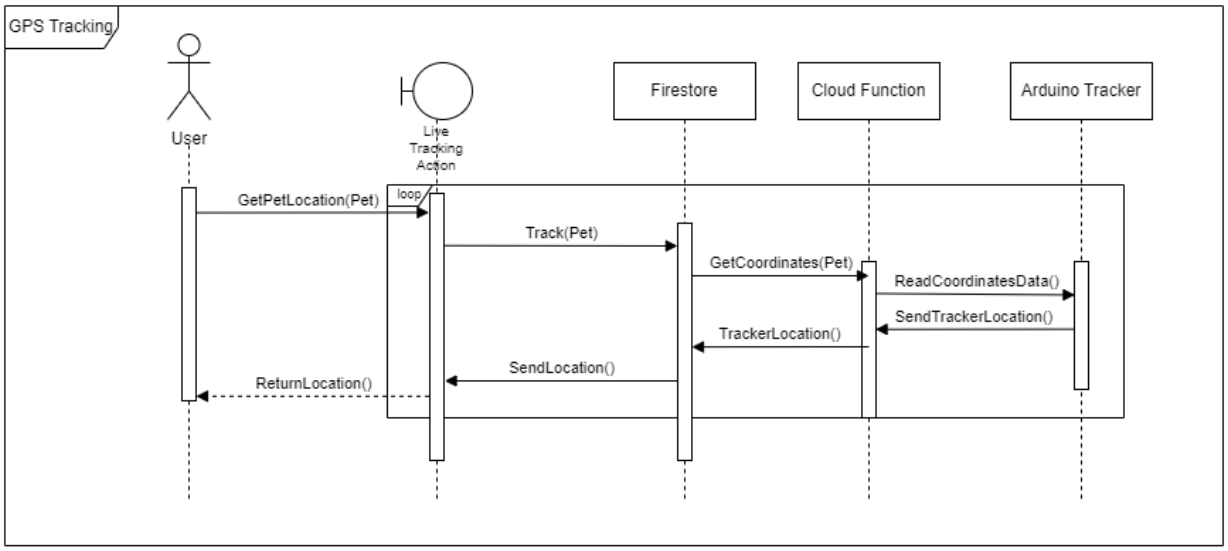



*Figure 12: Sequence diagram for the pet tracking activity*

## 4.2.Physical Model Design

### 4.2.1. User Interface Flow

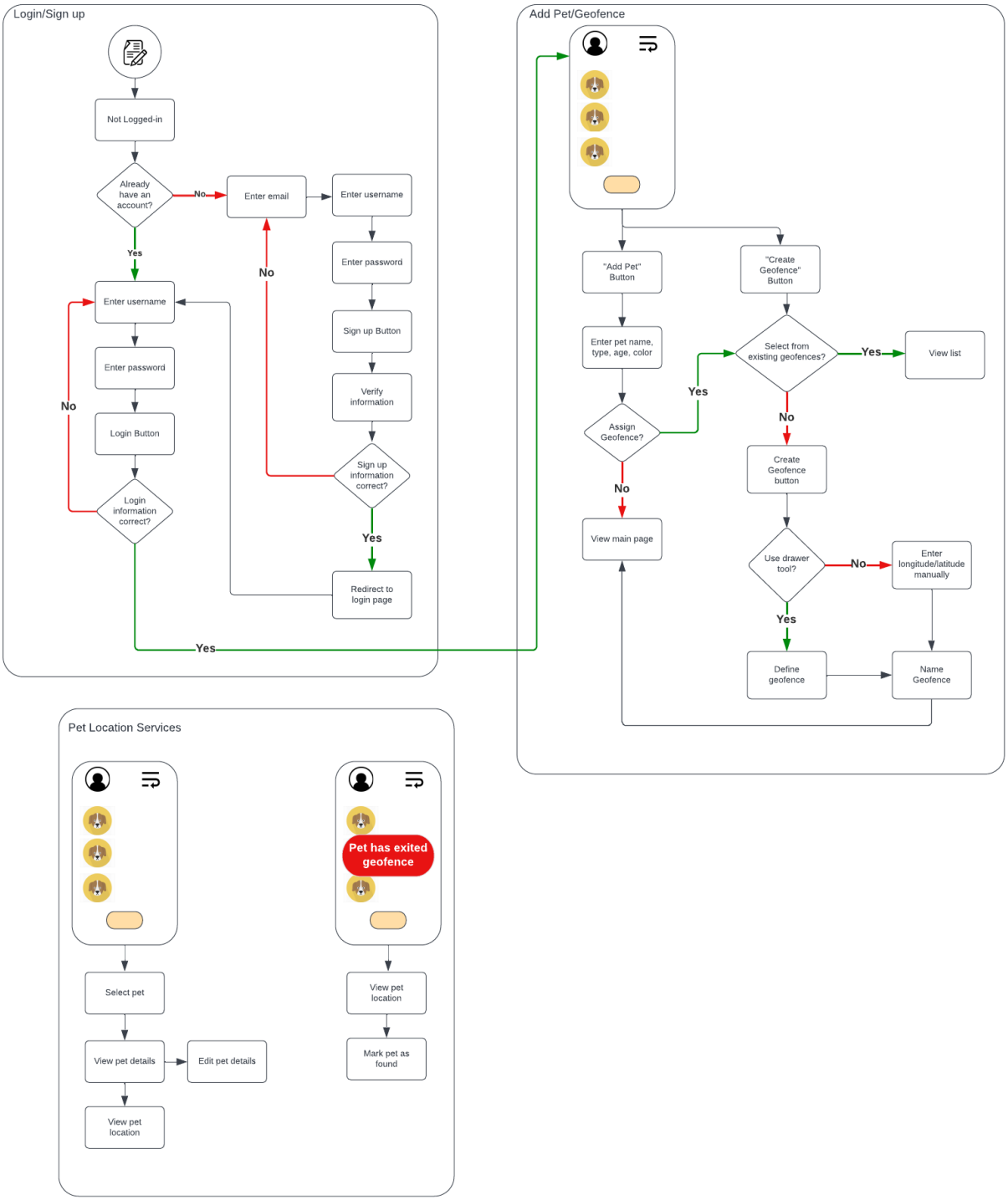
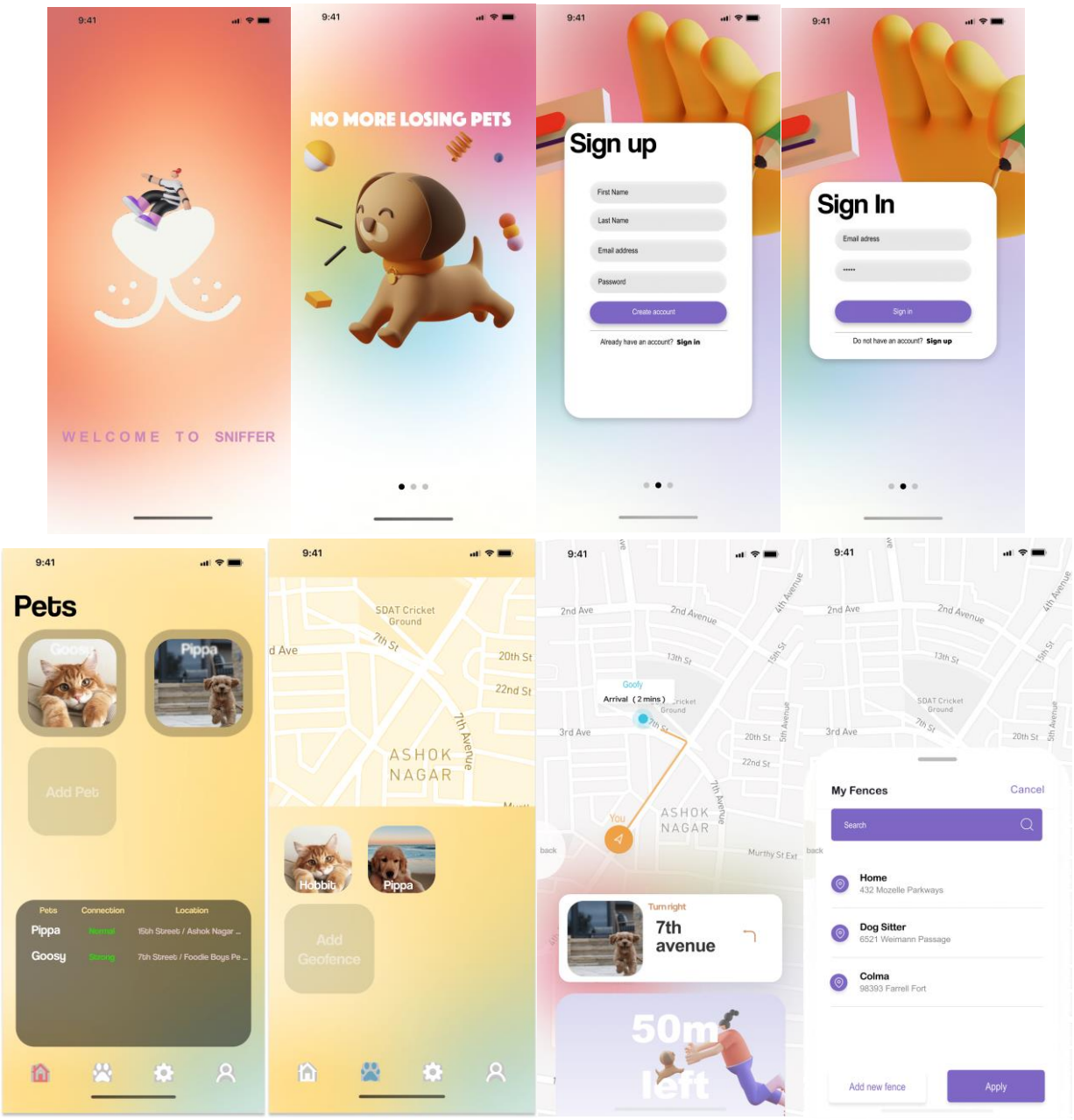



*Figure 13: User Interflow Diagram*

The user flow diagram of Sniffer provides a visual representation of the steps and interactions a user will experience when using the app. It shows the various screens, features, and functionalities that a user will encounter as they navigate through the app, such as signing up, logging in, adding a new pet, and creating a geofence. The diagram illustrates the flow of the user experience, including decision points and potential branches, such as the ability to cancel an action or navigate to a different feature. The user flow diagram serves as a guide for the design and development of the app, ensuring that the user experience is intuitive, efficient, and meets their needs

### 4.2.2. User Interface Prototype



# Chapter 5: System Implementation

## 5.1. Introduction

This chapter provides an overview of the technical aspects and details of how the system is implemented. This chapter aims to provide justifications and insights into the technologies, frameworks, and methodologies used during the development process.

## 5.2. Programming Languages, Technologies, & APIs

### 5.2.2.1. Mobile Application

*Table 15: Programming Languages, Technologies & APIs for the Flutter mobile application*

| Type | Tool/Language/API | Description | Justification |
|---|---|---|---|
| **IDE** | Android Studio | Integrated development environment (IDE) for Android app development | Used to develop the application, provides an emulator |
| **SDK** | Flutter | UI toolkit for building natively compiled applications for cross-platform applications (mobile [iOS/Android], web, and desktop) from a single codebase | Cross-platform framework that allows the code to be reused over Android and iOS |
| **Language** | Dart | Programming language used for building Flutter apps | Flutter's primary language, easy-to-learn syntax |
| **Database/Data Storage** | Firebase Cloud Firestore, & Firebase Storage | NoSQL Database provided by Firebase to sore and sync app data | Stores user, pet, tracker data and provides real-time synchronization and integration with other Firebase services |
| **Cloud Function** | Google Cloud Function | Serverless compute platform provided by Google Cloud for running server-side code in response to events | Reads location coordinates from GPS tracker then performs geofence comparisons and sends push notifications when a pet crosses a designated area |
| **API** | Firebase Cloud Messaging | A cloud-based messaging service provided by Firebase that allows sending push notifications to mobile devices | It is used to send real-time push notifications to mobile devices in the pet tracking app, informing users about geofence alerts |
| **API** | Firebase Auth | Authentication and user management service provided by Firebase | Manages user accounts and user authentication - ensures secure access |

| | | | |
|---|---|---|---|
| | | | to app data and features. |
| **API** | Firebase Core | Core package of Firebase SDK for Flutter, providing necessary functionality for Firebase integration | Initializes Firebase services in the Flutter application and enables communication with backend Firebase services |
| **API** | Google Maps | Mapping and location service provided by Google for displaying maps and location data in mobile applications | Visualizes pet location and geofence on map, allows user interaction when creating a geofence |
| **Testing tool** | Postman | API development and testing tool for making HTTP requests and validating API responses | Testing cloud function and ensuring proper functionality and data exchange |

#### 5.2.2.3 GPS Tracker

*Table 16: Programming Languages, Technologies & APIs for the GPS tracker*

| Type | Tool/Language/API | Description | Justification |
|---|---|---|---|
| **IDE** | Arduino IDE | Open-source software used for programming Arduino boards | It provides essential features such as code editing, compiling, and uploading sketches to the Arduino board |
| **Language** | Arduino Sketch | It is the code written in the Arduino programming language that runs on Arduino boards. | Arduino Sketches provide a structured and straightforward way to program Arduino devices, allowing for reading GPS data and communicating with the SIM7000E shield. |
| **Arduino** | Arduino UNO | A microcontroller board, widely used in various Arduino projects and offers a range of digital and analog input/output pins, as well as built-in components such as LEDs and USB connectivity. | Arduino UNO board is small in size, which is important for a pet tracker that will go around the pet's collar. It offers sufficient processing power and I/O capabilities to handle GPS tracking and communication tasks efficiently. |

| | | | |
|---|---|---|---|
| **Shield** | DFROBOT SIM7000E | DFROBOT SIM7000E is a GSM/GPRS module with integrated GPS functionality. It allows communication over cellular networks and provides GPS positioning capabilities | It provides accurate GPS coordinates, allowing the app to track the pet's location in real-time. The SIM7000E module supports communication via GSM/GPRS, ensuring reliable data transmission between the pet tracker device and the Google Cloud function which then writes the GPS coordinates to Firestore |
| **LTE** | Zain SIM Card | The SIM card is inserted into the SIM7000E module to establish cellular network connectivity and enable data communication. | Used to establish a cellular network connection for transmitting GPS data and communication with the server. |


*Figure 14: Arduino UNO*

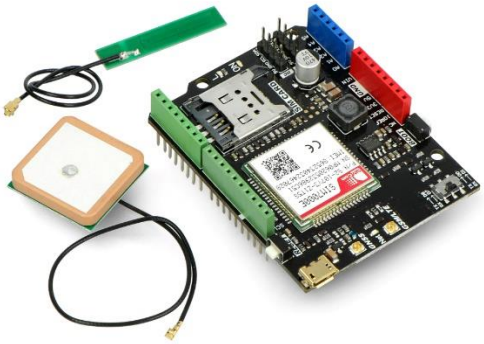


*Figure 15: DFROBOT SIM7000E Shield*

## 5.3. Implementation Details

### 5.3.1. Tracker

Below is a snippet of the Arduino sketch that retrieves the position from the SIM7000 module, formats it (latitude & longitude) into a JSON payload, and a trackerID is assigned a unique value for the tracker. It then sends an HTTP POST request to the Firebase Cloud Function endpoint.

```
  Serial.println("Getting position...");
  while(!sim7000.getPosition()){
    delay(1000);
  }

  String longitude = String(sim7000.getLongitude());
  String latitude = String(sim7000.getLatitude());

  Serial.print("Longitude: ");
  Serial.println(longitude);
  Serial.print("Latitude: ");
  Serial.println(latitude);

  String trackerId = "Allawi2003";
  String json = "{\"latitude\": " + latitude + ", \"longitude\": " + longitude +
", \"trackerId\": \"" + trackerId + "\"}";


  mySerial.println("AT+HTTPPARA=\"URL\",\"http://us-central1-sniffer-
de62b.cloudfunctions.net/function-2\"");
  delay(2000);
  readResponse();
```

Below is the Firebase Cloud Function that writes coordinate data to Firestore (function is written in Node.js). The function is triggered by an HTTP POST request from the GPS tracker (SIM7000E Shield).

After checking if the request method is POST, it extracts the request body and stores it in the **data** variable. Then, in the “trackers” collection in Firestore, it either creates or updates the document reference for the specified trackerId sent in the request body.

The latitude and longitude values from the request body are written to the Firestore document for the specified tracker ID.

The function then retrieves the document snapshot and checks if the tracker is disabled. If it is disabled, the function logs a message and returns a response indicating that the longitude and latitude data has been successfully written to Firestore.

It then retrieves the associated pet ID from the document snapshot and fetches the corresponding pet document from the "pets" collection. Next, it retrieves the fence ID from the pet document. If the fence ID is not present, the function logs a message and returns a response indicating the successful writing of the longitude and latitude data.

Subsequently, the function retrieves the fence document using the fence ID and obtains the fence coordinates from the document snapshot. Using the received GPS coordinates and the fence coordinates, the function determines if the current location is within the geofence by calling the **isPointInPolygon** function from the **geolib** library. If the location is outside the geofence, the function constructs a notification message and sends it using the Firebase Cloud Messaging (FCM) functionality, targeting the specific tracker by using the tracker ID as the topic.

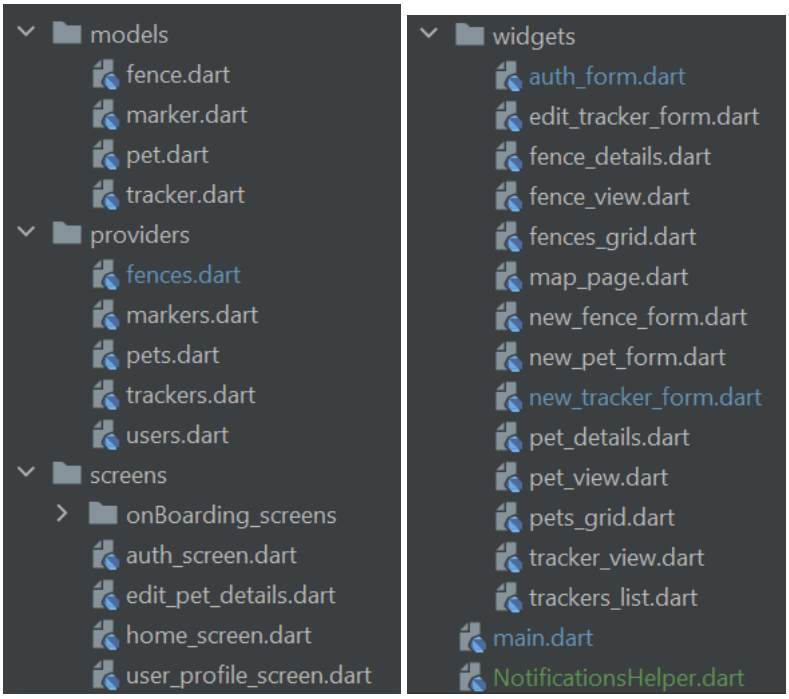
Finally, the function returns a response of 200 OK indicating the successful writing of the longitude and latitude data to Firestore. This Cloud Function enables real-time tracking of pets and triggers geofence alerts when a pet crosses the designated boundaries, providing timely notifications to the user.

```
const functions = require('firebase-functions');
const admin = require('firebase-admin');
const geolib = require('geolib');

admin.initializeApp();

const db = admin.firestore();

exports.writeToFirestore = (async (req, res) => {
  if (req.method !== 'POST') {
    res.status(405).send('Method Not Allowed');
    return;
  }

  const data = req.body;
  const trackerId = req.body;
  const docRef = db.collection('trackers').doc(trackerId);

  await docRef.set({
    latitude: data.latitude,
    longitude: data.longitude,

  },
    { merge: true }
  );

  const docSnapshot = await docRef.get();
  if (!docSnapshot.exists) {
    console.log('Document does not exist');
    return res.status(200).send('Document does not exist.');
  }
  const isDisabled = docSnapshot.get('isDisabled');

  if (isDisabled) {
    console.log('Tracker is disabled');
    return res.status(200).send('Successfully written longitude and latitude to Fire-
store.');
  }

  const petId = docSnapshot.get('petId');
  const petDocRef = db.collection('pets').doc(petId);
  const petSnapshot = await petDocRef.get();

  const fenceId = petSnapshot.get('fenceId');
  if (!fenceId) {
    console.log('fenceId is null or undefined');
    return res.status(200).send('Successfully written longitude and latitude to Fire-
store.');
  }

  const fenceDocRef = db.collection('fences').doc(fenceId);
  const fenceSnapshot = await fenceDocRef.get();
  const fenceCoordinates = fenceSnapshot.get('coordinates');
```

```
  const fenceId = petSnapshot.get('fenceId');
  if (!fenceId) {
    console.log('fenceId is null or undefined');
    return res.status(200).send('Successfully written longitude and latitude to
Firestore.');
  }

  const fenceDocRef = db.collection('fences').doc(fenceId);
  const fenceSnapshot = await fenceDocRef.get();
  if (!fenceSnapshot.exists) {
    console.log('Fence document does not exist');
    return res.status(200).send('Successfully written longitude and latitude to
Firestore.');
  }

  const fenceCoordinates = fenceSnapshot.get('coordinates');
  if (!fenceCoordinates || !Array.isArray(fenceCoordinates) || fenceCoordinates.length
=== 0) {
    console.log('Fence coordinates are null, undefined, or empty');
    return res.status(200).send('Successfully written longitude and latitude to
Firestore.');
  }

  const currentLocation = {
    latitude: data.latitude,
    longitude: data.longitude,
  };

  const isWithinGeofence = geolib.isPointInPolygon(currentLocation, fenceCoordinates);
  if (!isWithinGeofence) {
    const message = {
      notification: {
        title: 'Geofence Alert',
        body: 'The pet has crossed the geofence!',
      },
      topic: trackerId,
    };

    await admin.messaging().send(message);
  }

  return res.status(200).send('Successfully written longitude and latitude to
Firestore.');
});
```

### 5.3.2. Mobile Application

#### 5.3.2.1. Project Structure




In the context of Flutter development, state management holds significant importance in the software engineering process. It involves the handling and manipulation of data within an application, impacting its performance, security, and ease of maintenance.

For Sniffer, the **Provider** package was chosen as the preferred state management solution. Provider helps in clearly defining the areas responsible for state management, reducing the likelihood of errors caused by improper state handling.

It was used to manage services and stateful data including Pets, Fences, and Trackers. Providers were created for each component, encapsulating both the state and the permissible operations (add, search, update, delete, etc.). The Provider package for state management created a valuable layer of abstraction between widgets and direct Firebase operations. By encapsulating the state and Firebase-related operations within dedicated Providers, the widgets were shielded from the complexities of interacting directly with Firebase.

By integrating Provider with widgets, unnecessary code was minimized, and state updates became more streamlined. This implementation improved the overall structure, comprehensibility, and scalability of the application, enabling effective management and response to changes in the application's state. In summary, the adoption of Provider as the state management solution facilitated a well-organized architecture with clear separation of concerns and high maintainability. It contributed to the scalability and robustness of the application, optimizing the development experience.

### 5.3.2.1. Code Snippets

#### Add pet

Function addPet that adds a new Pet to Firebase Firestore using the collection('pets').add() method. The pet's details, such as name, age, image URL, description, owner ID, and fence ID, are passed as a map to be stored in the document.

```
Future<Pet> addPet(Pet pet) async {
  final user = FirebaseAuth.instance.currentUser;
  if (user == null) {
    print("Error: No user logged in.");
    throw Exception('No user logged in');
  }

  final newPet = Pet(
    id: DateTime.now().toString(),
    name: pet.name,
    age: pet.age,
    imageUrl: pet.imageUrl,
    description: pet.description,
    ownerId: user.uid,
    fenceId: pet.fenceId,
  );

  DocumentReference docRef;
  try {
    docRef = await _firestore.collection('pets').add({
      'name': newPet.name,
      'age': newPet.age,
      'imageUrl': newPet.imageUrl,
      'description': newPet.description,
      'ownerId': user.uid,
      'fenceId': newPet.fenceId,
    });
  } catch (error) {
    print('Error adding pet to Firestore: $error');
    throw error;
  }

  final createdPet = Pet(
    id: docRef.id,
    name: newPet.name,
    age: newPet.age,
    imageUrl: newPet.imageUrl,
    description: newPet.description,
    ownerId: user.uid,
    fenceId: newPet.fenceId,
  );

  _pets.add(createdPet);
  notifyListeners();

  try {
    await fetchPets();
  } catch (error) {
    print('Error fetching pets after add: $error');
    throw error;
  }

  print('Pet added successfully: ${createdPet.id}');
  return createdPet;
}
```

### Check Intersecting Coordinates – When Creating a New Fence

The function _isSelfIntersecting() checks if the added coordinate markers intersect and do not form a polygon. It returns true if _segmentsIntersect returns true for any two points that intersect. If none of the points in the fence coordinates

```
bool _segmentsIntersect(
    vm.Vector2 p1, vm.Vector2 q1, vm.Vector2 p2, vm.Vector2 q2) {
  final p1ToQ1 = q1 - p1;
  final p1ToP2 = p2 - p1;
  final p1ToQ2 = q2 - p1;
  final p2ToQ2 = q2 - p2;
  final p2ToP1 = p1 - p2;
  final p2ToQ1 = q1 - p2;

  final cross1 = p1ToQ1.cross(p1ToP2) * p1ToQ1.cross(p1ToQ2);
  final cross2 = p2ToQ2.cross(p2ToP1) * p2ToQ2.cross(p2ToQ1);

  return cross1 < 0 && cross2 < 0;
}

bool _isSelfIntersecting() {
  for (int i = 0; i < _fenceCoordinates.length; i++) {
    final p1 = vm.Vector2(
      _fenceCoordinates[i].latitude,
      _fenceCoordinates[i].longitude,
    );
    final q1 = vm.Vector2(
      _fenceCoordinates[(i + 1) % _fenceCoordinates.length].latitude,
      _fenceCoordinates[(i + 1) % _fenceCoordinates.length].longitude,
    );

    for (int j = i + 1; j < _fenceCoordinates.length; j++) {
      final p2 = vm.Vector2(
        _fenceCoordinates[j].latitude,
        _fenceCoordinates[j].longitude,
      );
      final q2 = vm.Vector2(
        _fenceCoordinates[(j + 1) % _fenceCoordinates.length].latitude,
        _fenceCoordinates[(j + 1) % _fenceCoordinates.length].longitude,
      );

      if (_segmentsIntersect(p1, q1, p2, q2)) {
        return true;
      }
    }
  }

  return false;
}
```

### Notifications Helper

The methods below subscribe and unsubscribe from a specific geofence topic, respectively, and provide visual feedback using SnackBar widgets. Each method takes a “tracker” parameter, representing the topic to subscribe to and receive notifications for through Firebase Cloud Messaging (FCM).

```
void subscribeToGeofenceAlerts(String tracker, BuildContext context) {
  FirebaseMessaging.instance.subscribeToTopic(tracker).then((value) {
    print('Subscribed to $tracker topic!');
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text('Enabled $tracker notifications'),
      ),
    );
  }).catchError((error) {
    print('Failed to subscribe to $tracker topic: $error');
  });
}


Future<void> unsubscribeFromGeofence(String tracker, BuildContext context)
async {
  try {
    await FirebaseMessaging.instance.unsubscribeFromTopic(tracker);
    print('Unsubscribed from $tracker topic');
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text('Disabled $tracker notifications'),
      ),
    );
  } catch (e) {
    print('Failed to unsubscribe from $tracker topic: $e');
  }
}
```

### Update Tracker Details

After updating the trackers list, the **notifyListeners()** method is called to notify any listeners of the changes. The notifyListeners() method is typically used in Flutter's state management approach, such as the **Provider pattern**, to notify listeners that the state has changed.

In the given code snippet, by calling notifyListeners(), any registered listeners, such as widgets that are dependent on the updated tracker data, will be notified of the change and can update their UI accordingly. This helps to keep the user interface synchronized with the updated data.

```
Future<void> updateTrackerDetails(
    Tracker tracker, BuildContext context) async {
  try {
    await _firestore
        .collection('trackers')
        .doc(tracker.id)
        .set(tracker.toMap());

    int index = _trackers
        .indexWhere((existingTracker) => existingTracker.id == tracker.id);
    if (index != -1) {
      _trackers[index] = tracker;
    } else {
      _trackers.add(tracker);
    }
    notifyListeners();
  } catch (error) {
    print('Error updating tracker: $error');
  }
}
```

### Delete Tracker

Removes a tracker from the Firestore database and updates the local state in Flutter.

```
Future<void> removeTracker(String trackerId) async {
  try {
    final trackerRef = _firestore.collection('trackers').doc(trackerId);

    await trackerRef.update({
      'ownerId': FieldValue.delete(),
      'title': FieldValue.delete(),
      'petId': FieldValue.delete(),
      'isDisabled': FieldValue.delete(),
    });

    _trackers
        .removeWhere((existingTracker) => existingTracker.id == trackerId);
    notifyListeners();
  } catch (e) {
    print('Error removing tracker: $e');
    throw e;
  }
}
```

### Add fence

Adds a new fence to the Firestore database and updates the local state in Flutter.

```
Future<void> addFence(Fence fence) async {
  try {
    final user = _auth.currentUser;
    if (user == null) {
      throw Exception('User not found');
    }

    final fenceData = fence.toMap();

    final docRef = await _firestore.collection('fences').add(fenceData);
    // Update the fence ID with the document ID from Firebase
    final updatedFence = fence.copyWith(id: docRef.id);

    _fences.add(updatedFence);
    notifyListeners();
  } catch (error) {
    throw error;
  }
}
```

### Password requirements

The passwordRequirements() function checks whether a given password meets certain requirements:

- **hasUppercase** = true if the password contains at least one uppercase letter (A-Z).
- **hasDigits** = true if the password contains at least one digit (0-9).
- **hasLowercase** = true if the password contains at least one lowercase letter (a-z).
- **hasSpecialCharacters** = true if the password contains at least one special character from the specified set (![!@#$%^&*(),.?":{}|<>]).
- **hasMinLength** = true if the password has a length between 8 and 15 characters (inclusive).

```
bool passwordRequirements(String password) {
  bool hasUppercase = password.contains(RegExp(r'[A-Z]'));
  bool hasDigits = password.contains(RegExp(r'[0-9]'));
  bool hasLowercase = password.contains(RegExp(r'[a-z]'));
  bool hasSpecialCharacters =
  password.contains(RegExp(r'[!@#$%^&*(),.?":{}|<>]'));
  bool hasMinLength = (password.length >= 8 && password.length <=15);

  return hasDigits &
  hasUppercase &
  hasLowercase &
  hasSpecialCharacters &
  hasMinLength;
}
```

## Code Versioning

During the development phase, GitHub was used as the version controlling tool. The use of version control with GitHub allows for maintaining a history of code modifications, making it easier to revert to previous versions if needed and ensuring that all team members were working on the latest codebase

# Chapter 6: Test Plan

## 6.1. Introduction

The aim of this chapter is to discuss the testing plan for the Sniffer mobile application and GPS tracker. This chapter outlines the test features (in-scope and out of scope features), test types and methodologies, testing environment used to test the software's main functionalities, test data and test case results.

## 6.2. Test Plan

### 6.2.1 Introduction

This section focuses on outlining the test items, in-scope features, and out-of-scope areas. Additionally, it defines the test strategy, metrics, entry and exit criteria, test deliverables, and environmental needs.

### 6.2.2. Test Items

1. Pet Tracker Mobile Application
2. GPS Tracker

#### 6.2.2.1 In-Scope Features

The listed features represent the major functional requirements of the software to be tested in this document:

- **Mobile Application:**
    - Create User Account
    - Login
    - Edit profile details
    - Add a pet
    - Edit pet details
    - Delete a pet
    - Create a geofence
    - View geofence
    - View pet
    - Delete a geofence
    - Register a tracker
    - Edit tracker details
    - Delete tracker
    - View pet's live location
    - Receive lost pet notifications
- **Tracker**
    - Send GPS Coordinates

### 6.2.4 Out-of-Scope

This test plan will not cover Unit testing, Security or Performance testing, nor Third-party integration (such as the Google Maps API). The focus of the test document is to test the app's own functionalities while assuming that the APIs work as expected.

### 6.2.5 Test Strategy

#### 6.2.5.2 Test Metrics

$$RunRate = \frac{ExecutedTestCases}{Total\ Test\ Cases}$$

$$PassRate = \frac{SuccessfulTestCases}{ExecutedTestCases}$$

### 6.2.5. Entry Criteria

1. Main requirements are implemented.
2. Test cases and test scenarios are prepared and reviewed.
3. Test environment and infrastructure, including devices/emulators and network connectivity, are set up.

### 6.2.8 Exit Criteria

1.100% of test cases have been executed
2. Pass rate >= 90%

### 6.2.9 Test Deliverables

1. Test plan & test cases
2. Test Execution Document

### 6.2.10 Environmental Needs

1. Hardware: Test devices or emulators that support the Flutter framework and are compatible with the target operating systems (e.g., Android, iOS).
2. Software: Flutter SDK, development environment (e.g., Android Studio, Visual Studio Code), necessary plugins and dependencies for Flutter development.

## 6.3 Test Execution Document

### 6.3.1 Introduction

The Test Execution Document provides an overview of the test execution process for the Sniffer app. This document outlines the approach, objectives, and scope of functional testing.

### 6.3.2 Functional Testing

#### 1. Black box testing

- **Static Testing:**
  - The high-level user requirements and low-level specifications have undergone a thorough review process to identify any ambiguities or inaccuracies and ensuring their clarity and correctness.
- **Dynamic Testing:**

#### Equivalence Partitioning

*Table 17: Boundary Equivalence Partitioning*

| Condition | Valid Partition | Valid Boundary | Invalid Partition | Invalid Boundary |
|---|---|---|---|---|
| **Email address** | has @<br>has .com | - | without @<br>without .com | - |
| **Password** | between 8-15 characters<br>at least one upper-case<br>at least one number<br>at least one special character | 8 characters<br>15 characters | <8 characters | |
| **Pet profile** | Integer age input<br>All fields entered | - | Non-integer age<br>Empty fields | - |
| **Geofence** | >=3 coordinates on map<br><=8 coordinates on map<br>No intersecting points | 3 coordinates<br>8 coordinates | <3 coordinates on map<br>Intersecting points | 2 coordinates<br>9 coordinates |

## Data Testing

*Table 18: Input Data Testing for Equivalence Partition*

| Test | Test case | Test data | Expected result | Actual result | Pass/ Fail |
|---|---|---|---|---|---|
| **Create a geofence** | 2 coordinates | 2 Points on map | Invalid | Invalid | Pass |
| | 3 coordinates, no intersecting points | 3 non-intersecting points on map | Valid | Valid | Pass |
| | 8 coordinates, no intersecting points | 8 non-intersecting points on map | Valid | Valid | Pass |
| | 8 coordinates with intersecting points | 8 points on map – at least 2 intersecting | Invalid | Invalid | Pass |
| | 3 coordinates with intersecting points | 3 intersecting points on map | Invalid | Invalid | Pass |
| | 9 coordinates | 9 points on map | Invalid | Invalid | Pass |
| **Register - Email address** | 7 characters | a@h.com | Invalid | Invalid | Pass |
| | 8 characters, with @ and .com | a@hi.com | Valid | Valid | Pass |
| | 80 characters with @ and .com | Somecompanysverylongemailaddressboundarytestingthesizeofemail@info-contactus.com | Valid | Valid | Pass |
| | 81 characters | somecompanysverylongemailaddressboundarytestingthesizeofemails@info-contactus.com | Invalid | Invalid | Pass |
| | 8 character, no @, with .com | test.com | Invalid | Invalid | Pass |
| | 8 character, with @, no .com | test@com | Invalid | Invalid | Pass |
| | 8 characters, no @, no .com | testmail | Invalid | Invalid | Pass |
| | 80 character, no @, with .com | Somecompanysverylongemailaddressboundarytestingthesizeofemailsinfo-contactus.com | Invalid | Invalid | Pass |
| | 80 character, with @, no .com | Somecompanysverylongemailaddressboundarytestingthesizeofemail@info-contactusmail | Invalid | Invalid | Pass |
| | 80 characters, no @, no .com | Somecompanysverylongemailaddressboundarytestingthesizeofemailsinfo-contactusmail | Invalid | Invalid | Pass |
| **Register - Password** | 7 characters, with uppercase, number, special character | Test12@ | Invalid | Invalid | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | 8 characters, with uppercase, number, special character | Test123@ | Valid | Valid | Pass |
| | 15 characters, with uppercase, number, special character | Testing@1234567 | Valid | Valid | Pass |
| | 16 characters, with uppercase, number, special character | Testing@12345678 | Invalid | Invalid | Pass |
| | 8 characters, no upper-case, with special character, with number | test1ng@ | Invalid | Invalid | Pass |
| | 8 characters, with uppercase, special character, without number | tesTing@ | Invalid | Invalid | Pass |
| | 8 characters, with uppercase, number, without special character | Testing123 | Invalid | Invalid | Pass |
| | 8 characters, no upper-case, no special character, with number | test1ng2 | Invalid | Invalid | Pass |
| | 8 characters, no uppercase, special character, without number | testing@ | Invalid | Invalid | Pass |
| | 8 characters, with uppercase, no number, no special character | Password | Invalid | Invalid | Pass |
| | 8 characters, no uppercase, | password | Invalid | Invalid | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | no number, no special character | | | | |

## API Testing

Testing the POST request to the Google Cloud Function to write GPS coordinates to Firestore.

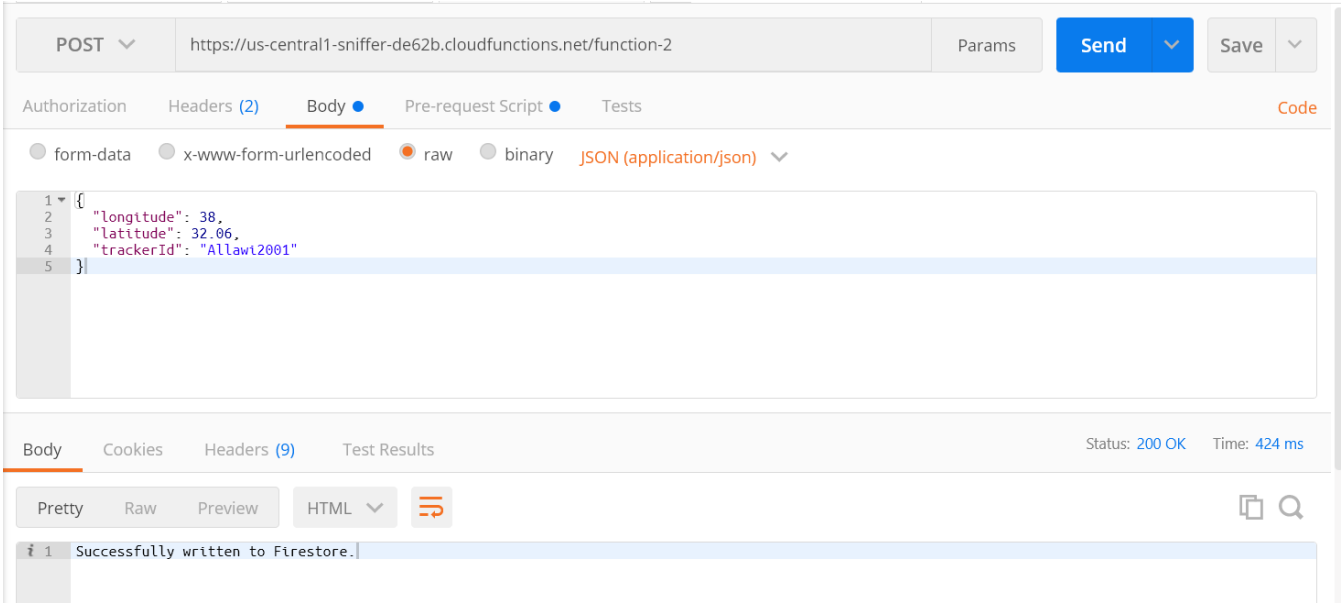



*Figure 16: Sample successful POST request to the Google Cloud Function through Postman*

## State-Transition Testing

The state-transition diagram below illustrates the progression of the tracker's state from the unassigned/disabled state, where no pet is assigned to the tracker, to the idle state, indicating that a pet is assigned and is currently within the geofence. It also highlights the transition to the trigger alerts state, triggered when the assigned pet goes outside the geofence.

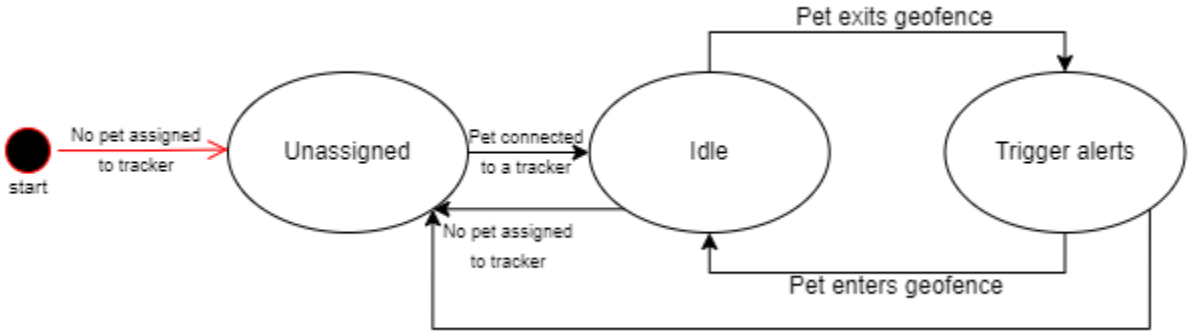



*Figure 17: State-Transition Diagram of Sniffer*

The test cases derived from the state-transition diagram focus on validating the correct behavior and transitions between different states of the tracker.

*Table 19: Test cases derived from state-transition diagram*

| **Test Case** | **TC1** | **TC2** | **TC3** | **TC4** | **TC5** |
|---|---|---|---|---|---|
| **Start State** | **Unassigned** | **Idle** | **Idle** | **Trigger Alerts** | **Trigger Alerts** |
| **Input** | Assign pet | Pet exits fence | Remove fence from pet | Pet enters fence | Remove fence from pet |
| **Finish State** | Idle | Trigger Alerts | Unassigned | Idle | Unassigned |

*Table 20: Description of the state-transition test cases and their expected results vs actual results*

| **Test Case ID** | **Steps** | **Expected Results** | **Actual Results** |
|---|---|---|---|
| **TC1** | 1. User logs into their Sniffer account<br>2. User adds a new tracker<br>3. User assigns pet to tracker | Tracker is in idle state | Tracker is in idle state |
| **TC2** | 1. User already has a pet assigned to a tracker<br>2. Tracker exits the geofence<br>3. User receives alerts that their pet exited the geofence | Alerts triggered | Alerts triggered |
| **TC3** | 1. User already has a pet assigned to a tracker<br>2. User removes pet's assigned geofence<br>3. Tracker is unassigned | Tracker is in unassigned/disabled state | Tracker is in unassigned/disabled state |
| **TC4** | 1. User already has a pet assigned to tracker and | Alerts stop. Tracker is in idle state | Alerts stop. Tracker is in idle state |

| | | | |
|---|---|---|---|
| | pet is outside the geofence<br>2. User receives alerts that their pet exited the geofence<br>3. Pet enters geofence again | | |
| **TC5** | 1. User already has a pet assigned to tracker and pet is outside the geofence<br>2. User receives alerts that their pet exited the geofence<br>3. User removes the pet's assigned geofence<br>4. Tracker is unassigned | Tracker is in unassigned/disabled state | Tracker is in unassigned/disabled state |

## Traceability Matrix

The traceability matrix below provides a clear mapping between the functional requirements and the corresponding test cases. Test cases 6-22 can be found in Appendix A:

*Table 21: Traceability matrix between functional requirements and test cases*

| Test case / Req ID | TC1 | TC2 | TC3 | TC4 | TC5 | TC6 | TC7 | TC8 | TC9 | TC10 | TC11 | TC12 | TC13 | TC14 | TC15 | TC16 | TC17 | TC18 | TC19 | TC20 | TC21 | TC22 | # Test cases for the REQ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REQ-1 | | | | | | X | X | | X | X | | | | | | | | | | | | | 4 |
| REQ-2 | | | | | | | | X | | | X | | | | | | | | | | | | 2 |
| REQ-3 | | | | | | | | | | | | X | X | | | | | | | | | | 2 |
| REQ-4 | | | | | | | | | | | | X | | | | | | | | | | | 1 |
| REQ-5 | | | | | | | | | | | | | | | | | | | | | X | | 1 |
| REQ-6 | | | | | | | | | | | | | | | | | | | | | X | | 1 |
| REQ-7 | | | | | | | | | | | | | | X | | | | | | | | | 1 |
| REQ-8 | | | | | | | | | | | | | | X | | | | | | | | | 1 |
| REQ-9 | | | | | | | | | | | | | | | X | | | | | | | | 1 |
| REQ-10 | | | | | | | | | | | | | | | | X | X | | | | | | 2 |
| REQ-11 | | X | | X | X | | | | | | | | | | | | | | | | | | 3 |
| REQ-12 | | | | | | | | | | | | | | | | | | | | | | X | 1 |
| REQ-13 | | | | | | | | | | | | | | | | | | | | | | X | 1 |
| REQ-14 | X | | | | | | | | | | | | | | | | | X | | | | | 2 |
| REQ-15 | X | | X | | | | | | | | | | | | | | | X | X | | | | 4 |
| REQ-16 | | | X | | X | | | | | | | | | | | | | | X | | | | 3 |
| REQ-17 | | | | | | | | | | | | | | | | | | | | X | | | 1 |

## 2. White-Box Testing

### Static Analysis

Static analysis was performed on the Sniffer Flutter application using the Dart Code Metrics package. The results provide valuable insights into various metrics that help evaluating the quality of the code.

The cyclomatic complexity metric, which measured at 400, alerts to areas of high complexity that require attention. We acknowledged the need to refactor and simplify those sections to enhance code comprehension and maintainability. The source lines of code metric, measuring 3633, gauges the size of the codebase and better estimate the efforts required for maintenance and future enhancements.

The maintainability index, scoring 71, indicates that the codebase is reasonably maintainable by adhering to best practices and coding standards.

Additionally, the number of arguments metric is 1, indicating a low complexity in passing arguments to functions. The maximum nesting is reported as 2, suggesting a moderate level of code nesting. Notably, the technical debt is reported as 0, indicating that the codebase is free from known technical debt issues.

**All files**

| Directory | Cyclomatic complexity | Source lines of code | Maintainability index | Number of Arguments | Maximum Nesting | Technical Debt |
|---|---|---|---|---|---|---|
| lib | 21 | 132 | 69 | 1 | 2 | 0.0 |
| lib\models | 37 | 82 | 65 | 1 | 1 | 0.0 |
| lib\providers | 67 | 357 | 72 | 2 | 2 | 0.0 |
| lib\screens | 85 | 1049 | 74 | 1 | 2 | 0.0 |
| lib\screens\onBoarding_screens | 9 | 82 | 81 | 1 | 1 | 0.0 |
| lib\widgets | 181 | 1931 | 68 | 1 | 2 | 0.0 |

Cyclomatic complexity : **400**

Source lines of code : **3633**

Maintainability index : **71**

Number of Arguments : **1**

Maximum Nesting : **2**

Technical Debt : **0**

*Figure 18: Snapshot of Dart Code Metrics Analysis results of the mobile application*

# Chapter 7: Conclusion & Future Works

Sniffer has provided an effective solution for pet owners to monitor and track their pets' whereabouts. The app incorporates various features, including real-time location tracking, geofence monitoring, and notifications, to ensure the safety and well-being of pets. Throughout the development process, a comprehensive set of functional requirements and corresponding test cases were defined to ensure the app's reliability and effectiveness.

The implementation of Sniffer involved integrating technologies such as Flutter for the mobile app, Firebase for backend services, and geofencing algorithms in the cloud for monitoring the pet's location. The app successfully meets the specified functional requirements, allowing users to assign pets to the tracker, track their location in real-time, define geofences, and receive notifications when their pets cross the geofence boundaries.

Although Sniffer has achieved its core objectives, there are several areas for potential future enhancements and expansions:

- Enhanced User Interface: The app's user interface can be further refined and improved to provide a more intuitive and user-friendly experience. User feedback and usability testing can help identify areas for improvement and optimize the app's interface design.

- Additional Features: Consider adding additional features based on user feedback and market demands. For example, integrating a health monitoring system that tracks vital signs or implementing a social component where pet owners can connect and share their experiences.

- Multi-Platform Support: Extend Sniffer's support to other platforms, such as web or desktop, to reach a wider audience and provide a seamless experience across multiple devices.

- Improve performance of the system: Refactor parts of the application that may be complex and cause performance issues (take a long time to render)

# Appendix A: Test Cases

*Table 22: Test Case 6*

| Field Name | Description |
|---|---|
| Test Case ID | TC06 |
| Test Title | Successful User Registration |
| Test Area | User Registration Activity |
| Pre-condition | - The user is on the registration screen<br>In each text field:<br>- User enters a valid email<br>- User enters their first and last name<br>- User enters a password with >8 characters, at least one number and one special character<br>- User confirms their password by entering it again |
| Expected Result | User account is successfully created and user is sent to email verification page |
| Actual Result | User account is successfully created and user is sent to email verification page |

*Table 23: Test Case 7*

| Field Name | Description |
|---|---|
| Test Case ID | TC07 |
| Test Title | Verify Email |
| Test Area | Verification |
| Pre-condition | User has registered an account and a verification email has been sent. |
| Expected Result | User can successfully verify their email address using the sent confirmation link. |
| Actual Result | User can successfully verify their email address using the sent confirmation link |

*Table 24: Test Case 8*

| Field Name | Description |
|---|---|
| Test Case ID | TC08 |
| Test Title | User Login |
| Test Area | Login Acitivity |
| Pre-condition | - User has a registered account with Sniffer<br>- User is on the registration screen<br>- User enters their email and password correctly |
| Expected Result | User is successfully logged in with the valid credentials |
| Actual Result | User is successfully logged in with the valid credentials |

*Table 25: Test Case 9*

| Field Name | Description |
|---|---|
| Test Case ID | TC09 |
| Test Title | Add user profile picture |
| Test Area | User profile |
| Pre-condition | - User is logged in and on the edit profile screen<br>- User adds a profile picture |
| Expected Result | User profile is updated and photo is added successfully |
| Actual Result | User profile is updated and photo is added successfully |

*Table 26: Test Case 10*

| Field Name | Description |
|---|---|
| Test Case ID | TC10 |
| Test Title | Edit User Details |
| Test Area | User profile |
| Pre-condition | - User is logged in and on the edit profile screen<br>- User edits their name/email/resets password |
| Expected Result | User profile is updated successfully |
| Actual Result | User profile is updated successfully |

*Table 27: Test Case 11*

| Field Name | Description |
|---|---|
| Test Case ID | TC11 |
| Test Title | Reset password |
| Test Area | User profile |
| Pre-condition | - User already has registered an account and is on the login screen<br>- User clicks on "forgot password" button<br>- User enters their email into the textfield<br>- User recieves an email with a link to reset password<br>- User clicks on the link and chooses a new password |
| Expected Result | User password is reset successfully |
| Actual Result | User password is reset successfully |

*Table 28: Test Case 12*

| Field Name | Description |
|---|---|
| Test Case ID | TC12 |
| Test Title | Add New Pet |
| Test Area | Pet profile |
| Pre-condition | - User is logged in and on the pet profile creation screen ("New pet").<br>- User adds all required pet information (name, age, description, and uploads a photo) |
| Expected Result | Pet profile is successfully created with valid inputs. |
| Actual Result | Pet profile is successfully created with valid inputs. |

*Table 29: Test Case 13*

| Field Name | Description |
|---|---|
| Test Case ID | TC13 |
| Test Title | Create Pet Profile - Missing Required Fields |
| Test Area | Pet profile |
| Pre-condition | - User is logged in and on the pet profile creation screen.<br>- User does not provide at least one of the required fields (name, age, description, or photo) |
| Expected Result | User receives an error message indicating that the required fields are missing. |
| Actual Result | User receives an error message indicating that the required fields are missing. |

*Table 30: Test Case 14*

| Field Name | Description |
|---|---|
| Test Case ID | TC14 |
| Test Title | Create a geographical fence |
| Test Area | Geofence |
| Pre-condition | - User is logged in and on the create geographical fence screen.<br>- User marks at least 3 points on the map that do not intersect<br>- User adds a name in the fence name field |
| Expected Result | Geographical fence is successfully created and named |
| Actual Result | Geographical fence is successfully created and named |

*Table 31: Test Case 15*

| Field Name | Description |
|---|---|
| Test Case ID | TC15 |
| Test Title | Delete geographical fence |
| Test Area | Geofence |
| Pre-condition | - User is logged in and on the geofence detail screen<br>- User clicks on the delete icon |
| Expected Result | Geofence is deleted successfully |
| Actual Result | Geofence is deleted successfully |

*Table 32: Test Case 16*

| Field Name | Description |
|---|---|
| Test Case ID | TC16 |
| Test Title | Assign Geofence to Pet |
| Test Area | Geofence/Pet |
| Pre-condition | - User is logged in and on the pet detail screen<br>- User clicks on the existing geofence or creates a new geofence |
| Expected Result | Geofence is successfully assigned to pet |
| Actual Result | Geofence is successfully assigned to pet |

*Table 33: Test Case 17*

| Field Name | Description |
|---|---|
| Test Case ID | TC17 |
| Test Title | Change pet's geofence |
| Test Area | Pet profile/Geofence |
| Pre-condition | - User is logged in and on the pet details page<br>- User clicks on the change geofence button<br>- User chooses a geofence<br>- Confirmation dialog appears<br>- User clicks on the confirm button |
| Expected Result | Pet geofence is updated successfully |
| Actual Result | Pet geofence is updated successfully |

*Table 34: Test Case 18*

| Field Name | Description |
|---|---|
| Test Case ID | TC18 |
| Test Title | Add New Tracker |
| Test Area | Tracker |
| Pre-condition | - User is logged in and on the "Add new tracker" page.<br>- User enters the tracker ID and clicks on "check tracker ID"<br>- User enters tracker title and chooses a pet to assign<br>- User clicks on "Add tracker" button |
| Expected Result | Tracker is added successfully and appears on the trackers page |
| Actual Result | Tracker is added successfully and appears on the trackers page |

*Table 35: Test Case 19*

| Field Name | Description |
|---|---|
| Test Case ID | TC19 |
| Test Title | Edit Tracker |
| Test Area | Tracker |
| Pre-condition | - User is logged in and on the Trackers page.<br>- User clicks on the "edit" icon<br>- User changes the assigned pet<br>- User clicks on the "save changes" button |
| Expected Result | Tracker is added successfully updated and appears on the trackers page |
| Actual Result | Tracker is added successfully updated and appears on the trackers page |

*Table 36: Test Case 20*

| Field Name | Description |
|---|---|
| Test Case ID | TC20 |
| Test Title | Delete Tracker |
| Test Area | Tracker |
| Pre-condition | - User is logged in and on the Trackers page<br>- User clicks on the "Delete" icon of the tracker<br>- User clicks "Delete" on the confirmation dialog |
| Expected Result | Tracker is successfully deleted |
| Actual Result | Tracker is successfully deleted |

| Field Name | Description |
|---|---|
| Test Case ID | TC21 |
| Test Title | Edit Pet Details |
| Test Area | Pet profile |
| Pre-condition | - User is logged in and already has at least one pet added<br>- User clicks on the Pets page<br>- User clicks on the pet<br>- User clicks on the "edit" button<br>- User edits pet details (image, name, age, description) |
| Expected Result | Pet profile is updated successfully and appears on the Pets page |
| Actual Result | Pet profile is updated successfully and appears on the Pets page |

*Table 37: Test Case 22*

| Field Name | Description |
|---|---|
| Test Case ID | TC22 |
| Test Title | Receive Alert Triggers |
| Test Area | Geofence/Tracker |
| Pre-condition | - User is logged in and already has at least one pet added and assigned a geofence and a tracker<br>- Pet exits the geofence<br>- User receives alerts about the lost pet<br>- User confirms pet has been found |
| Expected Result | User receives notifications |
| Actual Result | User receives notifications |

# Appendix B: Records of Change

# Chapter 3: Software Requirements Specifications

### Functional Requirements

*Table 3838: All changes and modifications in the functional requirements*

| Requirement ID | Requirement Name | Requirement Detail | Priority | Category for Change | Reason Behind Change |
|---|---|---|---|---|---|
| **REQ-1** | UserAccount.Registration | Create user account with username, email, password | High | Modified | Username requirement removed User now creates an account with their first & last names, email, and password. |
| **REQ-1.1** | UserAccount.Registration.Username | User chooses a unique username | Medium | Removed | Feature found unnecessary at this time |
| **REQ-1.2** | UserAccount.Registration.Password | User chooses a strong password | High | Not Modified | - |
| **REQ-1.3** | UserAccount.Registration.Verification | System validates user email using confirmation link | High | Not Modified | - |
| **REQ-2** | UserAccount.Login | User logs into the system with username and password | High | Modified | Login is through email & password instead of username |
| **REQ-2.1** | UserAccount.Login.KeepLoggedIn | User remains in “logged in” state until logged out | Medium | Not Modified | - |
| **REQ-3** | PetProfile.Create | User creates pet profile | High | Not Modified | - |
| **REQ-3.1** | PetProfile.Create.MultipleProfiles | User creates multiple pet profiles | Medium | Modified | Allowing users to add unlimited pet profiles instead of 3 max. |
| **REQ-4** | PetProfile.View | User views any created pet profile | Medium | Not Modified | - |
| **REQ-5** | PetProfile.Edit | User edits any pet profile | High | Not Modified | - |
| **REQ-6** | PetProfile.Delete | User deletes any pet profile | Medium | Not Modified | - |
| **REQ-7** | Pet.CreateGeographicalFence | User defines geofence | High | Not Modified | - |

| | | | | | |
|---|---|---|---|---|---|
| **REQ-8** | Pet.NameGeographicalFence | User names geofence | Medium | Not Modified | - |
| **REQ-9** | Pet.DeleteGeographicalFence | User deletes any geofence | High | Not Modified | - |
| **REQ-10** | Pet.AssignGeofenceToPet | User assigns a geofence to a pet | High | Not Modified | - |
| **REQ-11** | Pet.RecieveLostPetAlerts | User receives alerts if pet exits geofence | High | Not Modified | - |
| **REQ-12** | Pet.MonitorLocation | User can monitor pet location | High | Not Modified | - |
| **REQ-13** | Pet.ConfirmFound | User confirms pet has been found | High | Not Modified | - |
| **REQ-14** | Tracker.AddTracker | User can add a new tracker | High | New | Users need a way to add trackers |
| **REQ-15** | Tracker.AssignPet | User assigns pet to tracker | High | New | Users need a way to assign a pet to a tracker |
| **REQ-16** | Tracker.Edit | User can edit tracker name/assigned pet | Medium | New | Users need a way to edit the currently assigned pet |
| **REQ-17** | Tracker.Delete | User can delete the added tracker | Medium | New | Users need a way to delete a tracker that they add |

## Chapter 4: System Design

### Class Diagram

Before: Our initial class diagram represents our limited experience in hardware device development and mobile application development. It serves as a visual depiction of our understanding and knowledge at the beginning stages of the project. This diagram may lack complexity and completeness, reflecting the initial learning curve and exploration of the involved classes and their relationships

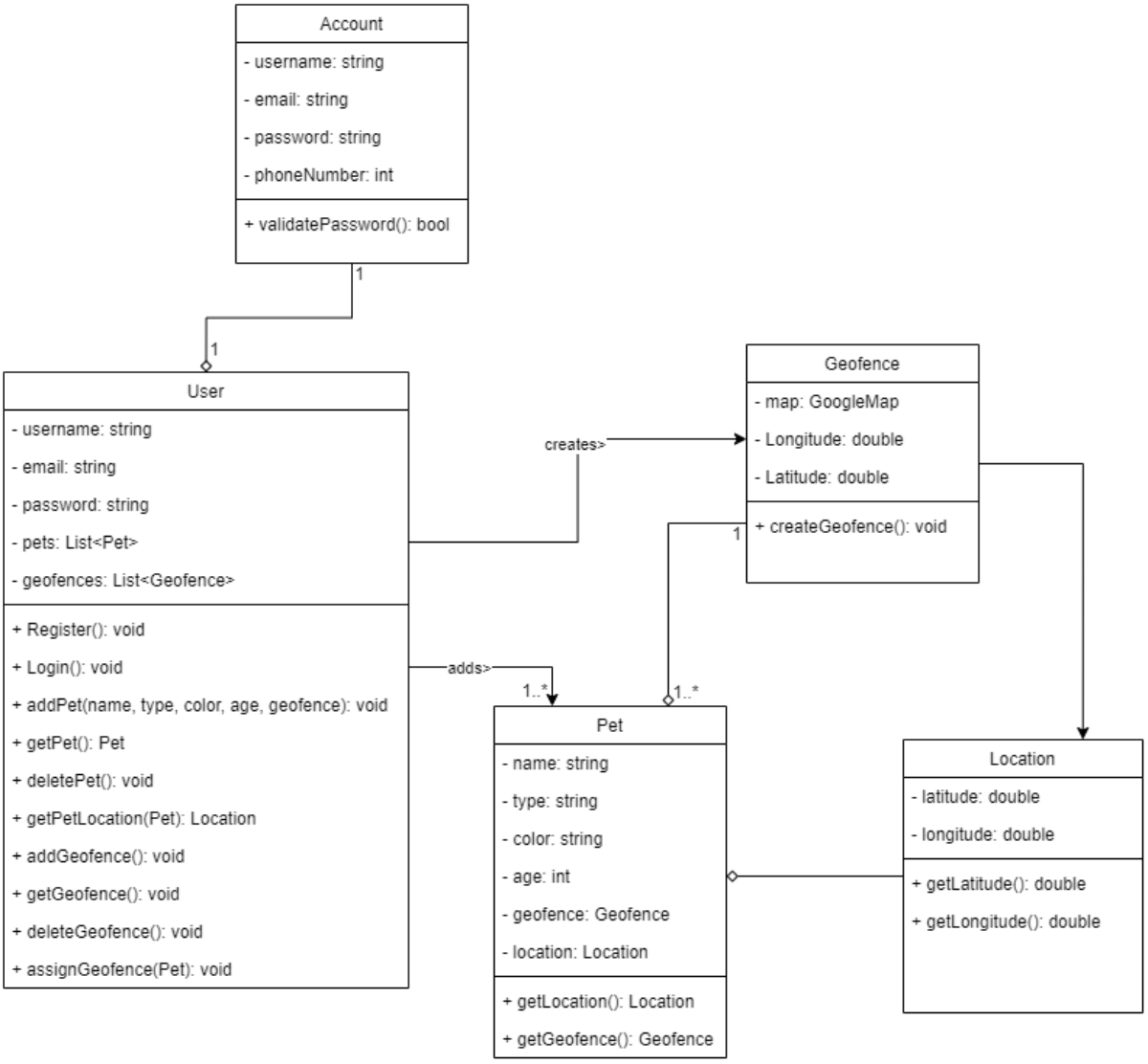



***Figure 19: Class diagram before editing***

After

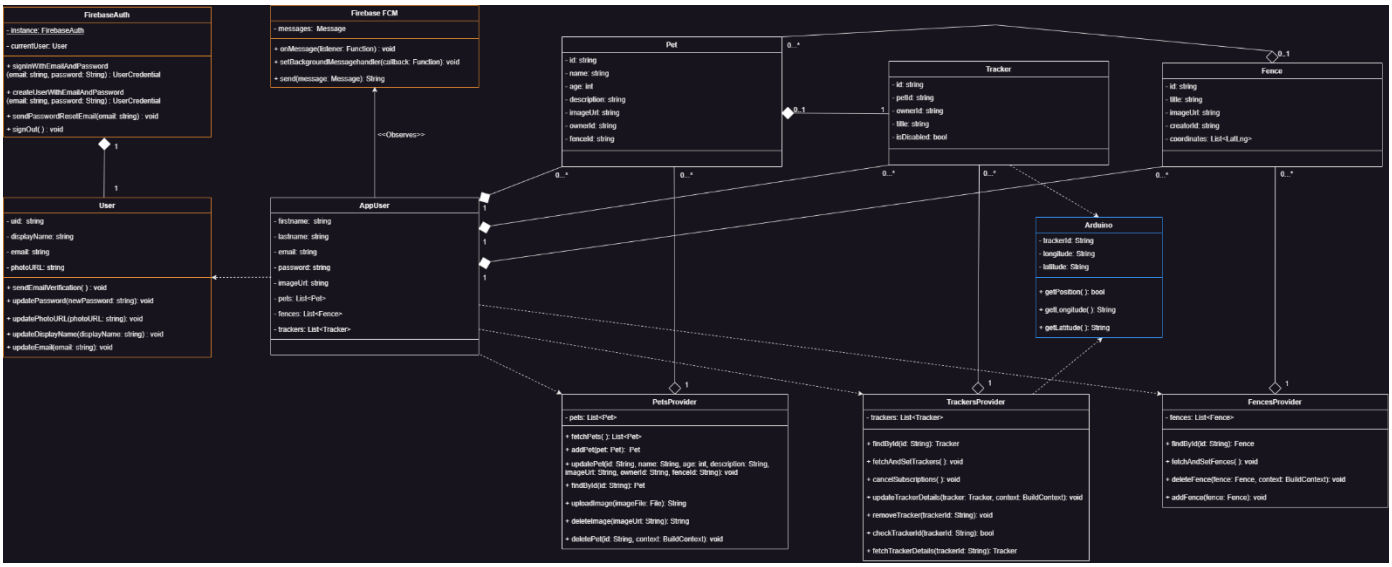



***Figure 20: Class diagram after editing***

This system is designed to manage a pet tracking application. At its core, the system leverages Firebase's FirebaseAuth and FirebaseUser classes to handle user authentication. FirebaseAuth is a singleton class and manages user logins, account creation, password resets, and email verification. Each FirebaseAuth instance contains a FirebaseUser object representing the currently logged in user. FirebaseUser holds user-specific information like uid, email, display name, and photo URL, and provides methods for updating this information. Once authenticated, a user is represented within the app as an AppUser object. This object stores information about the user (including name, email, password, and image URL) and keeps track of the user's pets, fences, and trackers.

Each pet in the system is represented as a Pet object, and each user may have multiple pets. Pets have properties such as id, name, age, and imageUrl. They also have an ownerId, linking them back to their owner. Similarly, the user's geofences (Fences) and tracking devices (Trackers) are also stored as lists of Fence and Tracker objects.

Pets, Fences, and Trackers are all managed by their respective provider classes - PetsProvider, FencesProvider, and TrackersProvider. These provider classes are responsible for performing operations on their respective data types, such as fetching, adding, updating, and deleting.

Each Tracker is linked to an Arduino/Tracker Hardware device. The Arduino/Tracker Hardware is represented by its own class in the system. This represents the physical device used for tracking pets and communicates with the TrackerProvider to provide location data.

The relationships between these classes involve various types of class associations including composition (e.g., a Pet cannot exist without an AppUser), aggregation (e.g., PetsProvider manages multiple Pets), and dependency (e.g., TrackersProvider uses Arduino/Tracker Hardware).

In essence, the system leverages Firebase for user authentication and then represents users, their pets, and tracking devices as objects within the app. Provider classes manage these objects, allowing the user to perform various operations related to pet tracking.

**Use Case Diagram**

Before: The initial use case diagram of our system represents a significant portion of its functionalities but may still be missing some. It serves as a visual representation of the primary use cases that we have identified at the early stage of design.

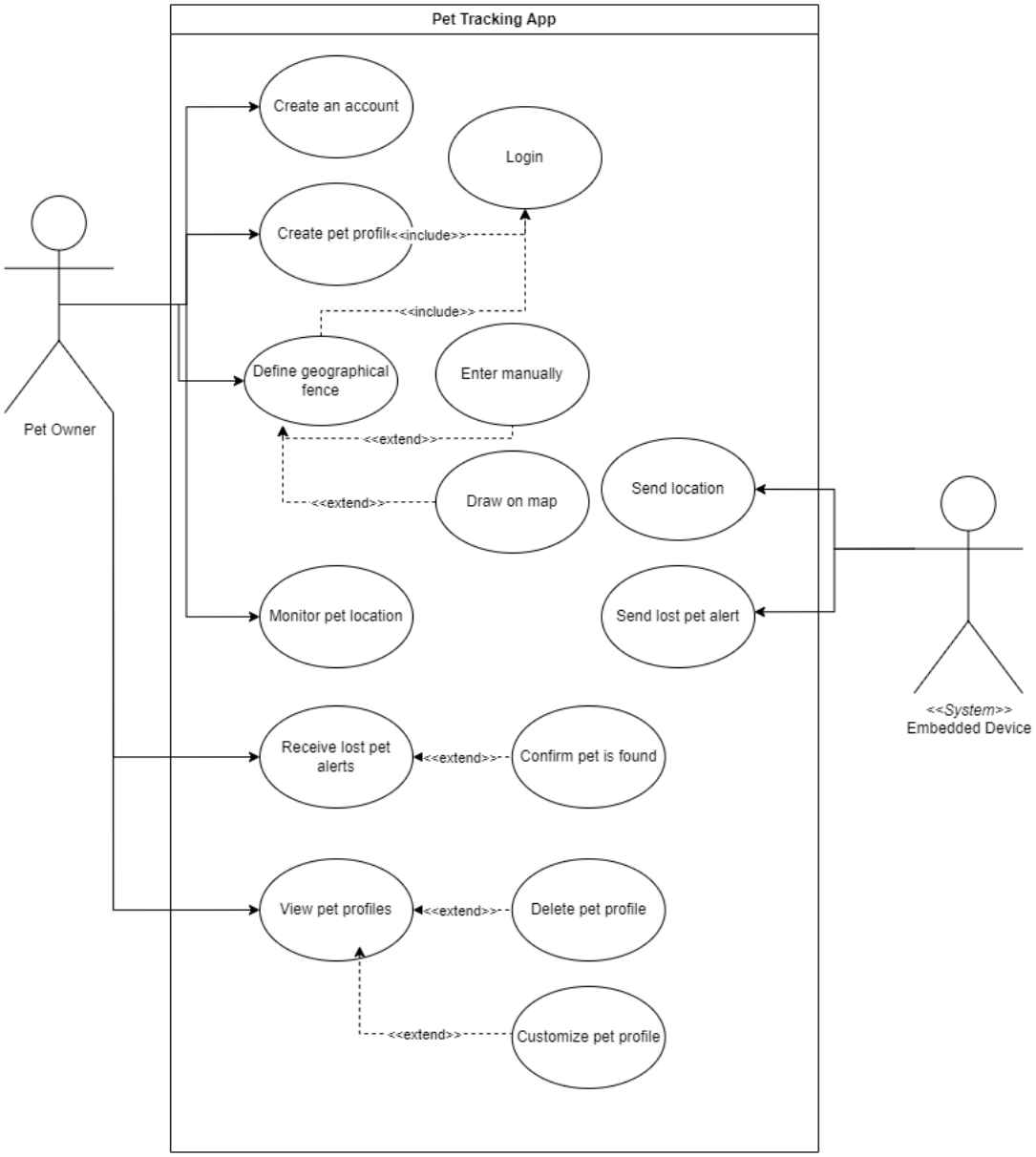



***Figure 21: Initial use case diagram***

After: As the scope of the project became clearer and the requirements were refined, we were able to identify and include all the necessary use cases. The diagram now depicts the key functionalities such as adding, viewing, and editing trackers, pets, and geofences.

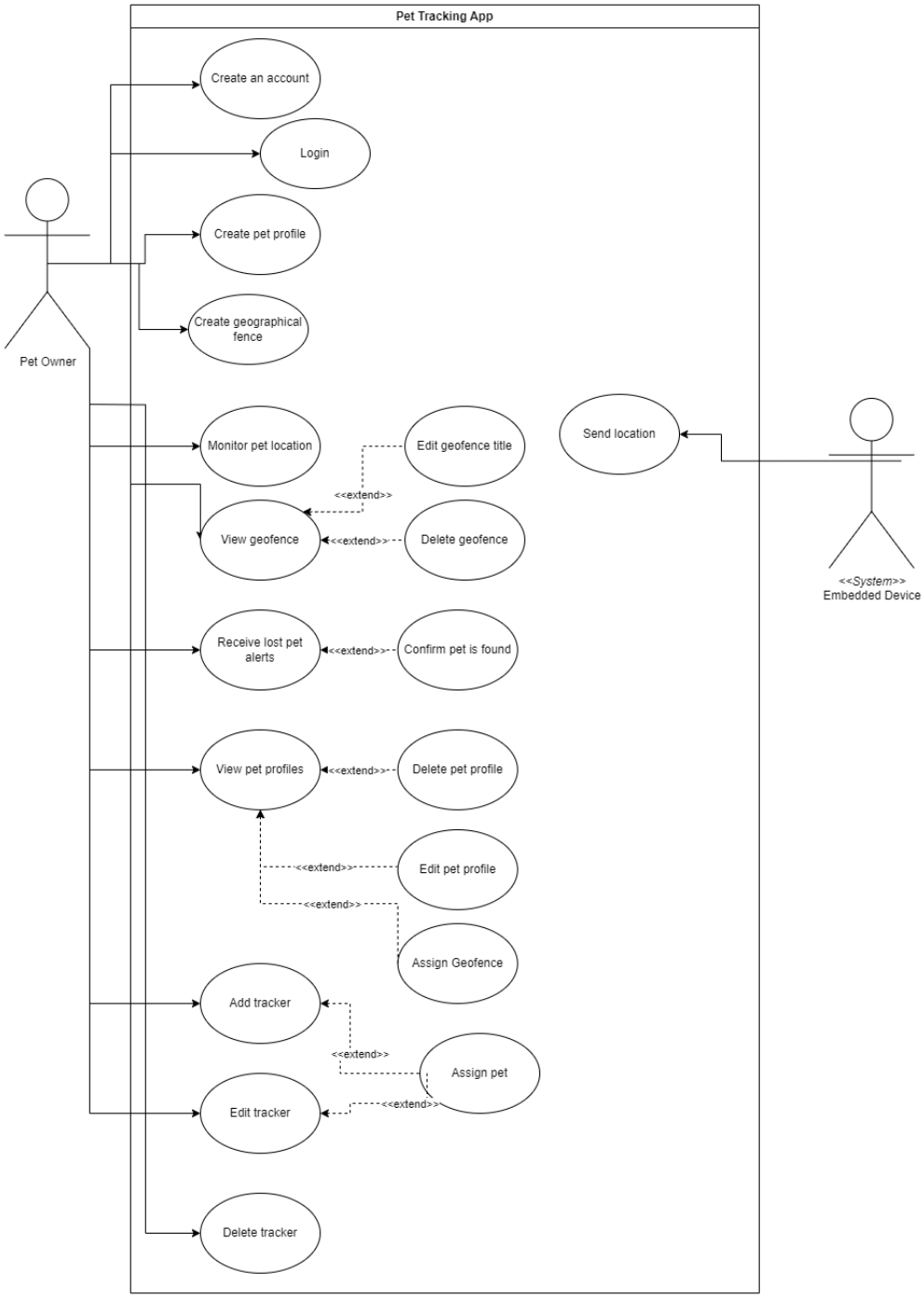



***Figure 22: Use case diagram after editing***

**UI Prototype:**

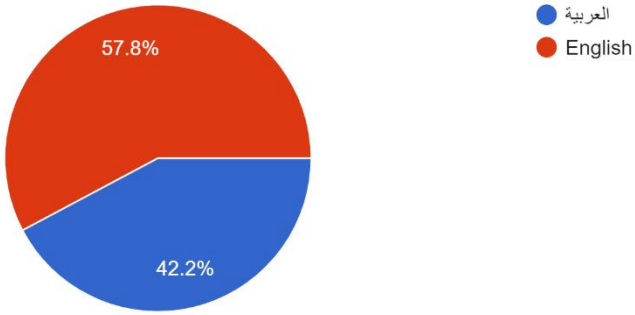
Refer to User Manual (Appendix D) for final User Interface Design

# Appendix C:

## Requirements Elicitation Survey Results:

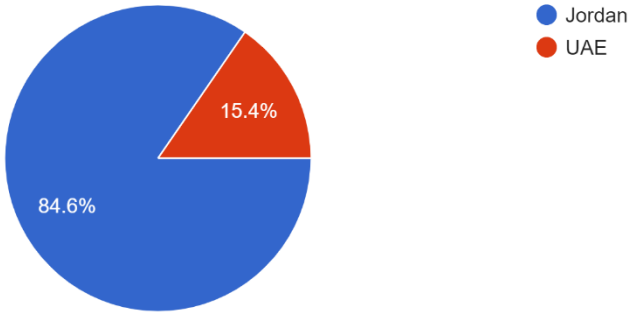
Choose the language اختر اللغة

45 responses



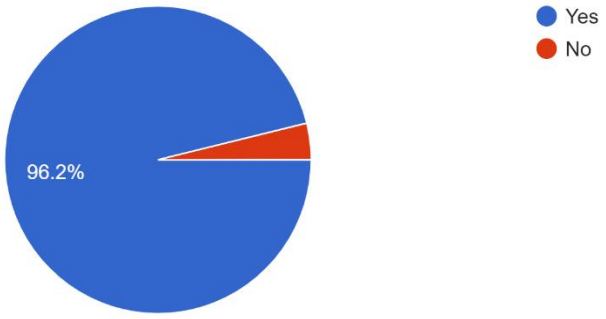

Country?

26 responses



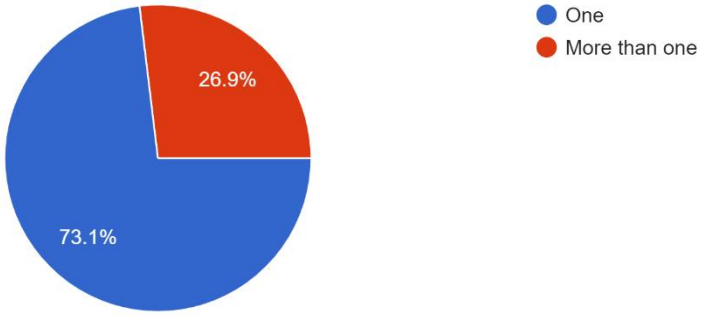
## Are you a pet owner?

26 responses



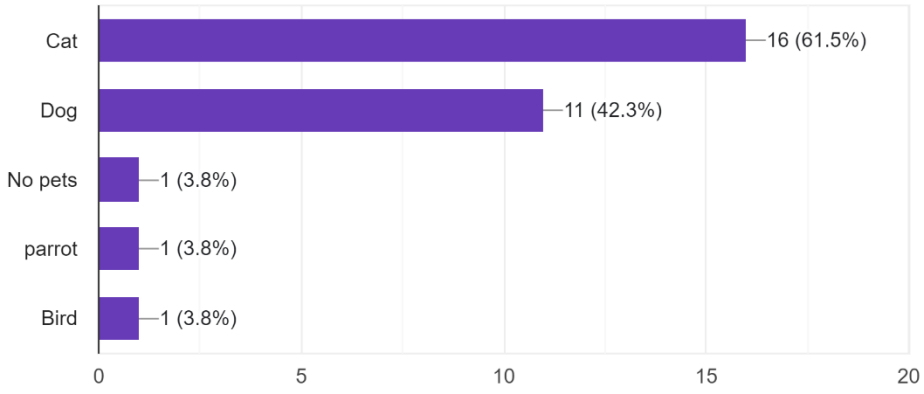

## How many pet(s) do you own?

26 responses



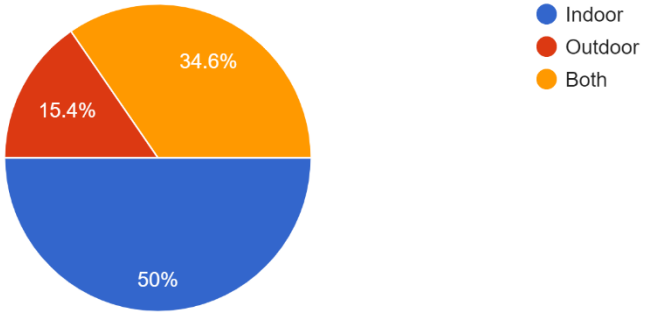

## What kind of pet(s) do you own?

26 responses



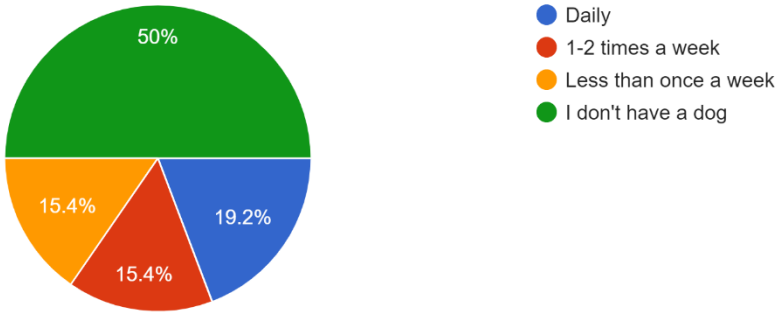
Is your pet indoor or outdoor pet?

26 responses



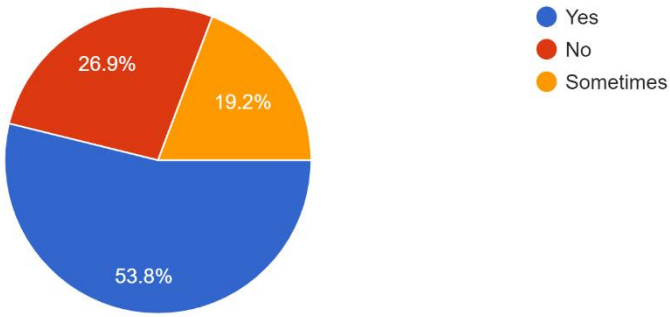

If you have a dog, how often is it off leash outdoors? whether it was in your front yard or in an off-leash dog park.

26 responses



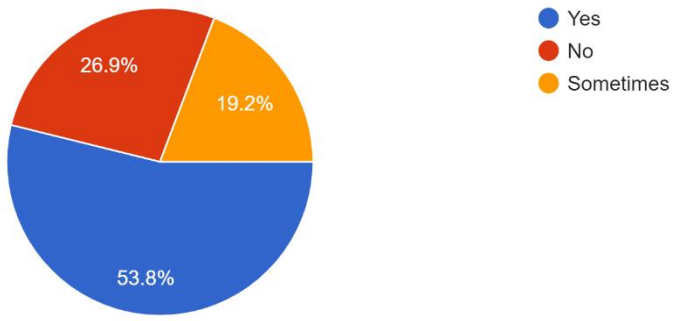

Does your pet wear a collar?

26 responses



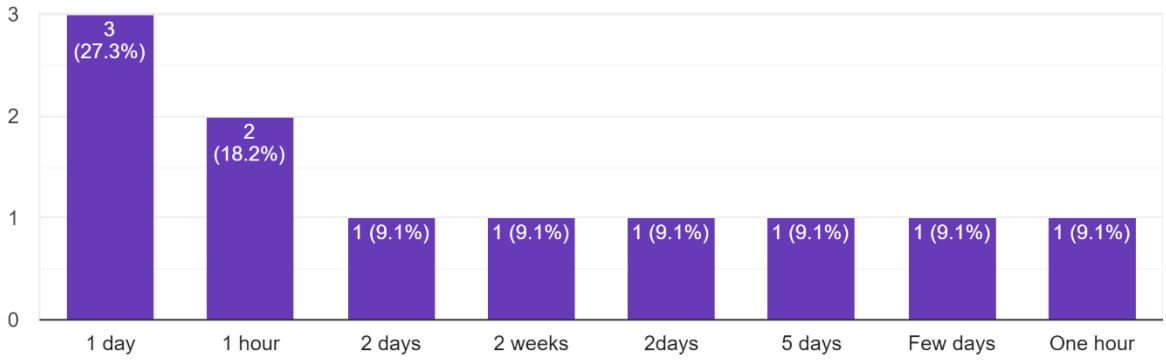
Does your pet wear a collar?

26 responses




If yes, how long were they lost for?

11 responses



And, how did you find them?

11 responses

- He came home
- A person saw my post and called me
- We posted on a page on facebook and someone recognized him
- Apple AirTag
- They were chilling on the sidewalk
- neighbors found her
- instagram pages for lost pets
- neighbors found them
- Kept Driving around with my car until I found them

How would you try to find your pet if they ever run away from home?

26 responses

- By contacts
- From its microchip
- I would post fliers, text the neighborhood's animal group chats, and go looking for him if he, God forbid, ever gets lost.
- I would try to find them by hanging posters in my neighborhood
- Facebook groups
- Searching or by putting a microchip to track the dog
- Cry
- Walk in my neighborhood looking for it
- Search

How would you try to find your pet if they ever run away from home?

26 responses

post online and inform the neighbors

social media

just look for them

Samsung Tracker but at some point i guess somebody remove the tracker from the cat and throw it

Posters, calling neighbours

Social media sites, open sooq listings, asking around the neighborhood, hanging posters

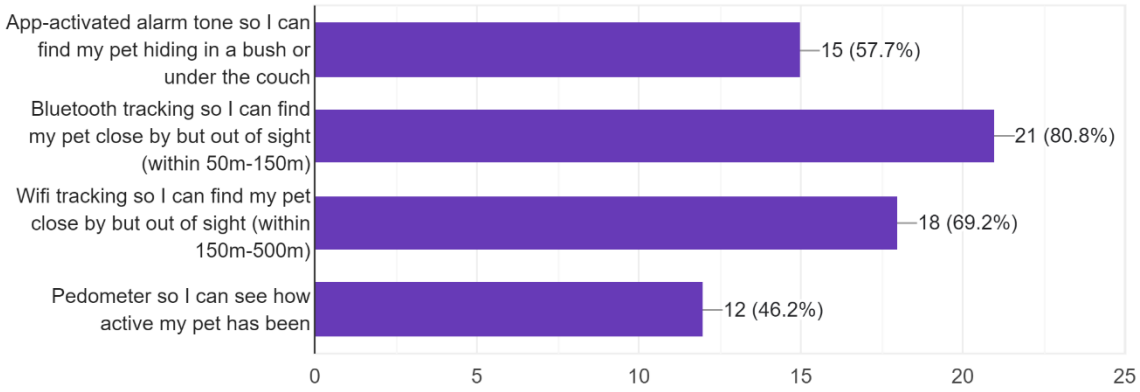
Share their pics on social media. Look for them myself. Keep their favorite food at the door in hopes they come back

look around the block

Search for them with my car

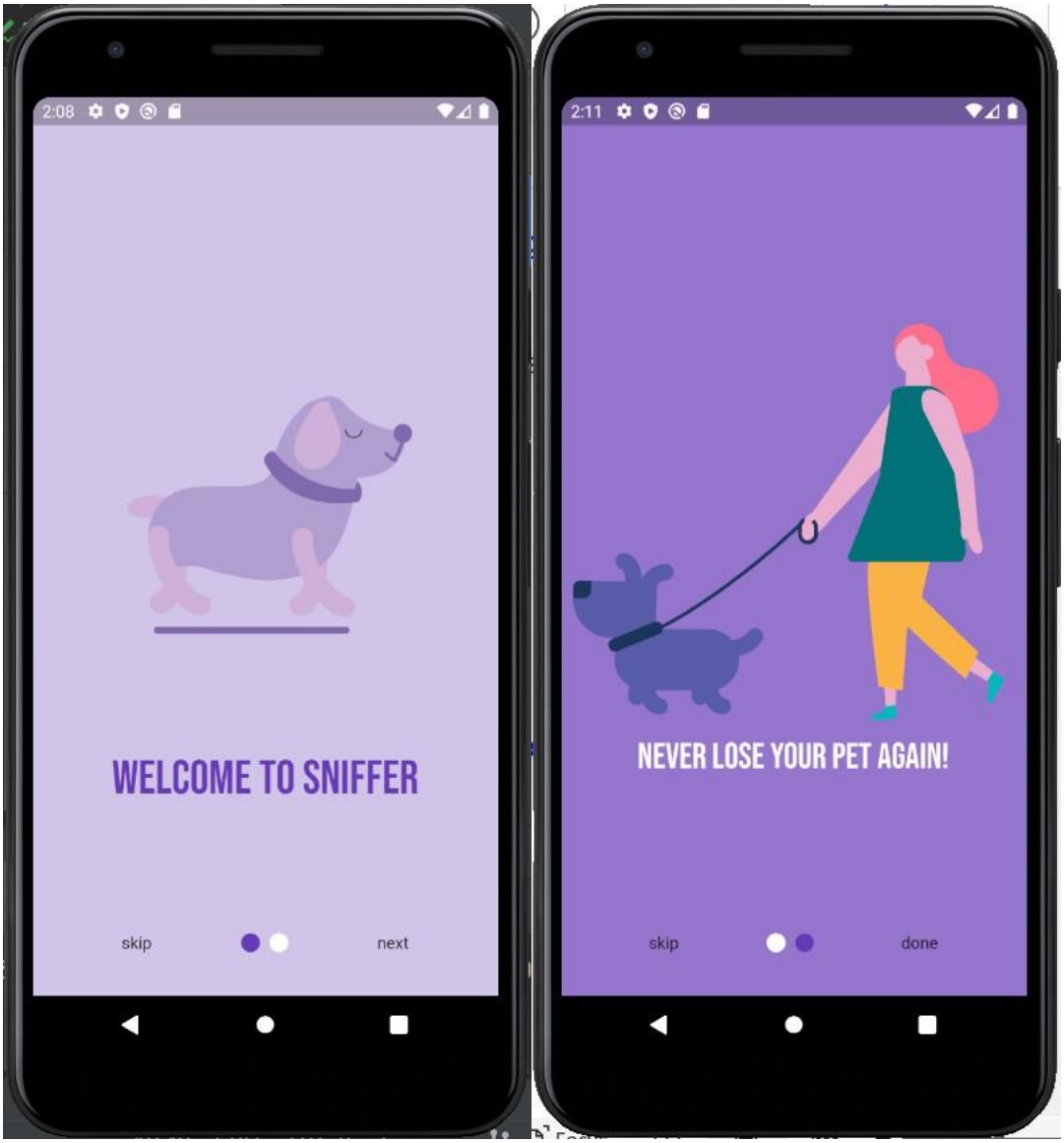
We are developing a pet tracker application which is connected to collar with a GPS device. It allows you to track your pet anywhere in the world...atures should we add that would be useful to you?

26 responses

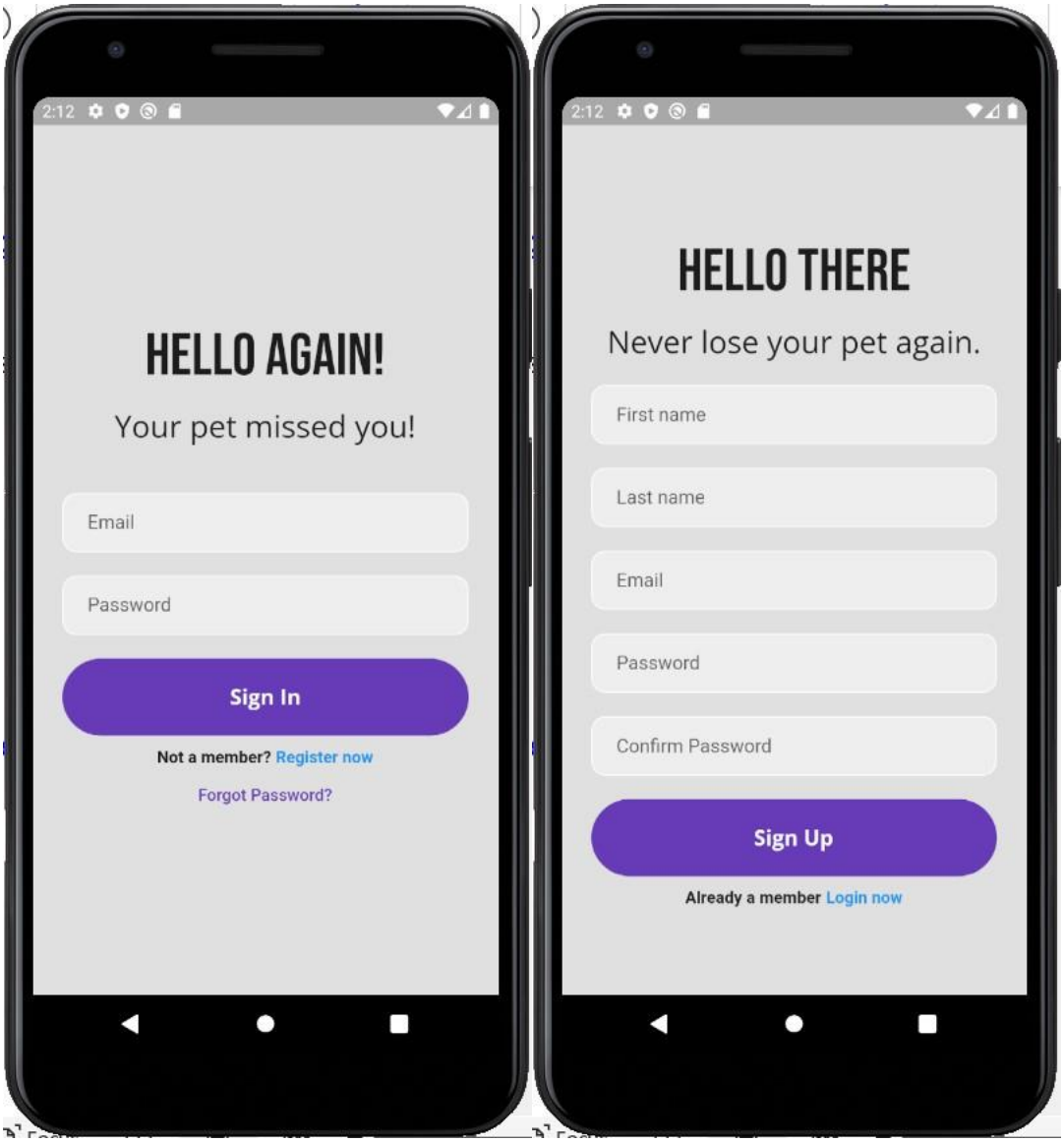


## Appendix D: User Manual

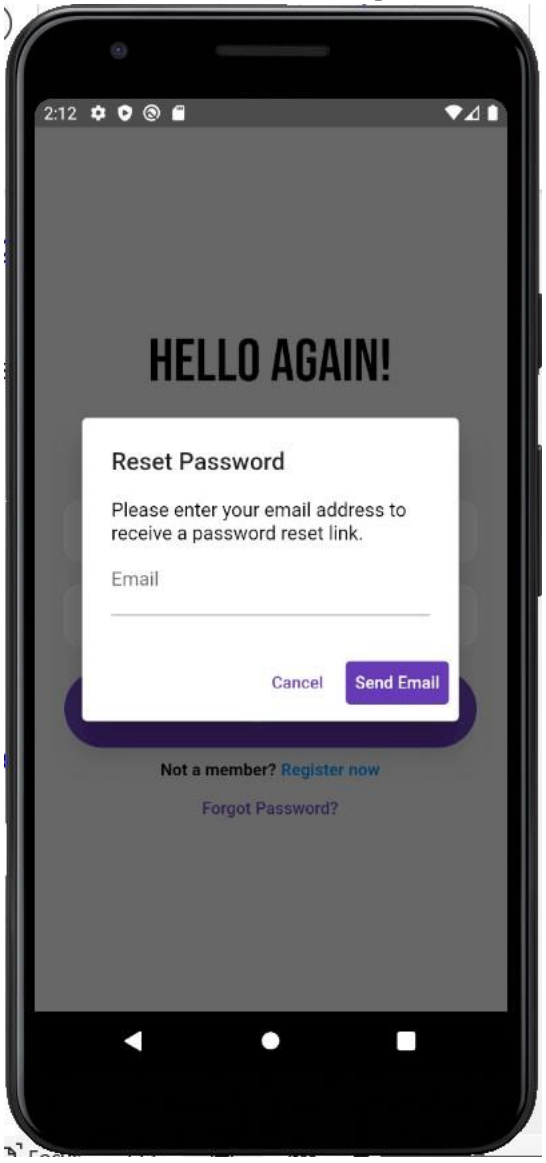
- When you open the application for the very first time, you are greeted with on-boarding screens



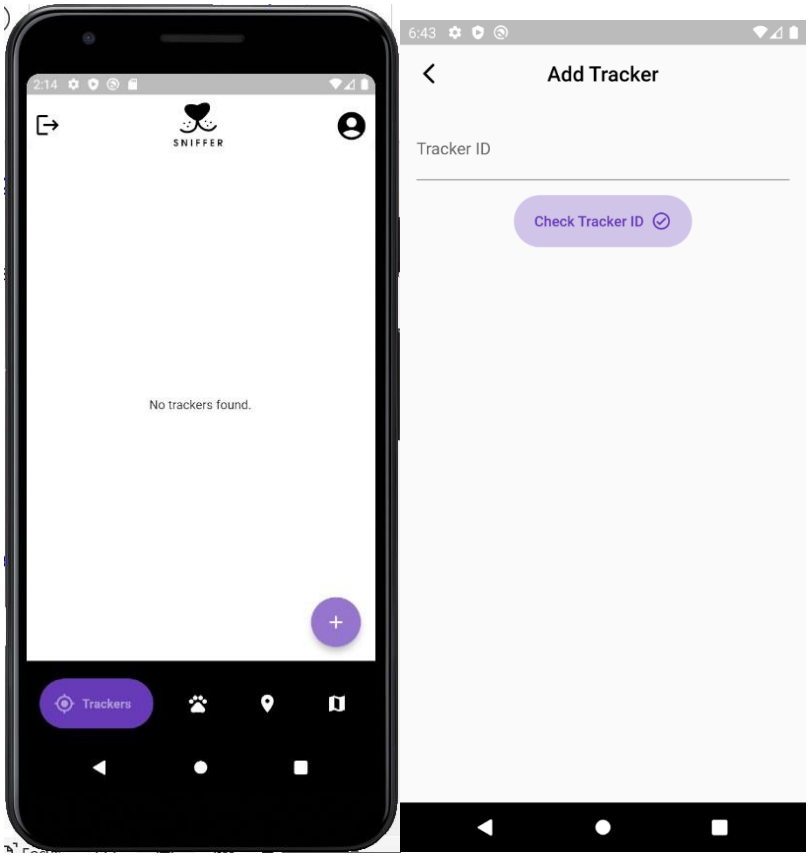

- These will then take you to the Login screen, where you can enter your credentials (email address and password) if you've previously created an account.
- If you are completely new to Sniffer, you can click on "Not a member? Register Now" to get started on creating a new account. You can create an account by entering:
    - First name
    - Last name
    - Email address
- Password (requirements: Must be between 8-15 characters and have at least one upper-case letter, special character, and number)

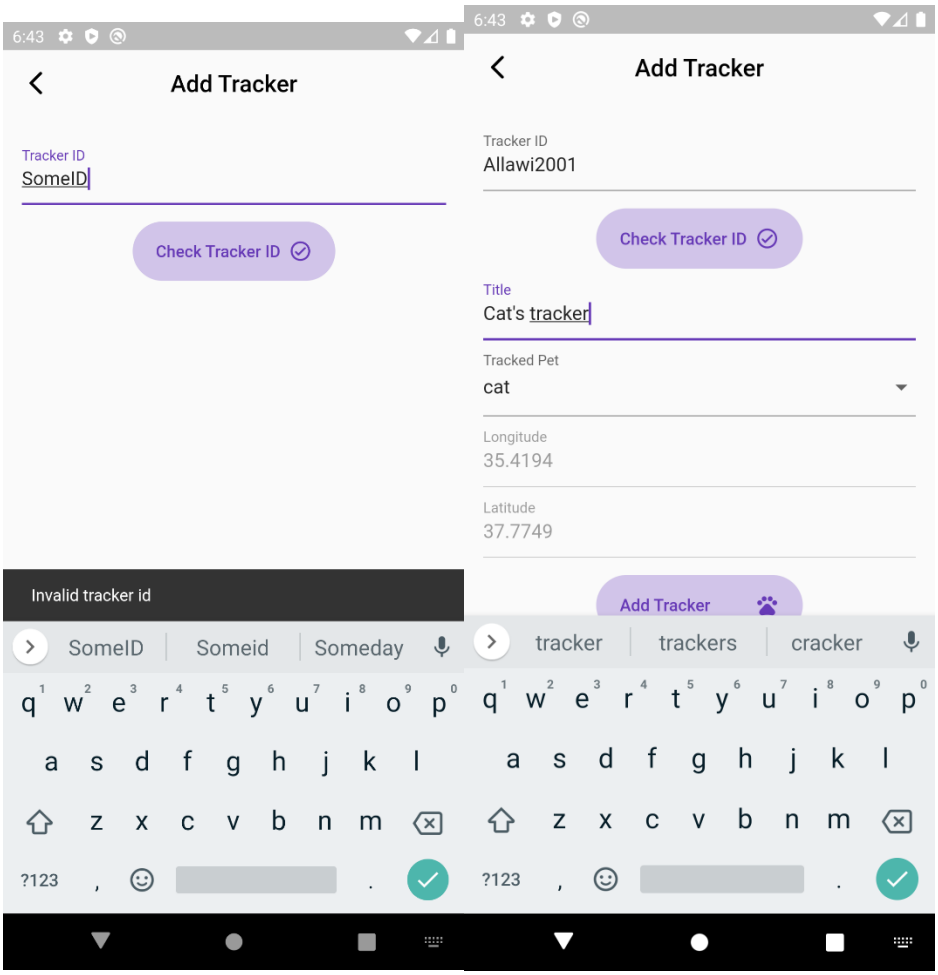

2:12
HELLO AGAIN!
Your pet missed you!
Email
Password
Sign In
Not a member? Register now
Forgot Password?
2:12
HELLO THERE
Never lose your pet again.
First name
Last name
Email
Password
Confirm Password
Sign Up
Already a member Login now

- If you've forgotten your password, "Forgot password?" will display a pop-up screen where you can enter your email address and receive a password reset link to your email.



- Once you are logged in, you can add a tracker by clicking on the (+) button
    - Validate your tracker by checking the tracker ID.
- Upon tracker verification, you can add a title for the tracker, and assign a pet if you've already created one.
    - Once you add the tracker it will appear on the trackers page
    - Clicking on the tracker will redirect you to its map page

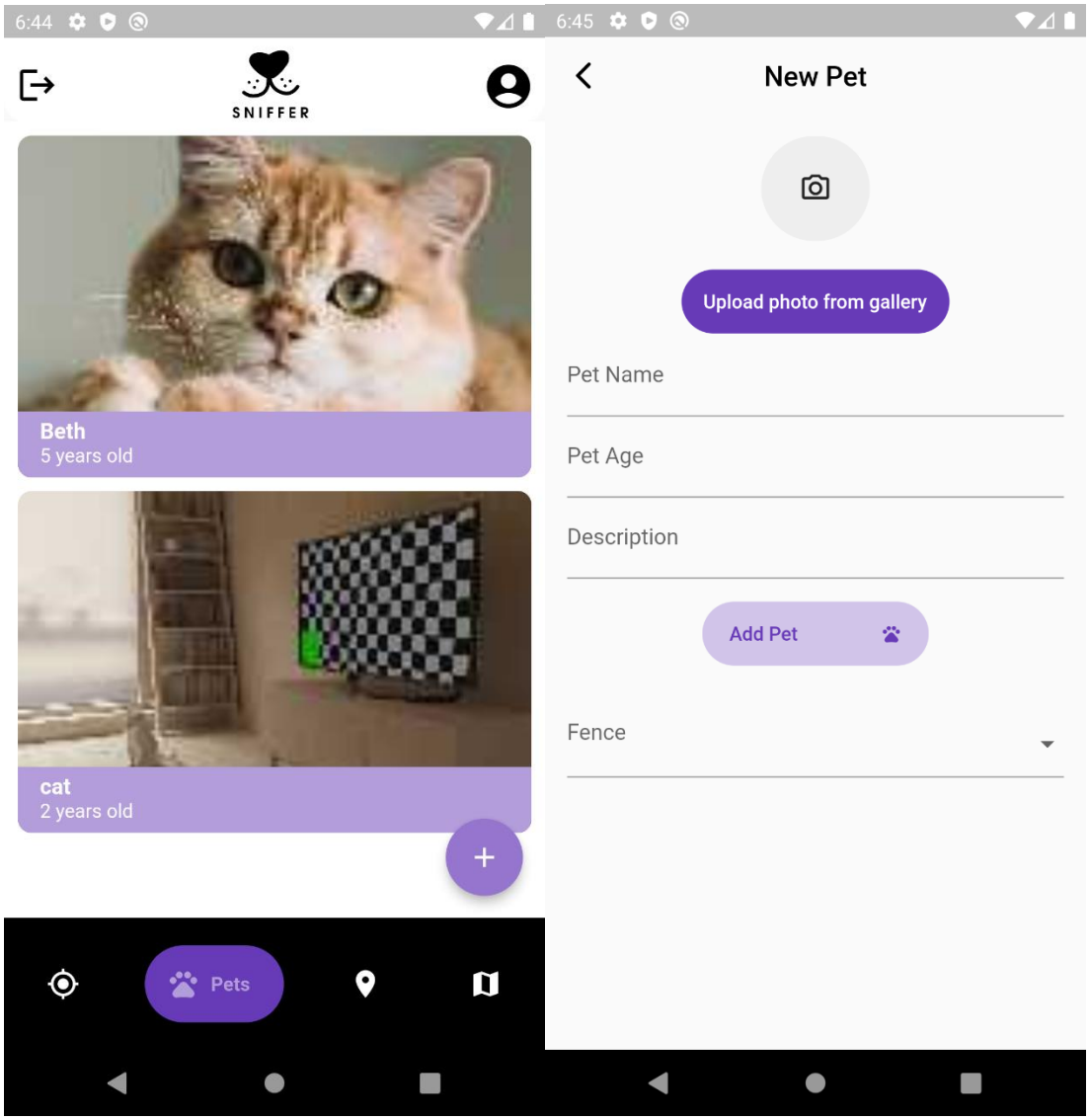


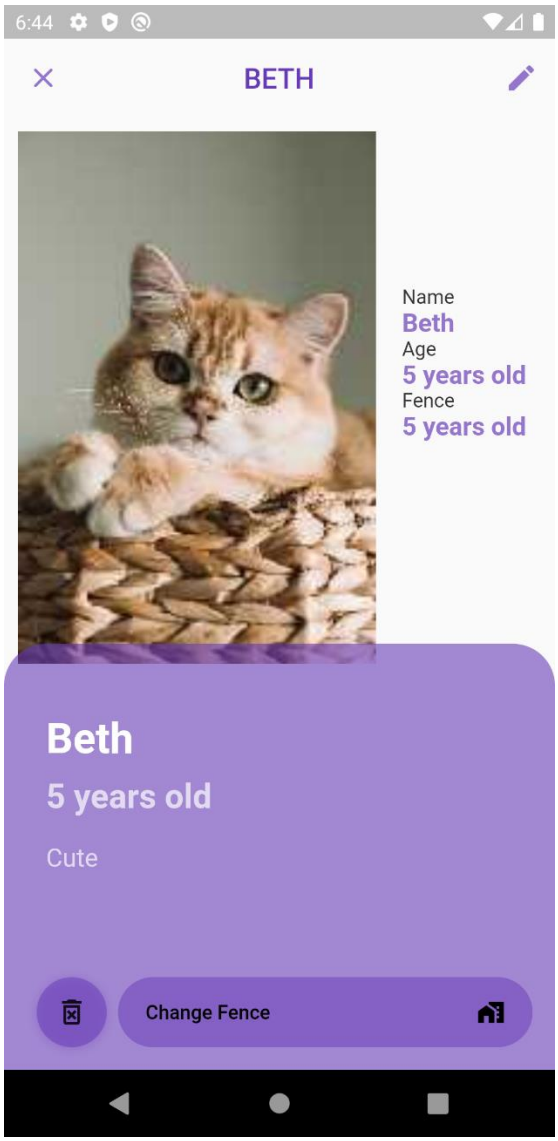

6:43
Add Tracker
Tracker ID
SomeID
Check Tracker ID
Invalid tracker id
SomeID
Someid
Someday
6:43
Add Tracker
Tracker ID
Allawi2001
Check Tracker ID
Title
Cat's tracker
Tracked Pet
cat
Longitude
35.4194
Latitude
37.7749
Add Tracker
tracker
trackers
cracker


6:42
SNIFFER
ALLAWI
Tracking Beth
Trackers

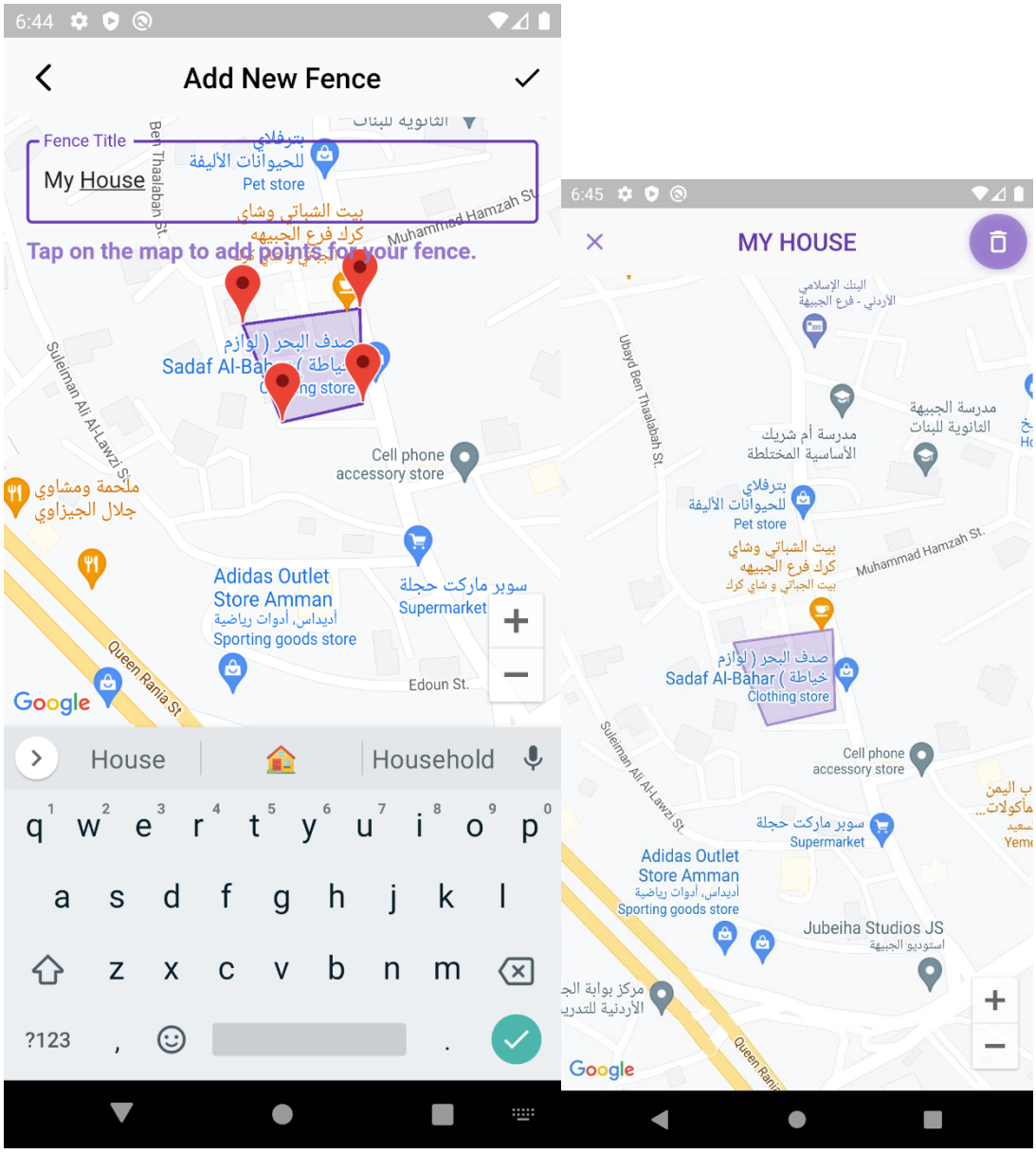

10:06
SNIFFER
مدرسة أم شريك الأساسية المختلطة
مدرسة الجبيهة الثانوية للبنات
بترفلاي للحيوانات الأليفة
Pet store
Ubayd Ben Thaalabah St.
Muhammad Hamzah St.
Sadaf Al-Bahar
Clothing store
Suleiman Ali Al Lawzi St.
Cell phone accessory store
سوبر ماركت حجلة
Supermarket
Google
Edoun St.
ALLAWI
Trackers

- The pet profile page allows you to add new pets or view your saved pets details.
- You can add a new pet by entering their name, age (number), description, and uploading their photo or taking a photo of them.

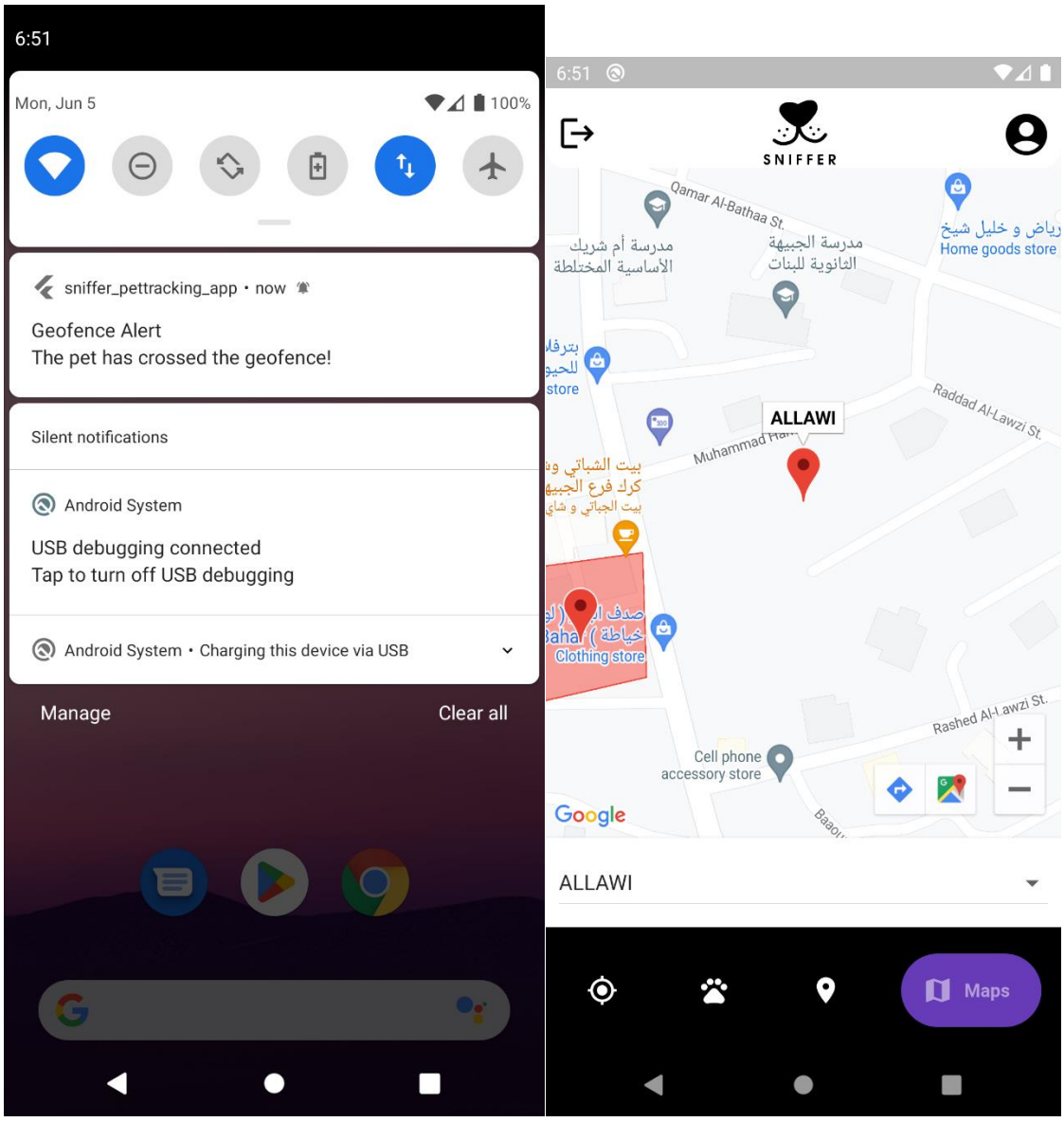



6:44
BETH
Name
Beth
Age
5 years old
Fence
5 years old
Beth
5 years old
Cute
Change Fence

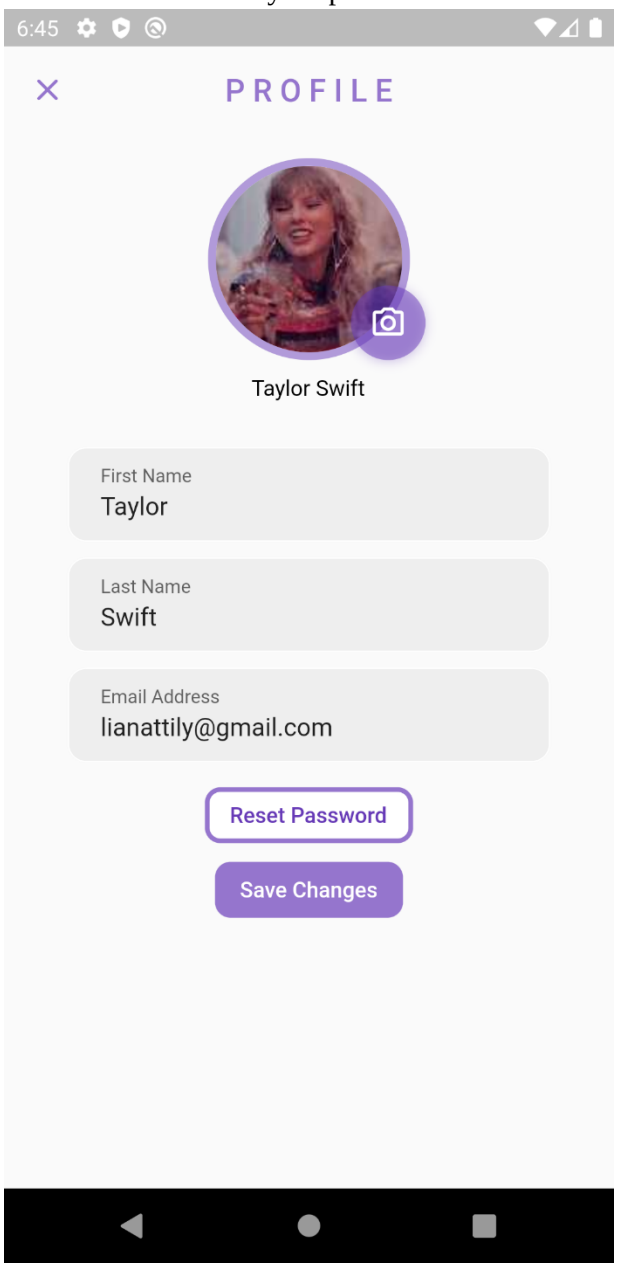
- The geofences page allows you to add geofences by tapping on the map to create markers and a border, then adding a title to the fence and then saving it .
    - You can also view or delete the added fence once you click on it.




- Once the pet crosses the geofence, the user will receive notifications like below, they can then check the current location of the tracker from the map page.


6:51
Mon, Jun 5
100%
sniffer_pettracking_app • now
Geofence Alert
The pet has crossed the geofence!
Silent notifications
Android System
USB debugging connected
Tap to turn off USB debugging
Android System • Charging this device via USB
Manage
Clear all
6:51
SNIFFER
Qamar Al-Bathaa St.
مدرسة أم شريك الأساسية المختلطة
مدرسة الجبيهة الثانوية للبنات
Home goods store
ALLAWI
Raddad Al-Lawzi St.
Clothing store
Rashed Al-l awzi St.
Cell phone accessory store
Google
ALLAWI
Maps

- You can edit your profile and adding a profile picture, or changing your display name or resetting your password.



# References


Basappa, P. (2021, June 16). US Missing Pet Epidemic and Euthanasia Statistics: Facts/Figures: Peeva. Retrieved November 4, 2022, from https://medium.com/nerd-for-tech/the-technology-behind-apples-airtag-c7983f9322b5

Hamilton, M. (n.d.). *US Missing Pet Epidemic and Euthanasia Statistics: Facts/Figures: Peeva*. Retrieved October 24, 2022, from https://peeva.co/blog/missing-pet-epidemic-facts-and-figures

Newman, L. (2022, December 21). Should You Put an AirTag on Your Dog's Collar? MUO. Retrieved November 4, 2022, from https://www.makeuseof.com/should-you-use-airtags-as-pet-trackers/

Reisen, J. (2021, August 14). How Do Pet Microchips Work and Should My Dog Have One? American Kennel Club. Retrieved November 16, 2022, from https://www.akc.org/expert-advice/lifestyle/how-do-dog-microchips-work/