

An Intelligent Animal Rescue System

BY

Eng Xian Yu

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

COPYRIGHT STATEMENT

© 2025 Eng Xian Yu. All rights reserved.

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

For all the assistance, encouragement, and support throughout this study, I would like to express my profound appreciation to my supervisor, Dr. Jasmina Khaw Yen Min. Her advice was valuable in the development of ResQMe, the Animal Rescue System mobile application.

I also appreciate the guidance from my family and friends for their unwavering support and encouragement. Lastly, I would like to thank my lecturers for their insightful counsel and advice during my academic career.

ABSTRACT

The ResQMe Animal Rescue System is a mobile application designed to streamline animal rescue operations by connecting the public, volunteers, and administrators on a single platform. Rescue operations sometimes face key challenges, including a lack of animal recognition, limited educational resources, and inadequate route planning systems. To overcome these limitations and make rescue operations more effective, ResQMe targets the implementation of a CNN-based animal recognition system, additional educational resources, and GPS-based real-time distance calculation and route planning to the nearest rescue clinics. Besides, the development of extra features, such as OCR-based IC verification, TTS pronunciation, built-in notifications, wildlife rescue reports, user extra description addition, and volunteer application submissions integration between Firebase and the Android Studio platform, was successfully implemented among admins and users. By combining core components, ResQMe meets the requirements in animal rescue operations and promotes public awareness in animal welfare.

Area of Study: Mobile Application Development, Computer Vision

Keywords: Animal Rescue, Mobile Application, Animal Recognition, OCR, Location Tracking, Educational Animal Resources

TABLE OF CONTENTS

TITLE PAGE	i
COPYRIGHT STATEMENT	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	x
LIST OF TABLES	xvi
LIST OF ABBREVIATIONS	xviii
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement and Motivation	1
1.2 Objectives	3
1.3 Project Scope and Direction	4
1.4 Contributions	4
1.5 Report Organization	5
CHAPTER 2 LITERATURE REVIEW	7
2.1 Previous works on Missing Pets	7
2.1.1 Missing Pets	7
2.2 Previous works on Pet Adoption	14
2.2.1 Pet Adoption	14
2.3 Previous works on Pawrpose	15
2.3.1 Pawrpose	19
2.4 Summary	
CHAPTER 3 SYSTEM METHODOLOGY/APPROACH (FOR DEVELOPMENT-BASED PROJECT)	21
3.1 System Design Diagram	21
3.1.1 Software Development Life Cycle (SDLC) phases	21

3.1.2	CRUD Operations	23
3.2	System Architecture	24
3.3	Animal Recognition Implementation	26
3.3.1	Data Collection and Preparation	26
3.3.2	Data Augmentation	27
3.3.3	Model Training Process	28
3.3.4	Training Performance Visualization	28
3.3.5	Model Performance	29
3.4	Use Case Diagram	30
3.4.1	Use Case Diagram Description	31
3.5	Activity Diagram	43
3.5.1	Log In / Sign Up Activity Diagram	43
3.5.2	Forgot Password Activity Diagram	44
3.5.3	Banner Activity Diagram	45
3.5.4	Categories Activity Diagram	46
3.5.5	News and Stories Activity Diagram	47
3.5.6	Camera Page Activity Diagram	48
3.5.7	Location Page Activity Diagram	49
3.5.8	Saved Animal Page Activity Diagram	50
CHAPTER 4	SYSTEM DESIGN	51
4.1	System Block Diagram	53
4.2	Storyboard	53
4.2.1	User Login and Registration Storyboard	53
4.2.2	Report Submission Storyboard	53
4.2.3	Admin Approval Storyboard	54
4.2.4	Camera Page Storyboard	54
4.2.5	Educational Section Storyboard	54
4.2.6	Description Submission Storyboard	55
4.2.7	Location Page Storyboard	55
4.3	Flowchart	56
4.4	System Components Specification	57
4.4.1	Registration and Authentication Module	57

4.4.2	Wildlife Reporting Module	57
4.4.3	Animal Recognition and AI Module	57
4.4.4	Volunteer Management Module	58
4.4.5	Location Service Module	58
4.4.6	Educational Content Module	59
4.4.7	News and Stories Module	59
4.5	Firebase Design	60
4.6	Prototype Design	71
4.7	Timeline	76
4.7.1	Timeline for FYP1	76
4.7.2	Timeline for FYP2	76

CHAPTER 5 SYSTEM IMPLEMENTATION (FOR DEVELOPMENT-BASED PROJECT)	77
5.1 Hardware Setup	77
5.2 Software Setup	78
5.3 Setting and Configuration	82
5.3.1 Android Studio Setup	82
5.3.2 Firebase Setup	82
5.3.3 TensorFlow Lite Setup	83
5.3.4 Third-Party APIs Setup	84
5.3.5 Algorithms Setup	96
5.3.6 Draw.io Setup	87
5.3.7 Figma Setup	87
5.4 System Operation	88
5.4.1 Splash Screen	88
5.4.2 Sign Up and Log in	89
5.4.2.1 User	89
5.4.2.2 Admin	91
5.4.3 Forgot Password	92
5.4.4 Home page	93
5.4.4.1 Drawer	93
5.4.4.2 Hotline Section	96
5.4.4.3 Report Section	97
5.4.4.4 Volunteer Section	101
5.4.4.5 Categories Section	105
5.4.4.6 Admin Approval Process	110
5.4.4.7 News and Stories	115
5.4.5 Camera Page	116
5.4.6 Location Page	119
5.4.7 Saved Animal Page	123
5.4.8 Admin Dashboard Page	125
5.4.9 ResQMe	126
5.5 Implementation Issues and Challenges	127
5.6 Concluding Remark	128

CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION	129
6.1 System Testing and Performance Metrics	129
6.1.1 User Login and Sign Up Module	131
6.1.2 Admin Login and Sign Up Module	131
6.1.3 Report Submission Module	132
6.1.4 Volunteer Application Module	133
6.1.5 Categories and Educational Resources Module	134
6.1.6 Admin Approval and Messages Module	136
6.1.7 Profile Setting Module	141
6.1.8 News and Stories Module	141
6.1.9 Camera and Animal Description Module	142
6.1.7 Location and Saved Animal Module	142
6.2 Testing Setup and Result	143
6.2.1 Authentication and Profile	143
6.2.2 Reports and Volunteer Module	144
6.2.3 Categories and Educational Resources Module	145
6.2.4 Admin, Messages, and Notifications	146
6.2.5 News and Stories Module	147
6.2.6 Camera Page Module	147
6.2.7 Location Page Module	148
6.2.8 Saved Animal Page Module	149
6.3 Project Challenges	150
6.4 Objectives Evaluation	151
6.5 Concluding Remark	152
CHAPTER 7 CONCLUSION AND RECOMMENDATION	153
7.1 Conclusion	153
7.2 Recommendation	154
REFERENCES	155
POSTER	158

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1	Main Page Missing Pets	7
Figure 2.2	Missing Page	8
Figure 2.3	Map for Lost Animal	8
Figure 2.4	Message Page	9
Figure 2.5	Notification Found Page	9
Figure 2.6	Found Page	10
Figure 2.7	Search Page	10
Figure 2.8	Search Page	10
Figure 2.9	Home Page for Pet Adoption	12
Figure 2.10	Animal Details	13
Figure 2.11	Animal Description	13
Figure 2.12	Filter feature	13
Figure 2.13	Filter Animal Types	13
Figure 2.14	Filter Time	13
Figure 2.15	Home Page Pawrpose	15
Figure 2.16	View All Animals Page	16
Figure 2.17	Search Page for Pawrpose	16
Figure 2.18	Community Page for Pawrpose	17
Figure 2.19	Blog Section	17
Figure 2.20	Blog Details Section	17
Figure 2.21	Profile Page for Pawrpose	18
Figure 3.1	Software Development Life Cycle (SDLC)	21
Figure 3.2	CRUD Operations	23
Figure 3.3	System Architecture	24
Figure 3.4	Graph Custom-Trained CNN Model Training Accuracy	29
Figure 3.5	Graph Custom-Trained CNN Model Training Loss	29
Figure 3.6	Use Case Diagram	30
Figure 3.7	Log In / Sign Up Activity Diagram	43

Figure 3.8	Forgot Password Activity Diagram	44
Figure 3.9	Banner Activity Diagram	45
Figure 3.10	Categories Activity Diagram	46
Figure 3.11	News and Stories Activity Diagram	47
Figure 3.12	Camera Page Activity Diagram	48
Figure 3.13	Location Page Activity Diagram	49
Figure 3.14	Saved Animal Page Activity Diagram	50
Figure 4.1	System Block Diagram	51
Figure 4.2a	Sign Up Storyboard	53
Figure 4.2b	Login Storyboard	53
Figure 4.3	Report Submission Storyboard	53
Figure 4.4	Admin Approval Storyboard	54
Figure 4.5	Camera Page Storyboard	54
Figure 4.6	Educational Section Storyboard	54
Figure 4.7	Description Submission Storyboard	55
Figure 4.8	Location Page Storyboard	55
Figure 4.9	Overall Flowchart	56
Figure 4.10	Firebase Design	60
Figure 4.11	Login and Sign-Up Prototype	71
Figure 4.12	Home Page Prototype	71
Figure 4.13	Hotline Prototype	71
Figure 4.14	Report Prototype	72
Figure 4.15	Volunteer Prototype	72
Figure 4.16	Categories Prototype	72
Figure 4.17	Animal Details Prototype	72
Figure 4.18	Add Description Prototype	73
Figure 4.19	News and Stories Prototype	73
Figure 4.20	News Article Prototype	73
Figure 4.21	Camera Page Prototype	73
Figure 4.22	Animal Description Prototype	74
Figure 4.23	Location Page Prototype	74
Figure 4.24	Clinic Detail Prototype	74
Figure 4.25	Saved Animal Page Prototype	75

Figure 4.26	Saved Animal Description Prototype	75
Figure 4.27a	Gantt Chart for FYP1	76
Figure 4.27b	Gantt Chart for FYP2	76
Figure 5.1a	Kaggle	78
Figure 5.1b	Android Studio	78
Figure 5.1c	Firebase	78
Figure 5.2a	Jupyter Notebook	79
Figure 5.2b	Java	79
Figure 5.2c	TensonFlow	79
Figure 5.2d	Keras	79
Figure 5.2e	OpenCV	80
Figure 5.2f	NumPy	80
Figure 5.3a	Draw.io	81
Figure 5.3b	Figma	81
Figure 5.3c	Canva	82
Figure 5.4a	Android Studio Setup	82
Figure 5.4b	Setup Android SDK	82
Figure 5.5a	Sign in Firebase	83
Figure 5.5b	Download google-services.json	83
Figure 5.5c	Enable components in build	83
Figure 5.6	Integrate JSON files and TFLite files	84
Figure 5.7	Splash Animation	88
Figure 5.8a	User Login	89
Figure 5.8b	User Sign Up	90
Figure 5.8c	User Sign Up Information	90
Figure 5.8d	Invalid IC photo	90
Figure 5.8e	Email Address Verification	91
Figure 5.9	Admin Login and Sign Up	91
Figure 5.10a	Change Password	92
Figure 5.10b	Reset Password	92
Figure 5.11a	Home Page Part 1	93
Figure 5.11b	Home Page Part 2	93
Figure 5.12a	Drawer User Type	94

Figure 5.12b	Update Profile and Username	94
Figure 5.12c	Empty Report	95
Figure 5.12d	Empty Messages	95
Figure 5.12e	Empty Application	95
Figure 5.12f	Empty Description	95
Figure 5.12g	Empty Volunteer Management	96
Figure 5.13a	Hotline Call and Email	96
Figure 5.13b	Rule-Based Chatbot in Hotline	87
Figure 5.14a	Auto-Filled Reporter Information	98
Figure 5.14b	Dropdown for Wildlife Type and Urgency Level	98
Figure 5.14c	Location Information and Photo Evidence	99
Figure 5.14d	Notifications for User	99
Figure 5.14e	Notifications from Both Drawer and Admin Dashboard	100
Figure 5.14f	Notification and Submission Approval	100
Figure 5.15a	Admin Create Volunteer Form	101
Figure 5.15b	Volunteer Form	102
Figure 5.15c	Button Volunteer Now	102
Figure 5.15d	Volunteer Opportunities and Details	103
Figure 5.15e	Application Submitted Successfully	103
Figure 5.15f	Notification for User	104
Figure 5.15g	Information Shown	104
Figure 5.15h	Admin Update the Form	104
Figure 5.16a	Categories Section	105
Figure 5.16b	Animal Categories Detail Page	105
Figure 5.16c	Search Functionality for Categories Section	106
Figure 5.16d	Animal Detail Page	106
Figure 5.16e	AI Chatbot	107
Figure 5.16f	Add Description Button	107
Figure 5.16g	Add Description Page	108
Figure 5.16h	Description Submitted Successfully	108
Figure 5.16i	Notification for Both Admin and User	109
Figure 5.16j	Description Waiting for Approval	109
Figure 5.17a	Admin Approved Wildlife Report	110

Figure 5.17b	Notification for User and Status Updated	111
Figure 5.17c	Application Approved and Status Updated	111
Figure 5.17d	Notification for User	112
Figure 5.17e	Description Rejected and Reason Given	112
Figure 5.17f	Notification for User	113
Figure 5.17g	Description Approve Confirmation	113
Figure 5.17h	Description Status Changed	114
Figure 5.17i	Description Updated for All Users	114
Figure 5.18a	View More Button in News and Stories	115
Figure 5.18b	News Loading and Article Shows	115
Figure 5.18c	Read Full Article and Navigate to Website	116
Figure 5.19a	Capture Image in Camera View	117
Figure 5.19b	Choose Image from Gallery	117
Figure 5.19c	Animal Description Page	118
Figure 5.19d	Save Animal	118
Figure 5.19e	Learn More Button to A-Z Animal Website	119
Figure 5.20a	Location Page	120
Figure 5.20b	Filter Search Radius	120
Figure 5.20c	Input Location Manually	121
Figure 5.20d	Location Searching via Input	121
Figure 5.20e	Shows All Clinics Nearby	122
Figure 5.20f	Search Button with Clinic's Name	122
Figure 5.20g	Navigate to Navigational Tool	123
Figure 5.21a	Saved Animal Page	124
Figure 5.21b	Deletion of Animal List	124
Figure 5.21c	Animal Detail Page in Saved Animal Page	125
Figure 5.22	Admin Dashboard Page	126
Figure 5.23	ResQMe Logo	126
Figure 6.1a	Error Messages for Blank Information in User Login	129
Figure 6.1b	Error Messages for User Sign Up Page	130
Figure 6.1c	Invalid Email Address and Password	130
Figure 6.1d	Error Message for IC Photo	131
Figure 6.2	Testing for Admin Login and Sign Up Page	131

Figure 6.3a	Blank Wildlife Type and Dropdowns Testing	132
Figure 6.3b	Take Photo and Select Photo Buttons Testing	132
Figure 6.3c	Submit Report Button Testing	132
Figure 6.4a	Volunteer Application Module Testing	133
Figure 6.4b	Duplicate Application Testing	133
Figure 6.5a	Categories Testing	134
Figure 6.5b	Search Functionality Educational Resources Testing	135
Figure 6.5c	Gemini AI Testing	135
Figure 6.5d	Add Description Testing	136
Figure 6.6a	Notification from Admin Dashboard and Drawer Testing	137
Figure 6.6b	Messages from Submissions	137
Figure 6.6c	Admin Approve Report Testing	137
Figure 6.6d	Admin Approve Description Testing	138
Figure 6.6e	Admin Approve Volunteer Application Testing	138
Figure 6.7a	Admin Reject Report Testing	139
Figure 6.7b	Admin Reject Description Testing	139
Figure 6.7c	Admin Reject Volunteer Application Testing	140
Figure 6.8a	Notifications and Messages Testing	140
Figure 6.8b	Submissions Status Update Testing	140
Figure 6.9a	Profile Changing Testing	141
Figure 6.9b	Profile Update Testing	141
Figure 6.10	News and Stories Testing	142
Figure 6.11	Remove Animal Testing	142

LIST OF TABLES

Table Number	Title	Page
Table 2.1	Strengths and Weaknesses of Missing Pets	11
Table 2.2	Strengths and Weaknesses of Pet Adoption	14
Table 2.3	Strengths and Weaknesses of Pawrpose	19
Table 2.4	Strengths and Weaknesses of Existing Animal Rescue Applications	19
Table 2.5	ResQMe Proposed Solutions	20
Table 3.1	CRUD Operations	23
Table 3.2	Login Use Case	31
Table 3.3	Sign Up Use Case	32
Table 3.4	Update Profile Use Case	33
Table 3.5	Log Out Use Case	34
Table 3.6	View Home Page Use Case	34
Table 3.7	View Categories Section Use Case	35
Table 3.8	View Hotline Session Use Case	36
Table 3.9	View News and Stories Section Use Case	37
Table 3.10	View Volunteer Section Use Case	37
Table 3.11	Receive Message Use Case	38
Table 3.12	View Camera Page Use Case	39
Table 3.13	View Location Page Use Case	39
Table 3.14	View Saved Animal Page Use Case	40
Table 3.15	Fill in Volunteer Form Use Case	41
Table 3.16	View Admin Dashboard Use Case	42
Table 5.1	Specifications of Laptop	77
Table 5.2	Specifications of Smartphone	77
Table 5.3	Main Software Tools	78
Table 5.4	Programming Language	79
Table 5.5	Libraries and Packages	79
Table 5.6	Java Packages	80

Table 5.7	Extra Tools from Network	81
Table 5.8	Third-Party APIs Setup	84
Table 5.9	Algorithms Setup	96
Table 6.1	Authentication and Profile	143
Table 6.2	Reports and Volunteer Module	145
Table 6.3	Categories and Educational Resources Module	145
Table 6.4	Admin, Messages, and Notifications	146
Table 6.5	News and Stories Module	147

LIST OF ABBREVIATIONS

<i>Admin</i>	Administrator
<i>AI</i>	Artificial Intelligence
<i>API</i>	Application Programming Interface
<i>CPU</i>	Central Processing Unit
<i>CRUD</i>	Create, Read, Update, Delete
<i>CNN</i>	Convolutional Neural Network
<i>GPS</i>	Global Positioning System
<i>GPU</i>	Graphics Processing Unit
<i>IC</i>	Identity Card
<i>LLM</i>	Large Language Model
<i>ML</i>	Machine Learning
<i>NLP</i>	Natural Language Processing
<i>OCR</i>	Optical Character Recognition
<i>RAD</i>	Rapid Application Development
<i>RAM</i>	Random Access Memory
<i>SDLC</i>	Software Development Life Cycle
<i>TFLite</i>	TensorFlow Lite
<i>TTS</i>	Text-to-Speech

Chapter 1

Introduction

This project entails developing an **Animal Rescue System** called **ResQMe** to assist the public in identifying animals, locating useful educational resources, and tracking them to the nearest clinics in case of an emergency. Features including a home page, camera-based animal recognition, location-based support, and a saved animal list are all included in the Android Studio-built app. The main features are a camera system that uses Convolutional Neural Networks (CNN) [1] to identify animals in images directly captured or uploaded from the gallery. Haversine Formula for GPS distance is also applied in this system for finding the nearest clinics [2]. After an animal has been discovered, the app gives users comprehensive details about it and lets them keep their preferred discoveries or look for further information from outside sources.

This project's goals include implementing animal identification capability. Two CNN models were combined, which are a pre-trained MobileNetV2 model [3], and a CNN model trained on a custom dataset. More recognition coverage is guaranteed by this dual-model method, particularly for animals that are absent from typical datasets. Besides, navigational algorithms like the Haversine Formula [2] for GPS distance are important to make calculations to find the nearest route. Educational resources are provided for enhancing knowledge of various animals. The app also has a bottom bar that allows users to navigate to important sites like Home, Camera, Location, and Saved Animals.

1.1 Problem Statement and Motivation

1.1.1 Problem Statement

1. Lack of Animal Recognition Feature in Existing Animal Rescue System

Many animal rescue applications, such as “Missing Pets [4]”, “Pets Adoption”, and “Pawrpose [5]”, primarily focus on raising awareness, sharing blogs, and encouraging donations, but lack advanced technologies like Convolutional Neural Networks (CNN) for animal identification and recognition [6]. This absence represents a missed opportunity to enhance rescue operations and efficiency, as current identification methods rely on slow and error-prone manual processes that hinder quick emergency responses. CNNs enable real-time image processing, which is

essential for rapid reactions during crises, and facilitate scalable rescue operations by quickly analyzing images from various sources. Additionally, by incorporating animal recognition features, rescue teams can better prepare for each specific case, such as bringing appropriate medications, allergy treatments, or species-specific supplies. This will ultimately reduce response time and improve rescue efficiency.

2. Lack of educational resources in existing animal rescue mobile applications

While “Pet Adoption” mainly targets animal rehoming, other animal rescue apps like Missing Pets [4] and Pawrpose [5] emphasize helping pet owners locate their lost animals. However, these applications tend to prioritize pet recovery and adoption over user education. Because of this, there is a lack of educational materials available to users that could explain various animal species and how to deal with wildlife or injured animals. This disparity emphasizes the necessity for a mobile application that not only aids in animal rescue efforts but also offers useful educational content to increase public awareness and knowledge.

3. Limited route planning for animal rescue centres

Current apps for animal rescue, such as "Pet Adoption," offer route planning capabilities primarily to direct users to pet adoption facilities. The mobile applications like "Missing Pets [4]" and "Pawrpose [5]" do not provide routes to any rescue facilities; instead, their main goal is to assist users in finding the last known location of missing pets.

These existing mobile applications do not provide the necessary route planning to link emergency animal cases to the proper rescue facilities. With this limitation, the user needs to do additional work for searching and browsing clinics via the Internet, which will delay the rescue operations, especially when the animal is in dangerous conditions.

1.1.2 Motivation

This project is motivated by limitations of features such as a lack of animal recognition, educational resources, and route planning to animal rescue clinics in several existing mobile applications. These gaps cause the slow response time for rescue operations and reduce public awareness. Therefore, by integrating the CNN animal recognition, additional animal educational resources, and route planning by distance calculation to reach the nearest clinics, this project will aim to enhance the public’s knowledge and enhance rescue efficiency.

1.2 Objectives

ResQMe is an innovative animal rescue system that significantly improves the efficiency and accuracy of rescue operations. The system aims to:

1. To Integrate CNN into the Animal Rescue System

A Convolutional Neural Network (CNN) model [1] will be included in the animal rescue mobile application to enable automated animal identification. Given their stellar reputation for image classification, CNNs are perfect for identifying various animal species in user-submitted photos.

By letting users snap fresh images or upload existing ones, the system can quickly and accurately identify the species. This expedites rescue response times while also assisting the public in identifying animals. Rescue teams can save time and improve the likelihood of a successful rescue by preparing the necessary medical supplies and handling skills ahead of time, with accurate identification.

2. To Provide Educational Resources Related to Animal Knowledge

This project will have an educational portion that classifies animals like birds, reptiles, and mammals to improve user experience and encourage appropriate animal interaction. Information on each species, such as habitat, will be provided in each category. This feature is to increase public knowledge of animal welfare. The project's goal of animal safety and education will be strengthened by providing easily accessible information that will better equip users to handle real-life interactions and encourage thoughtful, caring reactions.

3. To Implement Route Planning to Find the Shortest Route for Pet Clinics

To assist users in finding the closest animal and pet rescue centers, the app will incorporate a navigation function, such as the Haversine Formula [2] for GPS distance. This will be especially helpful in emergencies, filling a void in existing rescue applications by directing users to the proper facility for injured animals.

The system integrates location services to calculate the shortest route from the user's current location to the destination clinic, enabling the display of a map to visually point out the distance.

1.3 Project Scope and Direction

The project aims to develop a mobile application called ResQMe for animal rescue on the Android platform that incorporates teaching materials, real-time animal recognition, and a navigation system to rescue facilities. Besides, other additional features, such as integration between administrators and users, are also implemented for better user experience. Firebase Authentication and Firestore for both user and administrator authentication, reporting cases, and volunteer involvement.

The **Educational resources** were grabbed from six APIs, and Google LLM [7] provided such as Gemini AI, for better understanding and gaining knowledge. The **CNN-based animal identification system** and camera page's essential features were implemented into this project. Users can either take a picture of an animal or upload one, and the system will identify the species and provide relevant details. Users can also store animals in a personal list in the application for later use. **Location-based systems and navigational tools**, such as the Haversine Formula for GPS distance [2], for time estimating and distance calculations, especially when facing an emergency case.

This project aims to provide a comprehensive, user-friendly platform that supports domestic and wildlife rescue operations, raises public awareness, and aids in animal rescue initiatives.

1.4 Contributions

ResQMe, the animal rescue mobile application, showcases AI and cloud integration for both educational resource providing and location mapping. Architectures such as Firebase [8], Android Jetpack [9], and ML toolkits [10] are offered for improving public awareness of urgent cases and the completeness of the entire project.

The viability of incorporating deep learning technology into mobile animal rescue apps is confirmed by the deployment and testing of the CNN-based animal recognition system. First, even when trained on both bespoke and pre-trained datasets, the experiment shows that

CHAPTER 1

Convolutional Neural Networks (CNNs) can successfully recognize a variety of animals using real-time visual analysis. Second, the system addresses a significant gap in existing animal rescue applications by emphasizing the value of offering educational information in addition to rescue operations. Thirdly, by planning the integration of route planning elements that target both pet and wildlife rescue facilities, the project broadens the reach beyond conventional pet-focused rescue apps and lays the groundwork for creating a more comprehensive rescue solution.

1.5 Report Organization

This report includes 7 chapters, detailing the complete process for developing an intelligent animal rescue system in a mobile application.

Chapter 1 outlines the problem statements related to existing mobile application platforms, defines three main objectives of the project, and establishes the scope to be followed throughout its development.

Chapter 2 introduces the relevant research on Pet Adoption, Missing Pets, and comparable mobile applications are reviewed, and their advantages and disadvantages are examined. A summary table concluded all the strengths and weaknesses of the existing animal rescue platform and their improvements.

Chapter 3 explains the overall methodology and working flow, development process, model design, and system architectures and models used in ResQMe. This chapter defines the process from the initial to the final stage, integrated with Android toolkits and Firebase Cloud services, following the use case diagram and activity diagrams to explain the overall process in ResQMe.

Chapter 4 mentions the system design flow, such as storyboards, prototypes, and component specifications. This chapter lets the user have a better understanding of the design flow evaluation.

Chapter 5 concludes the overall implementation process, such as hardware and software setup, and challenges during the implementation process.

CHAPTER 1

Chapter 6 evaluates the system's performance, testing process, expected and real output, and challenges in the testing process.

Chapter 7 concludes the whole project with key findings and future enhancements.

Chapter 2

Literature Review

This chapter reviews several existing animal rescue mobile applications and their core features in rescue operations. By analyzing platforms such as Missing Pets [4], Pet Adoption, and Pawrpose [5], this chapter identifies their strengths and weaknesses, comparing and summarizing all existing animal rescue mobile applications, and discovers the lack of some features. ResQMe then addresses the current limitations and gives a better solution to enhance rescue operations.

2.1 Previous works on Missing Pets

2.1.1 Missing Pets [4]

Missing Pets is a mobile application focused on lost and found animals. It helps locate missing dogs and cats by allowing users to report lost pets or assist in finding them. The app encourages community involvement in pet rescue efforts.

1. Clean and easy-to-understand main page

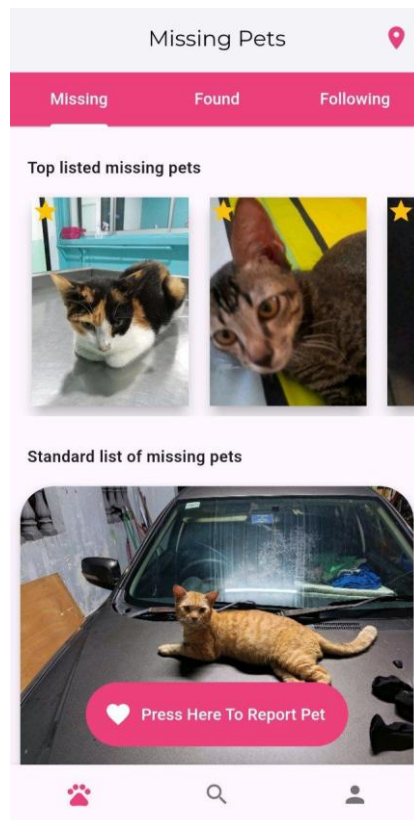


Figure 2.1 Main Page Missing Pets

CHAPTER 2

2. Missing page for details

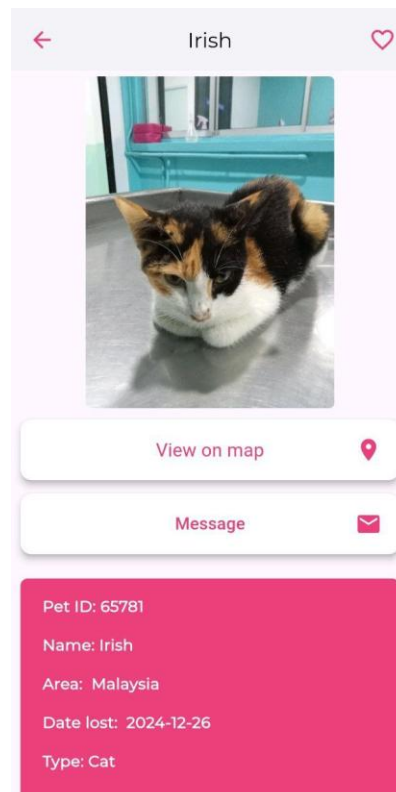


Figure 2.2 Missing Page

3. Map page

Missing Pets includes a feature that displays the last known location of a lost animal.

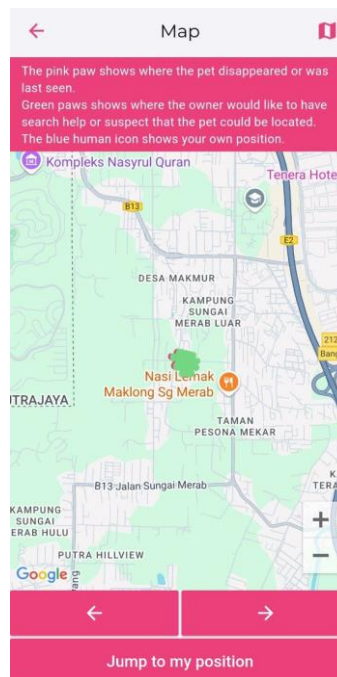


Figure 2.3 Map for Lost Animal

4. Message owner page

Once a missing pet is found, users can choose to contact the owner directly.



Figure 2.4 Message Page

5. Found Animal Page with the listing of pets

This page displays animals that have been found but whose owners are still unknown.

Users can browse through these animals to check if any of them belong to them.

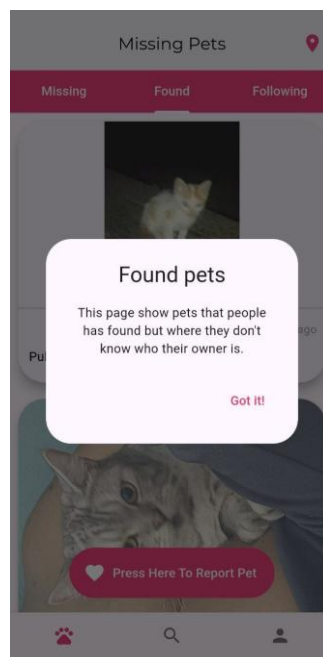


Figure 2.5 Notification Found Page

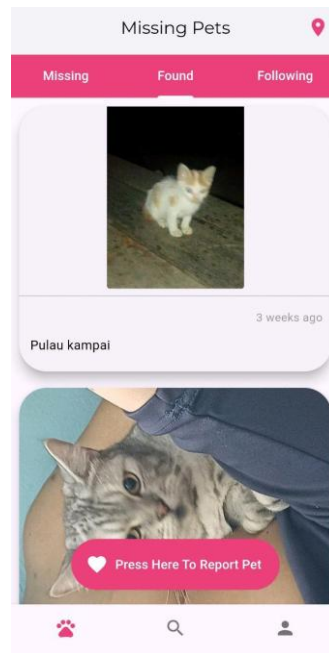


Figure 2.6 Found Page

6. Search page

Users can search for pets using a unique pet ID and follow them to receive updates if new information becomes available.

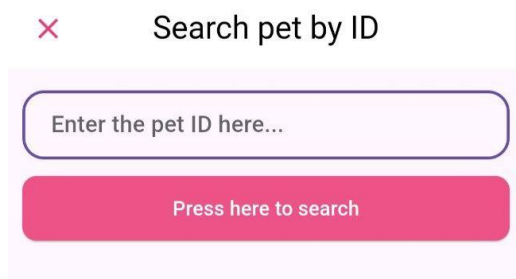


Figure 2.7 Search Page

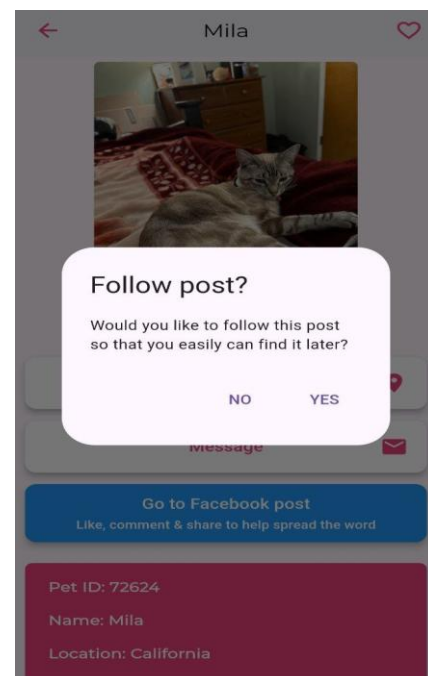


Figure 2.8 Search Page2.1.2

Strengths and Weaknesses of Missing Pets

Table 2.1 Strengths and Weaknesses of Missing Pets

Missing Pets	
Strengths	Weaknesses
Provides a platform for users to report missing pets and search for them.	Focuses only on missing pets, not on rescuing emergency or injured animals.
Displays missing animals on a map showing the last known location.	The map feature only shows lost pet locations, not nearby rescue centers.
Allows messaging between users and pet owners for better communication.	Lacks educational resources to inform users about different animal types or emergency handling.
Includes separate "Found" and "Following" pages for better organization.	Lack of animal recognition features, limited enhancement of animal knowledge.

2.2 Previous works on Pet Adoption

2.2.1 Pet Adoption

1. Home Page

Pet Adoption allows users to browse nearby pets, keeping them informed about available animals and enabling them to contact the owners.

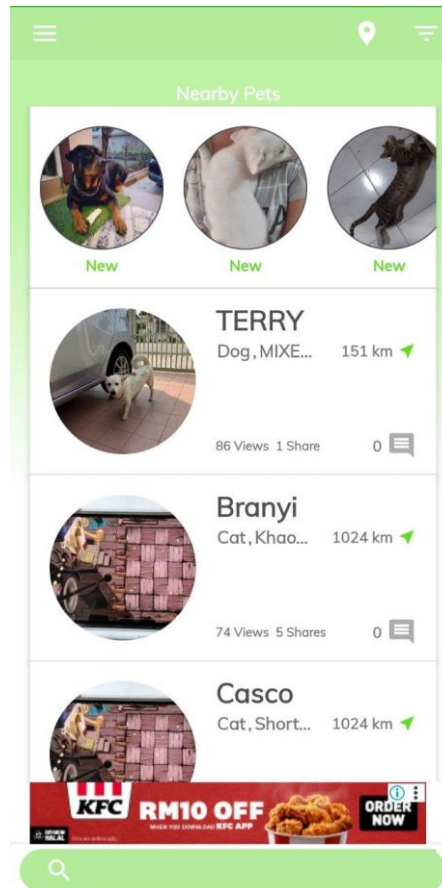


Figure 2.9 Home Page for Pet Adoption

2. Animal Details Page

This page provides detailed information about pets' adoption. Users can view the animal's species, sex, a short description ("About Me"), and the location of the animal. The pet owner also includes their name and contact number, allowing users to either call or send a message if they are interested.

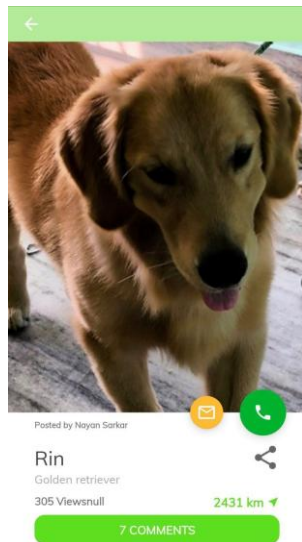


Figure 2.10 Animal Details

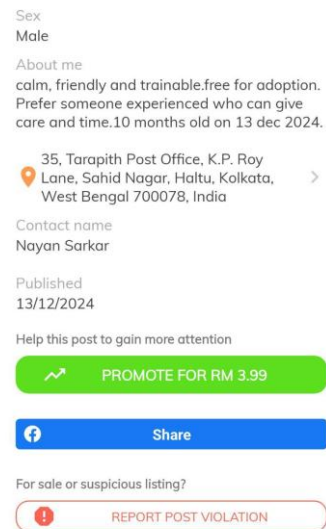


Figure 2.11 Animal Description

3. Filter feature

Users can filter animals by type, such as dogs, cats, birds, fish, and others, as well as by date, ranging from the last 30 days to the past 6 months.

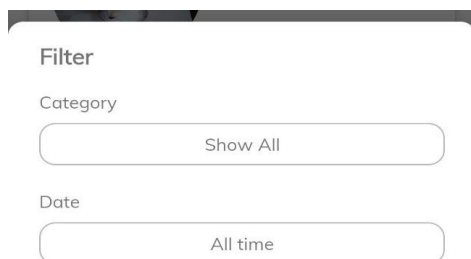


Figure 2.12 Filter feature

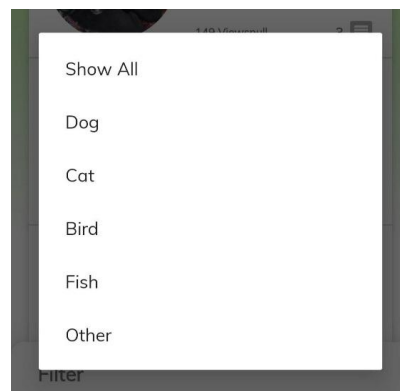


Figure 2.13 Filter Animal Types

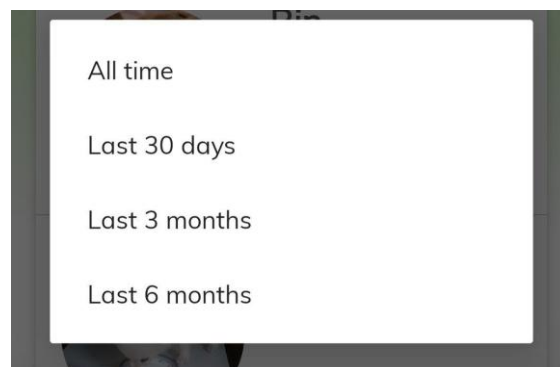


Figure 2.14 Filter Time

Strengths and Weaknesses of Pet Adoption

Table 2.2 Strengths and Weaknesses of Pet Adoption

Pet Adoption	
Strengths	Weaknesses
Offers a detailed list of pets available for adoption, including their name, sex, and other basic information.	Focuses mainly on pet adoption, without features for rescuing injured or emergency animals.
Enables user interactions through comments on pet profiles.	Does not provide a map showing nearby rescue centers, only the location of pets available for adoption.
	No educational content or resources to increase user knowledge about animal rescue or welfare.

2.3 Previous works on Pawrpose

2.3.1 Pawrpose [5]

1. Clearly Defined Home Page

Pawrpose is a platform designed for animal rescue alerts and pet adoption. The homepage displays lost animals categorized by priority levels, which are high, medium, and low, to help users identify which animals need urgent care. It also features listings for both lost and found pets, making it easier for the community to assist in reunification efforts.

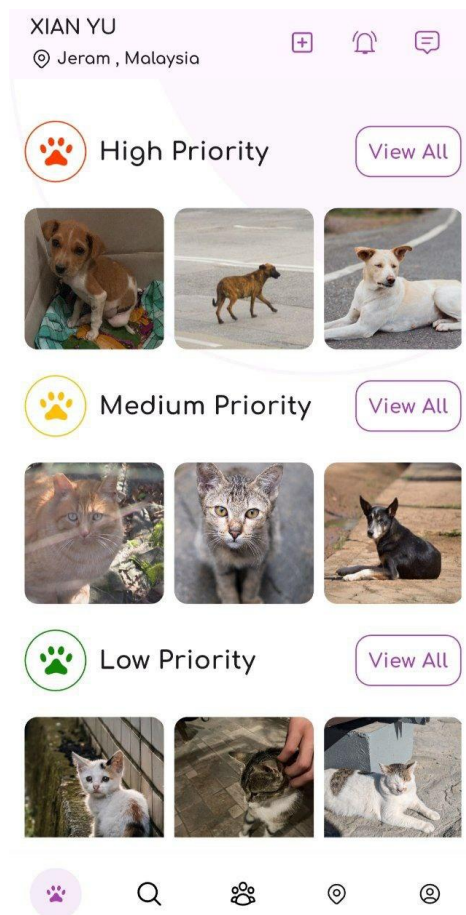


Figure 2.15 Home Page Pawrpose

2. View All Page

Users can select a low, medium, or high priority level to view a list of animals needing attention. Each animal's image includes a label according to its condition, such as "Lost," "Stray," or "Adopt," to let users have a better understanding of the situation of animals. Besides, users can click "View on Map" to see the animal's last known location displayed on an interactive map.

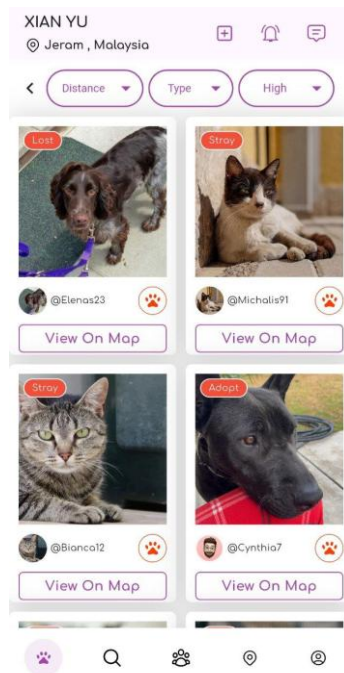


Figure 2.16 View All Animals Page

3. Search Page

On this page, users can discover animals in need of help. It also displays the latest blog posts from Pawrpose and shows nearby users, allowing volunteers to connect and coordinate rescue operations more effectively.

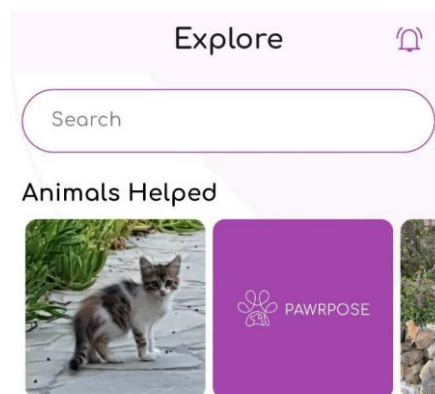


Figure 2.17 Search Page for Pawrpose

4. Community Page

Pawrpose includes a community page featuring both blogs and a community section. The blog page covers a variety of topics such as effective ways to support animal shelters and rescues, responsible pet care, choosing the right veterinary services, and

protecting animals. No matter the topic, users can find helpful and informative blogs related to animal welfare here.

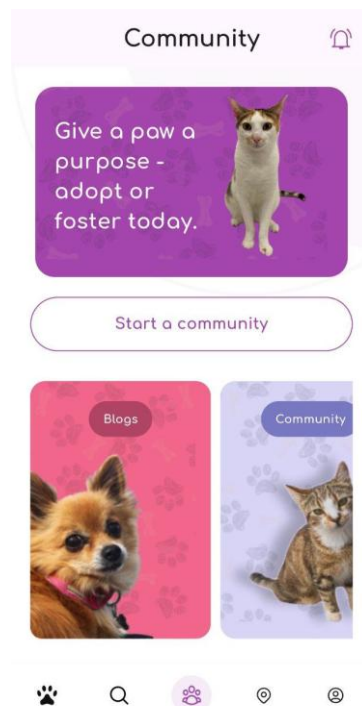


Figure 2.18 Community Page for Pawrpose

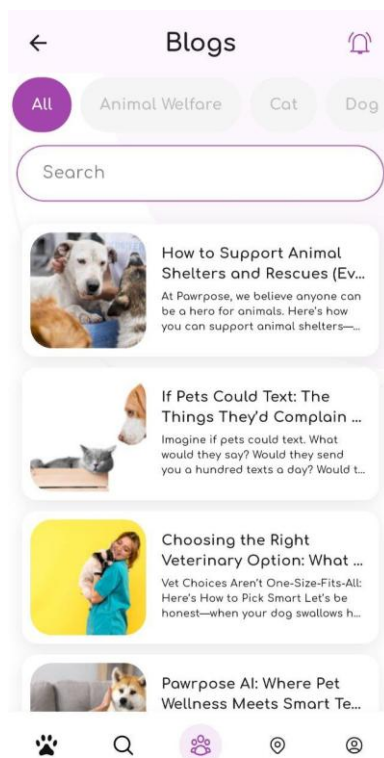


Figure 2.19 Blog Section



Figure 2.20 Blog Details Section

5. Profile Page

The profile page displays users' information, such as username and profile picture, enhancing the security of identity in user authorization and authentication.

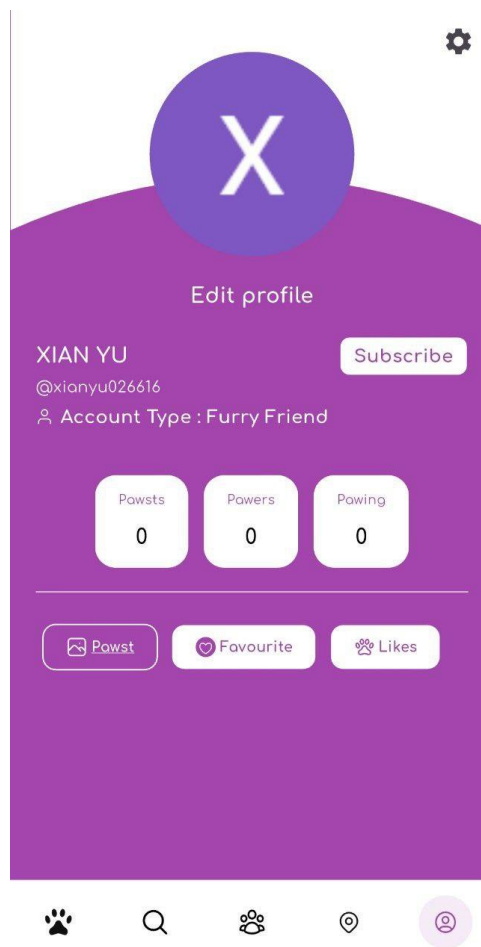


Figure 2.21 Profile Page for Pawrpose

Strengths and Weaknesses of Pawrpose

Table 2.3 Strengths and Weaknesses of Pawrpose

Pawrpose	
Strengths	Weaknesses
Categorizes lost animals by urgency levels (high, medium, low), helping prioritize rescue efforts.	Primarily focuses on lost pets and pet adoption, not emergency rescue of injured animals.
Inform users of the animals' last known locations on an interactive map.	No route planning or map showing the location of nearby rescue centres.
Includes a community page with blogs and articles on animal welfare and pet care.	Although it has blogs, it lacks structured educational resources about different animal species and wildlife handling.
Helps users connect with nearby volunteers to coordinate rescue efforts.	

2.4 Summary

Table 2.4 Strengths and Weaknesses of Existing Animal Rescue Applications

Application	Strengths	Weaknesses
Missing Pets	User-friendly interface	Limited animal identification features
	An effective lost pet notification system	No automated species recognition
	Map integration for location tracking	Requires manual data entry
Pet Adoption	Comprehensive animal listing	Lack of educational resources
	Detailed animal descriptions	No integration with rescue services
	Good filtering system by animal type and time	Limited integration with location tracking
Pawrpose	Strong community features	Basic search functionality

	Blog section for educational content	No automated animal recognition
	Comprehensive user profile system	Limited emergency response capabilities

Table 2.5 ResQMe Proposed Solutions

Weaknesses in Existing Systems	Solution in ResQMe
Lack of an animal recognition system relies on manual reports.	Implement a CNN animal recognition model [1] to detect animal names by uploading a photo
Lack of structured animal educational resources.	Provides animal educational resources by implementing according to animal categories.
Limited route planning direct to animal rescue centers.	Integrated real-time GPS navigation using the Haversine Formula [2] and Mapbox API [11] to find the nearest clinics.
Focusing only on rescuing pets, ignoring wildlife rescue operations	Provide interaction between users and admins through wildlife report submission and volunteer participation.
Limited real-time notifications for posting.	Always update the real-time notification once the action is taken.

In summary, applications such as Pawrpose, Pet Adoption, and Missing Pets provide rudimentary functions like tracking down missing pets and perusing pet information; however, they are devoid of essential features. Interestingly, **none of them use Convolutional Neural Networks (CNNs)** to recognize animals, which means they are missing out on the opportunity to increase rescue response and species identification. Additionally, these applications **lack route planning tools** that would direct users to local animal or pet rescue facilities and offer very little in the way of instructional content. By creating a more intelligent, **educational**, and rescue-focused application, this project seeks to close these gaps and provide better assistance to both users and animals in need.

Chapter 3

System Methodology/Approach OR System Model

3.1 System Design Diagram

Software Development Life Cycle (SDLC) phases in Figure 3.1, the project workflow, database CRUD operations in Figure 3.2, and the design of UI and UX are the core methodologies to develop ResQMe, an intelligent animal rescue mobile application.

3.1.1 Software Development Life Cycle (SDLC) phases

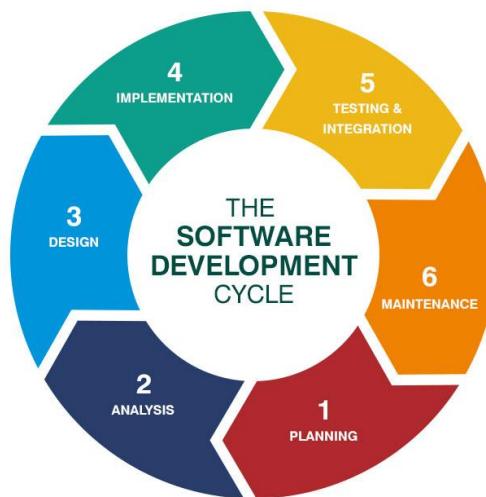


Figure 3.1 Software Development Life Cycle (SDLC)

1. Planning

In the planning stage, key tasks included defining the project scope based on objectives. Besides, it is important to identify stakeholders, including end-users, volunteers, administrators, and rescue organisations. Android-Firebase technology stack chosen for rapid prototyping and database storing. A high-level resource plan is defined, such as development laptop specifications, Android test device, Google APIs, and Firebase subscription.

2. Analysis

Functional requirements were derived from three problem statements and received feedback from interviews with potential users.

- User must be able to create a new account, log in, log out, and reset their passwords.
- Wildlife rescue reporter requires their real-time location, urgency level, photo upload, and description for the animal reported.
- Administrators can either approve or reject users' submissions from the report or volunteer form.
- Notification must reach both users' and administrators' mailboxes after submission, approval, or rejection action is taken.

Non-functional requirements are defined as high availability, high-demand data load, data security, and local caching of sessions.

3. Design

Design covered various database designs, architectures and models, database schema, and the interaction process.

Database collected from different species, architectures such as client-cloud pattern with Android front-end and Firebase backend. Besides, the database schema from collections in Firebase, such as `admin_config`, `animal_descriptions`, `notifications`, `reports`, `saved_animals`, `users`, `volunteer_applications`, and `volunteer_forms`, is clearly defined to make sure all flows are clear and smooth.

UI and UX were designed as wireframes for each screen, a colour palette for accessibility, such as a sharpen keyword with a unique colour, and a navigational drawer and footer layout.

4. Implementation

The system was implemented in iterative increments. Development in user management, which includes `FirebaseSessionManager`, email verification, and profile editing. With the help of OCR integration [12], the `OCRHelper` uses Google ML Kit [13] to extract the text from IC and perform validation for each user. Besides, in the report and volunteer modules, activities and adapters can handle submissions, image uploads, and real-time updates in Firebase. Notifications will be received after that in the admin dashboard for approval workflows.

5. Testing and Integration

Multiple layers of Testing:

- Unit Tests – Validating helpers such as OCR text extraction and Base64 image decoding.
- Integration Tests – Verifying end-to-end flows: report submission → admin approval → user notification.
- UI/UX Testing – Make the design clean and not miss keywords.
- Performance Tests – Measuring notification latency and OCR recognition accuracy.

6. Maintenance

Maintenance tasks include a heavy load on the system and updating libraries. Future improvements planning is important, involving deep-learning animal recognition, multi-language support, and community section development.

3.1.2 CRUD Operations

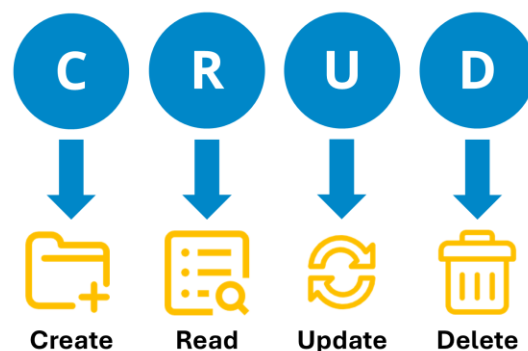


Figure 3.2 CRUD Operations

Table 3.1 CRUD Operations

Entity	Create	Read	Update	Delete
User	Register a new account, store profile image.	Retrieve profile/session data.	Update username, phone, profile image, password reset.	Delete saved animal.
Rescue Report	Submit a new wildlife report	Retrieve all reports by user	Admin approves/rejects,	User cancels process

	with Base64 image and location.	or status for the admin.	adds notes, changes urgency.	submission of the report
Volunteer Application	Submit the application form with a message.	View application status.	Status changed after the action of approval was done.	User cancels the application.
Educational Description	Registered users submit new description content.	Public users browse species details.	Admin approves or rejects the submission.	User canceled submission.

3.2 System Architecture

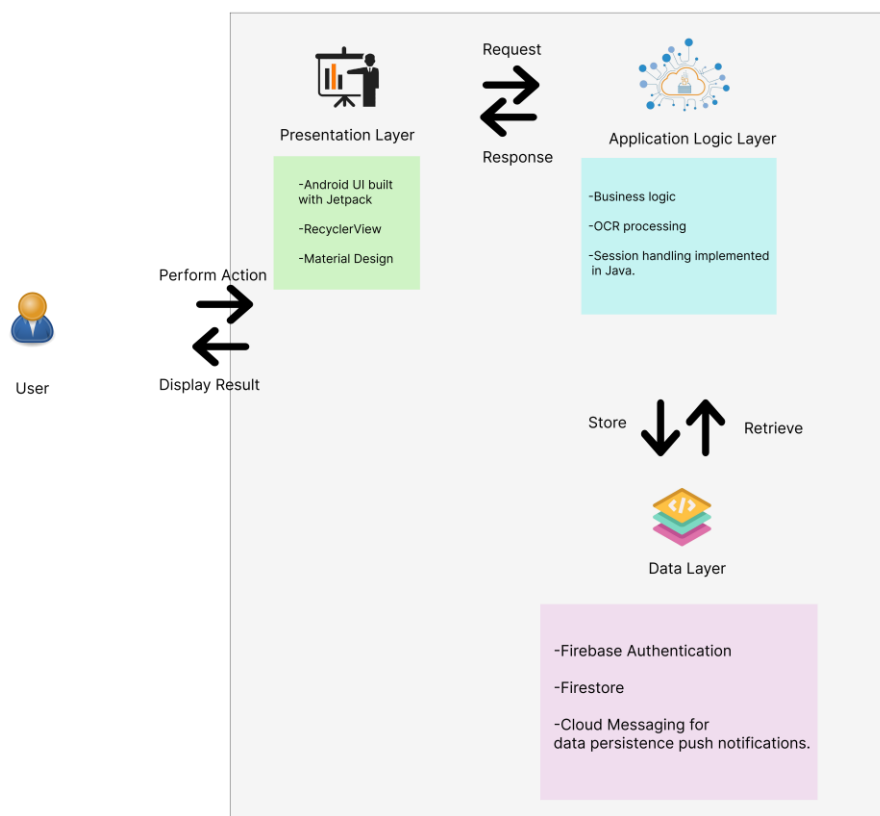


Figure 3.3 System Architecture

From Figure 3.3, the architecture design, the **Presentation Layer** consists of the elements the user will interact with, such as the home page, camera page, location page, and saved animal

page. For example, this contains the home page with the stored animals' area, location features, camera functionality for animal identification, instructional information, and hotlines. Because of the user interface's easy design, users may easily switch between various functionalities using the bottom navigation bar.

Besides, the **Application Logic Layer** manages essential functionalities. It involves using both the pre-trained MobileNetV2 model [3] for animal recognition and a custom CNN model with custom datasets. This layer classifies animals, processes user-selected or taken photos, and prepares pertinent data for presentation. When users click "Learn More" about animals, it also handles the implementation of Application Programming Interface (API) integration, which yields more details. Another logic layers include functionalities of components, such as click search icon, will trigger it to search the keywords, submit button trigger will handle interaction between administrators and users.

The **Data Layer** handled data retrieval and persistence. The TFLite models for offline animal detection save information locally, and in online mode, which is cloud storage if the user has completed the login process. It also connects to external APIs for more animal data. This layer makes sure that the program can run even with spotty connectivity and that user preferences are preserved between sessions. Firebase authentication makes data private and secure.

Eventually, the architecture demonstrates a clear information flow that user actions cause requests to be sent to the application logic layer, which uses data and models that have been stored to process the requests and return responses that are shown to the user.

3.3 Animal Recognition Implementation

A personal computer (PC) and an Android smartphone were the two main pieces of hardware used in the creation. While the Android smartphone was used to test app functions such as camera capture, gallery image selection, and live animal prediction, the PC was primarily used for application development, dataset preparation, and CNN model training.

The following steps are included before generating a complete custom CNN model:

1. Dataset Collection
 - A variety of animal picture datasets from Kaggle were downloaded, encompassing a wide variety of taxa, including birds, amphibians, mammals, reptiles, and marine life.
2. Dataset Organization and Cleaning
 - Blurry images, duplicates, or images with the wrong labels were manually eliminated.
 - Every picture was arranged into a folder based on the class of animal.
3. Image Resizing
 - To guarantee a constant input size for the CNN model, all photos were shrunk to a specified dimension (224x224 pixels).
4. Data Augmentation
 - To strengthen the model's resistance to variations in real-world graphics, a variety of augmentation approaches were used to artificially extend the dataset.
5. Training and Validation Splitting
 - To assess model performance during training, the cleaned and enhanced datasets were divided into training and validation sets in an 80:20 ratio.
6. Model Development
 - To address the multi-class animal classification job, a MobileNetV2 architecture was chosen and optimized.
7. Model Saving
 - The model was stored and ready to be converted to a format that was compatible with mobile devices after it had reached a suitable level of accuracy.

3.3.1 Data Collection and Preparation

A vast number of animal photos from different datasets are combined to train the Convolutional Neural Network (CNN) model. In particular, the dataset contained:

- 139,362 training images across 269 distinct animal classes.
- 27,344 validation images across the same 269 classes.

To guarantee consistency and quality, a crucial first step was to achieve dataset preparation. To avoid overfitting and enhance model generalization, animal photos with a range of lighting conditions, angles, and settings were gathered from databases, public archives, and Kaggle.

After being gathered, the photos were carefully arranged. A systematic and user-friendly folder system was created by assigning each image to the directory that corresponded to its unique class label. In addition to facilitating effective data handling during training, this organization made sure that data and its labels corresponded clearly. Training sets and validation sets are separated from the datasets. The CNN model was taught the unique characteristics of each animal class using the training set, and its performance and capacity for generalization were evaluated regularly during the training phase using the validation set.

The dataset underwent a crucial preprocessing step before training could start. All the photographs were reduced to an input size that matched the CNN's design to guarantee consistency in the data provided to the model. Efficiency and stability of training can be improved by normalized pixel values; this project normalized from 0 to 1. Real-world testing was stimulating using a data augmentation approach such as flipping, rotation, and color correction.

Overall, the custom CNN model's success was largely due to this meticulous and systematic approach to dataset preparation. A high-quality, well-structured, and appropriately preprocessed dataset made it possible for the model to learn significant representations and produce precise predictions on brand-new, untested animal photos.

3.3.2 Data Augmentation

The training images were subjected to data augmentation techniques to increase the CNN model's resilience and reduce the possibility of overfitting. TensorFlow's ImageDataGenerator feature was used.

By creating altered versions of preexisting photos, augmentation was used to augment the training dataset. When presented with diverse, real-world animal photos, the CNN model's

performance improved because of its ability to acquire more generic characteristics and patterns.

3.3.3 Model Training Process

The training processes followed the structured workflow:

1. Pre-trained Model Loading: The basis model was MobileNetV2.
2. Freezing Early Layers: To preserve the low-level features that had already been learned, the early convolutional layers were frozen.
3. Adding Custom Layers: To modify the model for the animal classification task, new classification layers were added.
4. Training Strategies:
 - When there was no more improvement in validation loss, training was stopped using early stopping.
 - Techniques for lowering learning rates were used to dynamically modify the learning pace.

These techniques avoided overfitting and maximized training time.

3.3.4 Training Performance Visualization

To track the model's learning over time, accuracy and loss graphs were plotted for both training and validation data.

The following steps were included in the visualization process:

- Plotting the accuracy of training and validation across periods
- Training and validation loss plotted across epochs

These graphs in Figure 3.4 and Figure 3.5 offered an understandable visual depiction of:

- The accuracy curve shows how effectively models learned to accurately classify animals over time.
- How the model's performance improved as its loss decreased during training (loss graph)

Better model fine-tuning was made feasible by the early detection of overfitting, underfitting, or other anomalies by trend analysis.

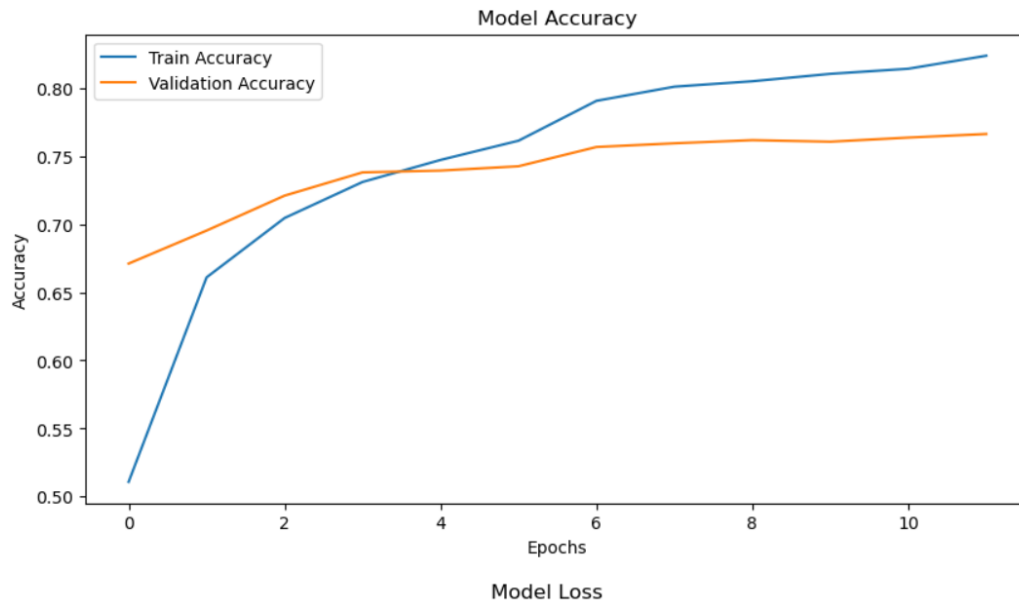


Figure 3.4 Graph Custom-Trained CNN Model Training Accuracy

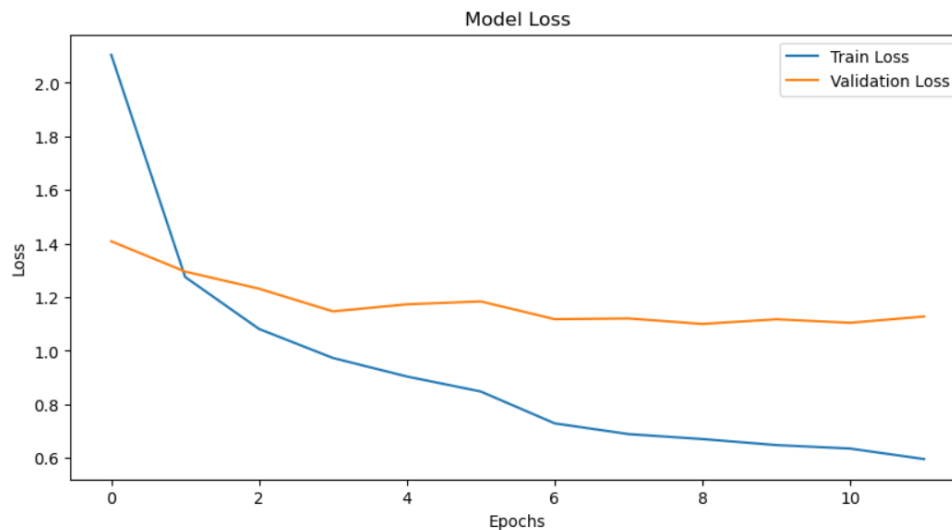


Figure 3.5 Graph Custom-Trained CNN Model Training Loss

3.3.5 Model Performance

The training phase's outcomes showed promising performance metrics:

- Training Accuracy: 82.37%
- Training Loss: 0.5960
- Validation Accuracy: 76.63%
- Validation Loss: 1.1281

When tested on unseen validation images, these values show that the model was able to learn from the training data efficiently while retaining a respectable level of generalization ability.

3.4 Use Case Diagram

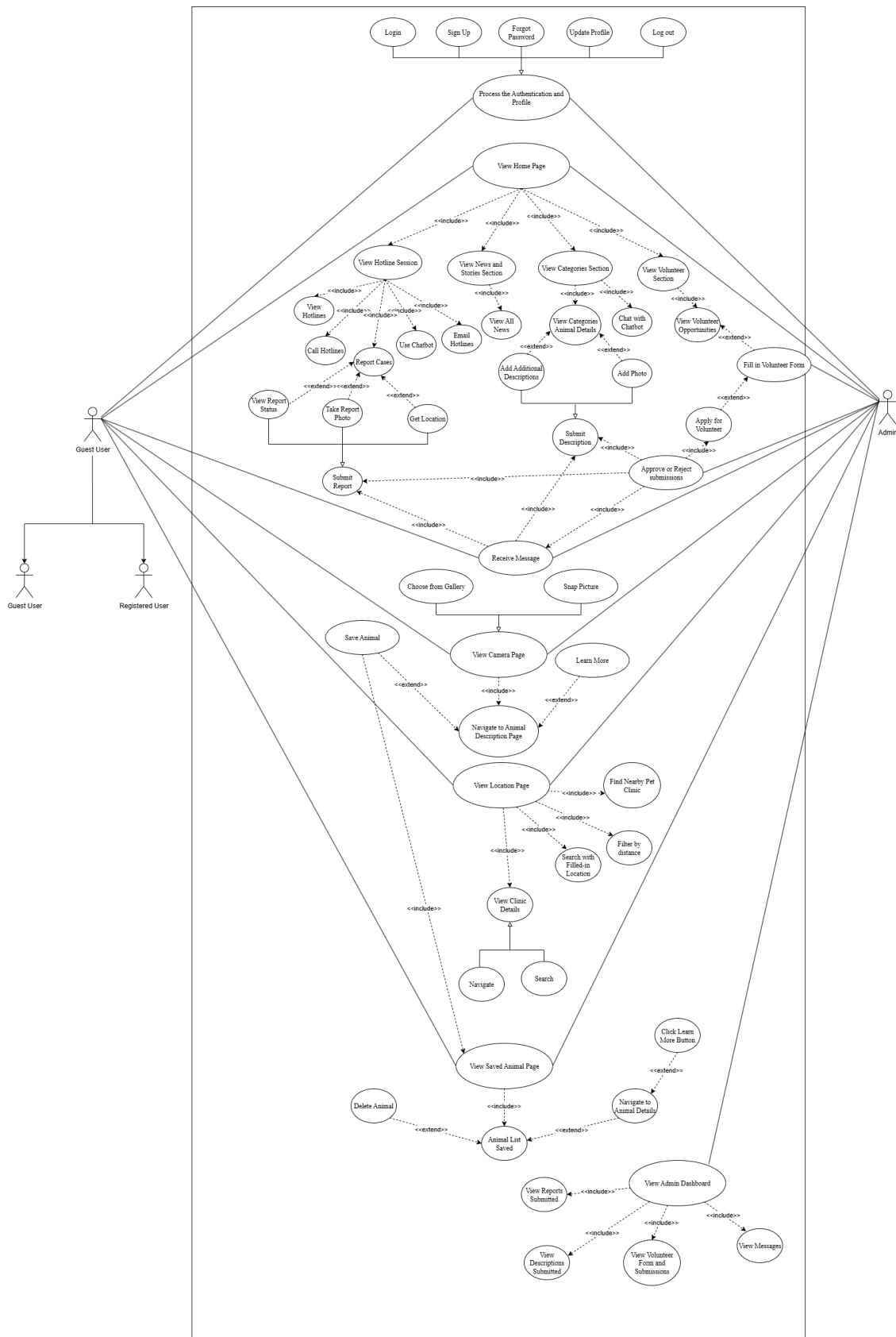


Figure 3.6 Use Case Diagram

3.4.1 Use Case Diagram Description

Use Case 1: Login

Table 3.2 Login Use Case

Use Case Name: Login	ID: 1	Important Level: High
Primary Actor: Guest User, Admin	Use Case Type: detail, essential	
Stakeholders and Interests: Guest User, Admin - Wants to access personalized features and secure areas of the application, such as the report and description sections.		
Brief Description: This use case allows a guest user or admin to authenticate and gain access to the system with their credentials.		
Trigger: User or admin opens the drawer on the home page and clicks the login/sign-up button. Type: External		
Relationships: Association: Guest User, Admin Include: Provide Authentication and Profile Extend: Forgot Password, Update profile Generalization:		
Normal Flow of Events: 1. User opens the app. 2. User navigates to the drawer and clicks the login/sign up button. 3. System displays the login form. 4. User logs in with their existing account if they have one. 5. User logs in successfully if the information is correct.		
SubFlows: 1. Admin opens the app. 2. Admin navigates to the drawer and clicks the login/sign up button. 3. System displays the login form. 4. Admin clicks “Admin Login” button 5. Admin logs in with their existing account. 6. Admin logs in successfully.		
Alternate/Exceptional Flows: 4a. Invalid credentials: System displays error message and prompts retry		

Use Case 2: Sign Up

Table 3.3 Sign Up Use Case

Use Case Name: Sign Up	ID: 2	Important Level: High
Primary Actor: Guest User, Admin	Use Case Type: detail, essential	
Stakeholders and Interests: Guest User, Admin - Wants to create a new account to access application features.		
Brief Description: This use case allows a new user or new admin to register and create an account on the sign-up page.		
Trigger: User and admin open the drawer on the home page and click the login/sign-up button, then click the create new account button on the login page. Type: External		
Relationships: Association: Guest User, Admin Include: Provide Authentication and Profile Extend: Login Generalization:		
Normal Flow of Events: 1. User opens the app. 2. User navigates to the drawer and clicks the login/sign up button. 3. System displays the login form. 4. User clicks the “Create New Account” button. 5. User navigates the sign up form. 6. User keys in information and verify their email address. 7. User signs up successfully.		
SubFlows: 1. Admin opens the app. 2. Admin navigates to the drawer and clicks the login/sign up button. 3. System displays the login form. 4. Admin clicks the “Admin Login” button. 5. System navigates to admin sign up page. 6. Admin keys in their information and admin pass code. 7. Admin signs up successfully.		

Alternate/Exceptional Flows: 6a. Validation fails: System displays error messages for invalid fields

Use Case 3: Update Profile

Table 3.4 Update Profile Use Case

Use Case Name: Update Profile	ID: 3	Important Level: High
Primary Actor: Registered User	Use Case Type: detail, essential	
Stakeholders and Interests: Registered User - Wants to modify personal information and account settings.		
Brief Description: This use case allows registered users to update their profile information.		
Trigger: User opens the drawer on the home page and clicks the profile settings. Type: External		
Relationships: Association: Registered User Include: Provide Authentication and Profile Extend: Forgot Password Generalization:		
Normal Flow of Events: 1. User opens the app. 2. User navigates to the drawer and clicks the profile settings button. 3. User navigates to profile settings. 4. User changes and modifies information. 5. User clicks save changes. 6. System update user’s profile on the home page and Firebase.		
SubFlows: None		
Alternate/Exceptional Flows: 6a. Validation fails: System highlights errors and requests corrections		

Use Case 4: Log Out

Table 3.5 Log Out Use Case

Use Case Name: Log Out	ID: 4	Important Level: High
Primary Actor: Registered User, Admin	Use Case Type: detail, essential	
Stakeholders and Interests: Registered User, Admin - Wants to log out of their account logged in before.		
Brief Description: This use case allows registered users or admin to log out of their existing account.		
Trigger: User or admin opens the drawer on the home page and clicks the log out button.		
Relationships: <div>Association: Registered User, Admin</div> <div>Include: Provide Authentication and Profile</div> <div>Extend:</div> <div>Generalization:</div>		
Normal Flow of Events: <div>1. User or admin opens the app.</div> <div>2. User or admin navigates to the drawer and clicks the log out button</div> <div>3. System pop-up alert confirmation “Are you sure you want to logout?”</div> <div>4. User or admin click “yes”.</div> <div>5. User or admin logged out successfully.</div>		
SubFlows: None		
Alternate/Exceptional Flows: None		

Use Case 5: View Home Page

Table 3.6 View Home Page Use Case

Use Case Name: View Home Page	ID: 5	Important Level: High
Primary Actor: User, Admin	Use Case Type: detail, essential	
Stakeholders and Interests: User, Admin - Wants to access different sections like Hotline, Volunteer, Categories, and News.		

Brief Description: This use case allows the user or admin to open the Home Page and navigate to various sections.
Trigger: User or admin opens the app or clicks the Home button.
Relationships: Association: User, Admin Include: Hotline Session, News and Stories Section, Categories Section, Volunteer Section Extend: Generalization:
Normal Flow of Events: 1. User opens the app. 2. User navigates home page via bottom footer navigation.
SubFlows: None
Alternate/Exceptional Flows: None

Use Case 6: View Categories Section

Table 3.7 View Categories Section Use Case

Use Case Name: View Categories Section	ID: 6	Important Level: High
Primary Actor: User	Use Case Type: detail, essential	
Stakeholders and Interests: User - Wants to browse different types of animals.		
Brief Description: This use case allows the user review different types of animals.		
Trigger: User navigate to home page and click one of the types of categories.		
Relationships: Association: User Include: View Categories Animal Details Extend: Add Additional Descriptions, Add photo Generalization:		
Normal Flow of Events: 1. User or admin opens the app. 2. User or admin navigates to home page.		

3. User or admin clicks one of the types from categories section.
4. System display various types of animals from the type user chosen.
SubFlows: None
Alternate/Exceptional Flows: None

Use Case 7: View Hotline Session

Table 3.8 View Hotline Session Use Case

Use Case Name: View Hotline Session	ID: 7	Important Level: High
Primary Actor: User, Admin	Use Case Type: detail, essential	
Stakeholders and Interests: User, Admin - Wants to view all states of hotlines in Malaysia		
Brief Description: This use case allows the user or admin to review all hotlines in each state in Malaysia.		
Trigger: User or admin navigates to the hotline banner and clicks the “View Hotline” button.		
Relationships: <div>Association: User, Admin</div> <div>Include: View Hotlines, Call Hotlines, Report Cases, Use Chatbot, Email Hotlines</div> <div>Extend:</div> <div>Generalization:</div>		
Normal Flow of Events: <div>1. User or admin opens the app.</div> <div>2. User or admin navigates the hotline banner.</div> <div>3. User or admin clicks the “View Hotline” button.</div> <div>4. User or admin navigates to the hotline page.</div>		
SubFlows: None		
Alternate/Exceptional Flows: None		

Use Case 8: View News and Stories Section

Table 3.9 View News and Stories Section Use Case

Use Case Name: View News and Stories Section	ID: 8	Important Level: High
Primary Actor: User, Admin	Use Case Type: detail, essential	
Stakeholders and Interests: User, Admin - Wants to view all news and stories.		
Brief Description: This use case allows the user or admin to review all news and stories for animals.		
Trigger: User or admin navigates to the News and Stories section in home page.		
Relationships: Association: User, Admin Include: View All News Extend: Generalization:		
Normal Flow of Events: 1. User or admin opens the app. 2. User or admin navigates home page. 3. User or admin scroll down to news and stories section.		
SubFlows: None		
Alternate/Exceptional Flows: None		

Use Case 9: View Volunteer Section

Table 3.10 View Volunteer Section Use Case

Use Case Name: View Volunteer Section	ID: 9	Important Level: High
Primary Actor: Registered User, Admin	Use Case Type: detail, essential	
Stakeholders and Interests: Registered User, Admin - Wants to volunteer for animal rescue.		
Brief Description: This use case allows the user or admin to volunteer in animal-saving activities.		

Trigger: Registered user or admin navigates to the volunteer banner and click “Volunteer Now” button.
Relationships: Association: Registered User, Admin Include: View Volunteer Opportunities Extend: Generalization:
Normal Flow of Events: 1. User or admin opens the app. 2. User or admin navigates to the volunteer banner. 3. User or admin clicks the “Volunteer Now” button.
SubFlows: None
Alternate/Exceptional Flows: None

Use Case 10: Receive Message

Table 3.11 Receive Message Use Case

Use Case Name: Receive Message	ID: 10	Important Level: High
Primary Actor: Registered User, Admin	Use Case Type: detail, essential	
Stakeholders and Interests: Registered User, Admin - Wants to review the submission status.		
Brief Description: This use case allows the user or admin to receive message for submission status.		
Trigger: Registered user or admin navigates to the drawer and click “Messages” button.		
Relationships: Association: Registered User, Admin Include: Submit Report, Submit Description Extend: Generalization:		
Normal Flow of Events: 1. User or admin opens the app. 2. User or admin navigates to the drawer in home page.		

3. User or admin click “Messages” button.
SubFlows: None
Alternate/Exceptional Flows: None

Use Case 11: View Camera Page

Table 3.12 View Camera Page Use Case

Use Case Name: View Camera Page	ID: 11	Important Level: High
Primary Actor: User, Admin	Use Case Type: detail, essential	
Stakeholders and Interests: User, Admin - Wants to capture or select an animal image.		
Brief Description: This use case allows the user or admin to access the camera page to snap or pick an image.		
Trigger: User or admin clicks the camera page.		
Relationships: Association: User, Admin Include: Navigate to Animal Description Page Extend: Generalization: Choose from Gallery, Snap Picture		
Normal Flow of Events: 1. User or admin opens the app. 2. User or admin navigates to Camera Page. 3. User or admin snaps a picture or selects from the gallery.		
SubFlows: None		
Alternate/Exceptional Flows: None		

Use Case 12: View Location Page

Table 3.13 View Location Page Use Case

Use Case Name: View Location Page	ID: 12	Important Level: High
Primary Actor: User, Admin	Use Case Type: detail, essential	
Stakeholders and Interests:		
User, Admin - Wants to view all pet clinics nearby.		

Brief Description: This use case allows the user or admin to review the nearest pet clinics.
Trigger: User or admin clicks the location page.
Relationships: Association: User, Admin Include: View Clinic Details, Search with Filled-In Location, Filter by Distance, Find Nearby Pet Clinic Extend: Generalization:
Normal Flow of Events: 1. User or admin opens the app. 2. User or admin navigates to the Location page.
SubFlows: 1. User or admin opens the app. 2. User or admin navigates to the Location page. 3. User and admin fill in their desired location address. 4. System finds the nearest location based on the filled-in location.
Alternate/Exceptional Flows: None

Use Case 13: View Saved Animal Page

Table 3.14 View Saved Animal Page Use Case

Use Case Name: View Saved Animal Page	ID: 13	Important Level: High
Primary Actor: Registered User, Admin	Use Case Type: detail, essential	
Stakeholders and Interests: Registered User, Admin - Wants to view the saved animal from the animal description page.		
Brief Description: This use case allows the registered user or admin to review their saved animal.		
Trigger: Registered User or admin clicks the saved animal page		
Relationships: Association: Registered User, Admin		

Include: Animal List Saved Extend: Generalization:
Normal Flow of Events: User or admin opens the app. User or admin navigates to the Saved Animal Page
SubFlows: None
Alternate/Exceptional Flows: None

Use Case 14: Fill in Volunteer Form

Table 3.15 Fill in Volunteer Form Use Case

Use Case Name: Fill in Volunteer Form	ID: 14	Important Level: High
Primary Actor: Admin	Use Case Type: detail, essential	
Stakeholders and Interests: Admin - Wants to fill in volunteer activities.		
Brief Description: This use case allows the admin to fill in all volunteer activities inside a form.		
Trigger: Admin navigates to drawer and clicks “Volunteer Management” button.		
Relationships: Association: Admin Include: Extend: Apply for Volunteer Generalization:		
Normal Flow of Events: Admin opens the app. Admin navigates to drawer and clicks “Volunteer Management” button. Admin clicks the “+” button on the bottom right corner.		
SubFlows: None		
Alternate/Exceptional Flows: None		

Use Case 15: View Admin Dashboard

Table 3.16 View Admin Dashboard Use Case

Use Case Name: View Admin Dashboard	ID: 15	Important Level: High
Primary Actor: Admin	Use Case Type: detail, essential	
Stakeholders and Interests: Admin - Wants to view the admin dashboard.		
Brief Description: This use case allows the admin to review users’ data from the dashboard.		
Trigger: Admin logs in to the admin account and clicks the dashboard page.		
Relationships: Association: Admin Include: View Reports Submitted, View Descriptions Submitted, View Volunteer Form and Submissions, View Messages Extend: Generalization:		
Normal Flow of Events: 1. Admin opens the app. 2. Admin logs in with the admin’s account. 3. Admin navigates to the dashboard page.		
SubFlows: None		
Alternate/Exceptional Flows: None		

3.5 Activity Diagram

3.5.1 Log In / Sign Up Activity Diagram

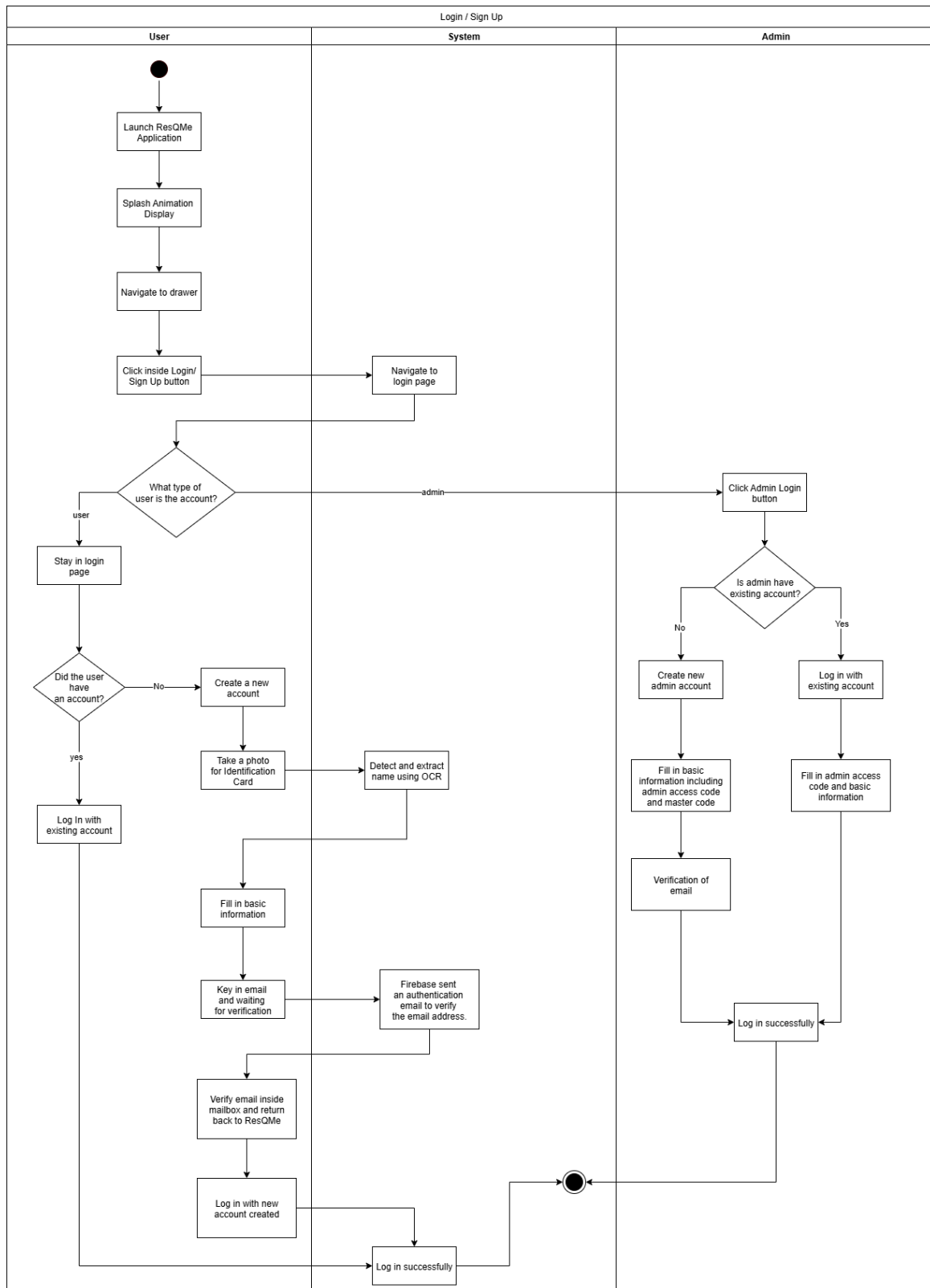


Figure 3.7 Log In / Sign Up Activity Diagram

3.5.2 Forgot Password Activity Diagram

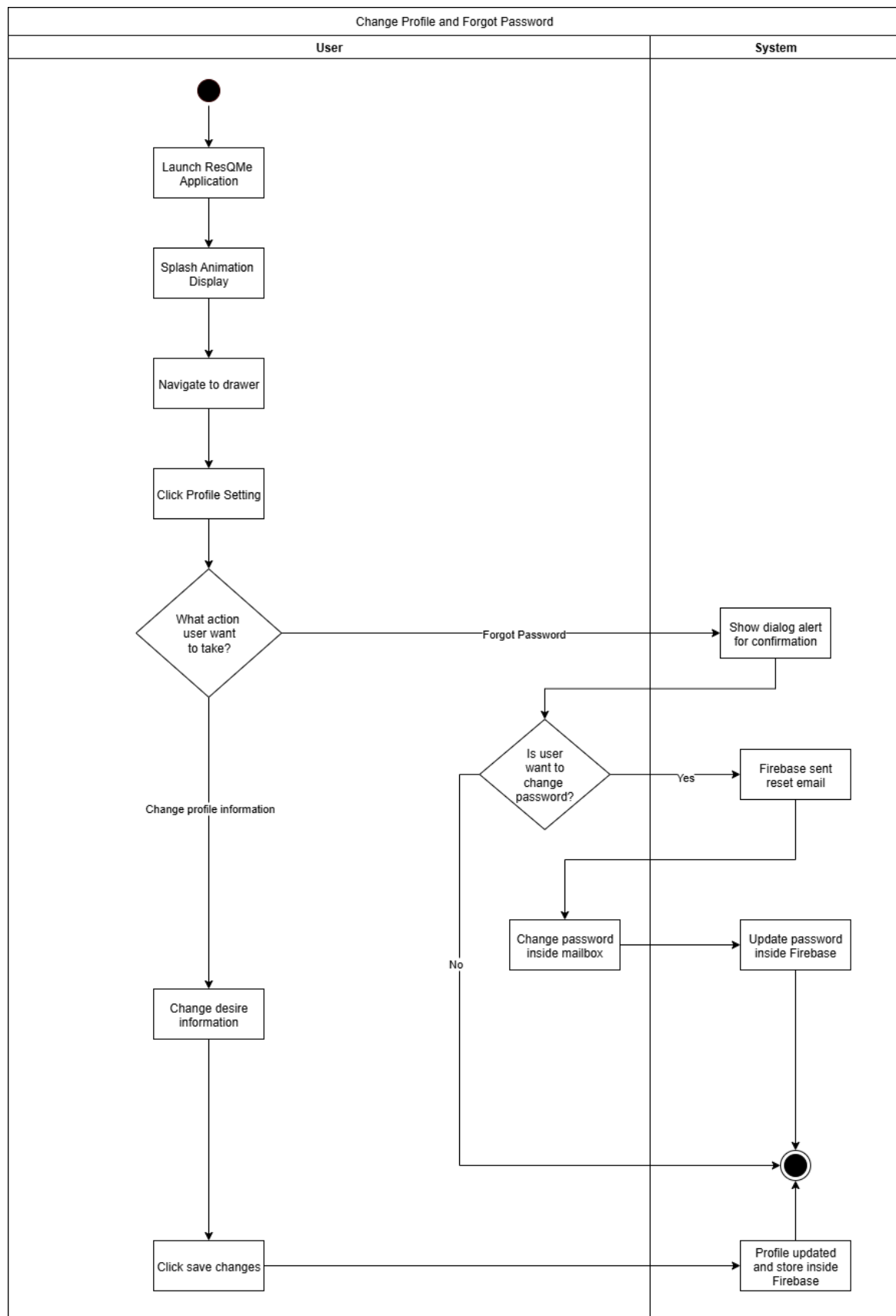


Figure 3.8 Forgot Password Activity Diagram

3.5.3 Banner Activity Diagram

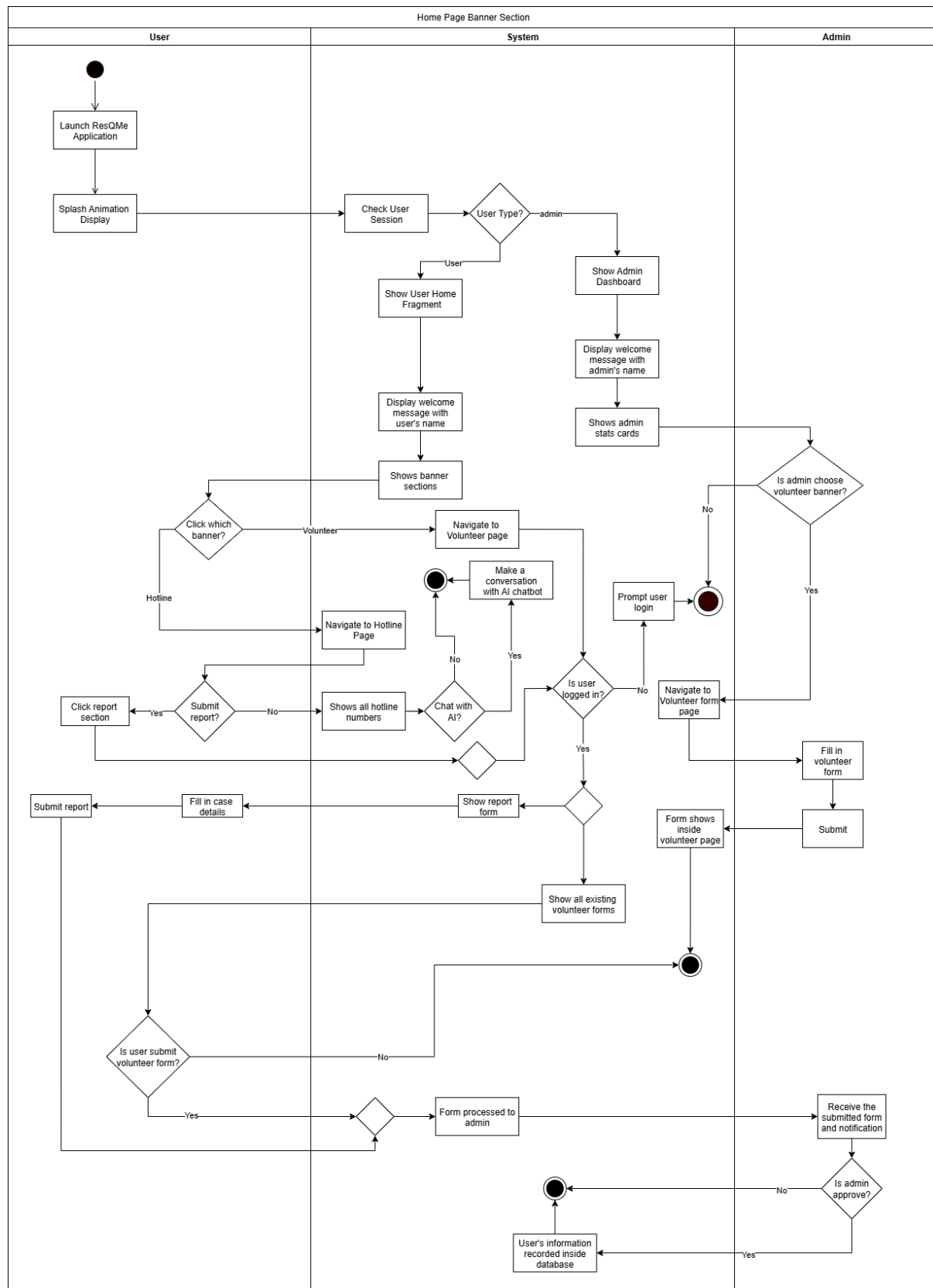


Figure 3.9 Banner Activity Diagram

3.5.4 Categories Activity Diagram

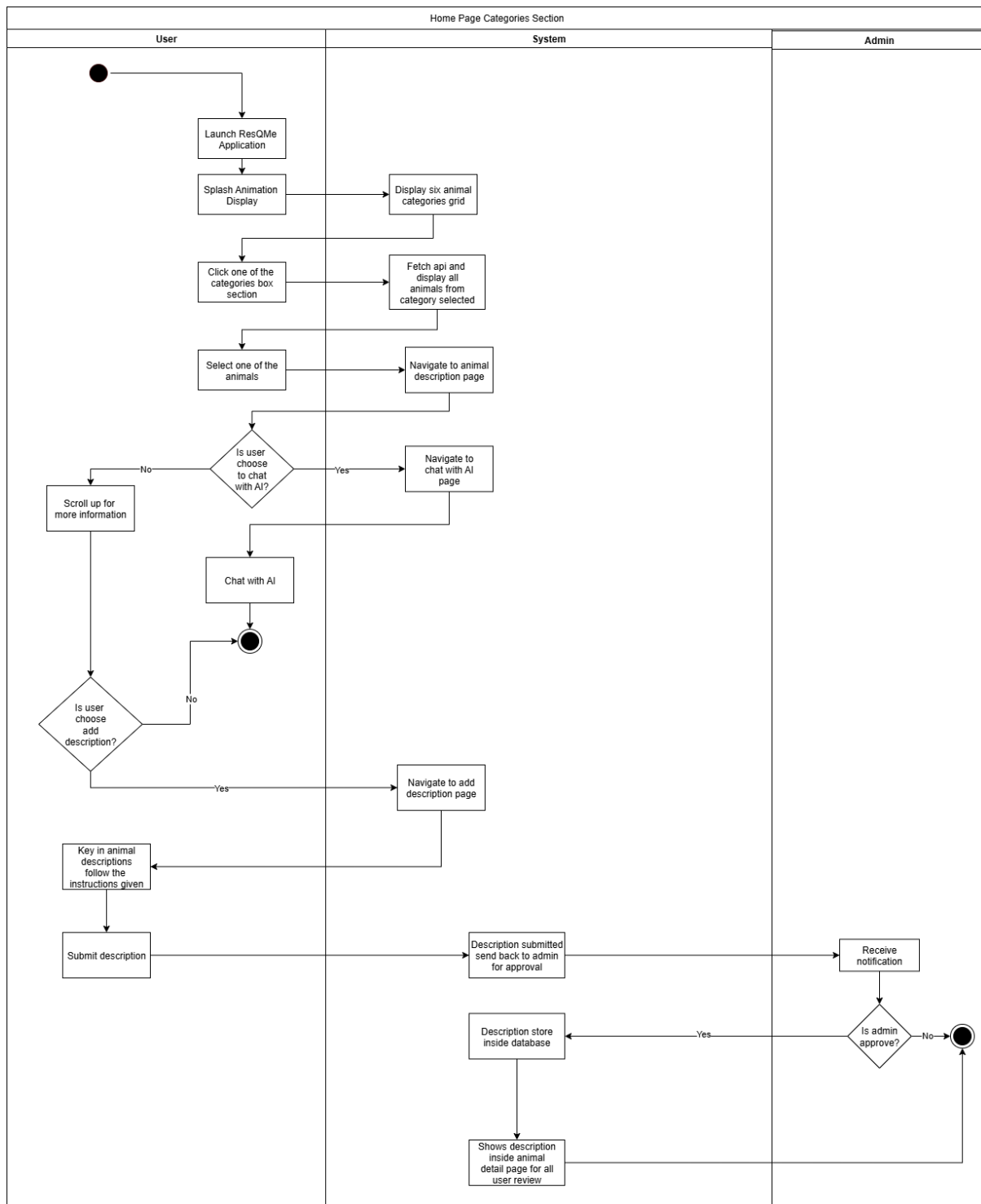


Figure 3.10 Categories Activity Diagram

3.5.5 News and Stories Activity Diagram

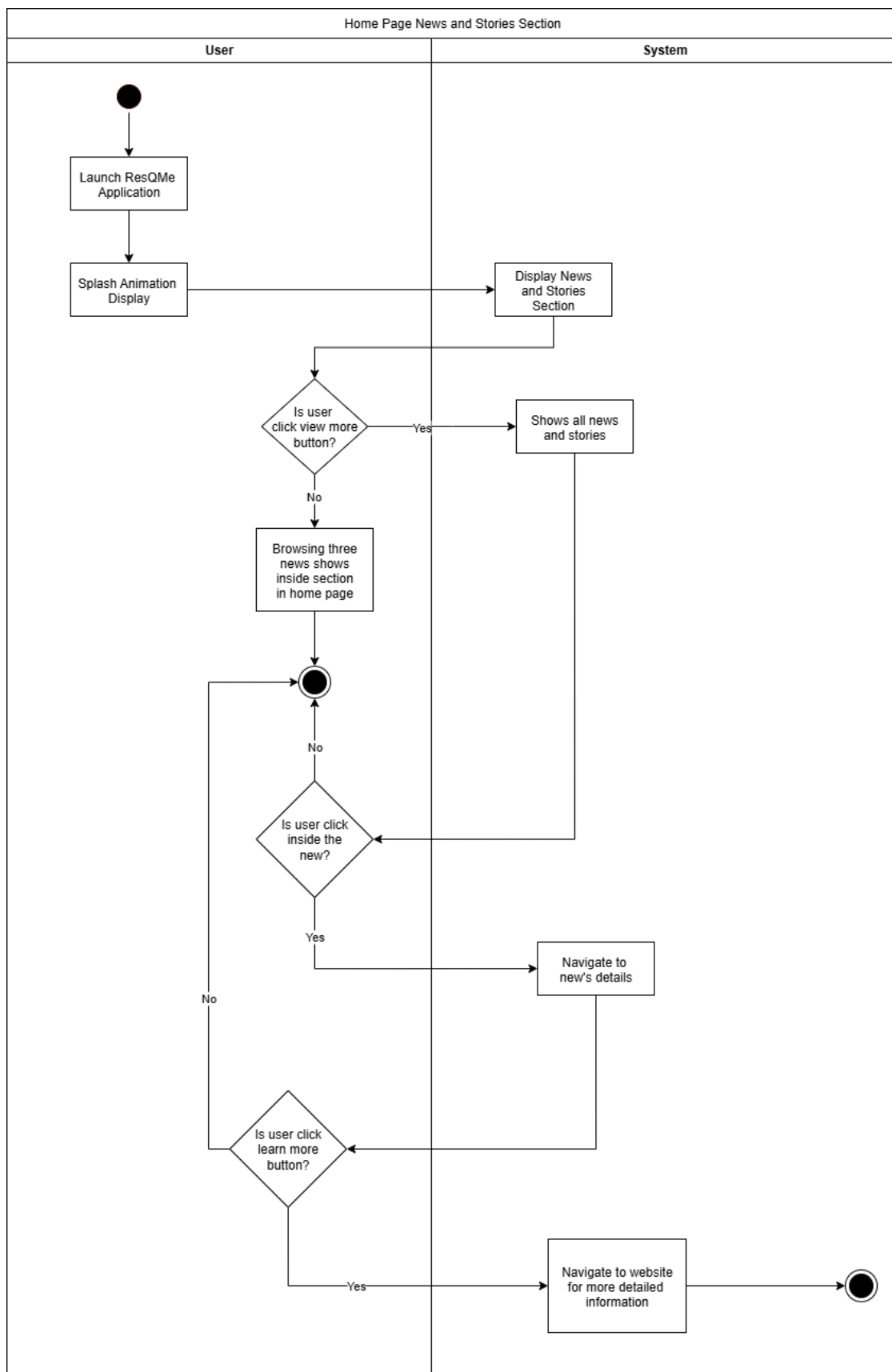


Figure 3.11 News and Stories Activity Diagram

3.5.6 Camera Page Activity Diagram

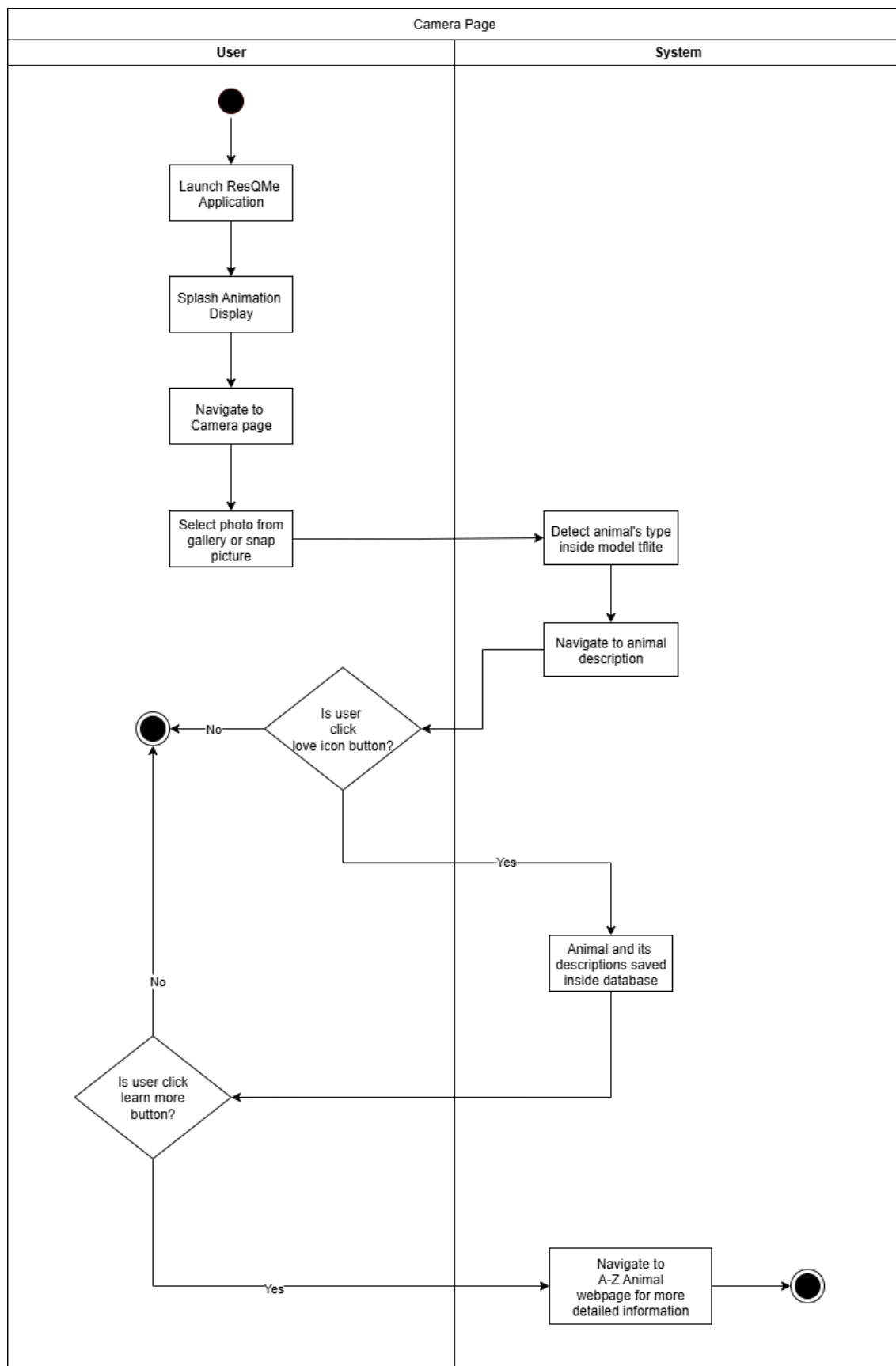


Figure 3.12 Camera Page Activity Diagram

3.5.7 Location Page Activity Diagram

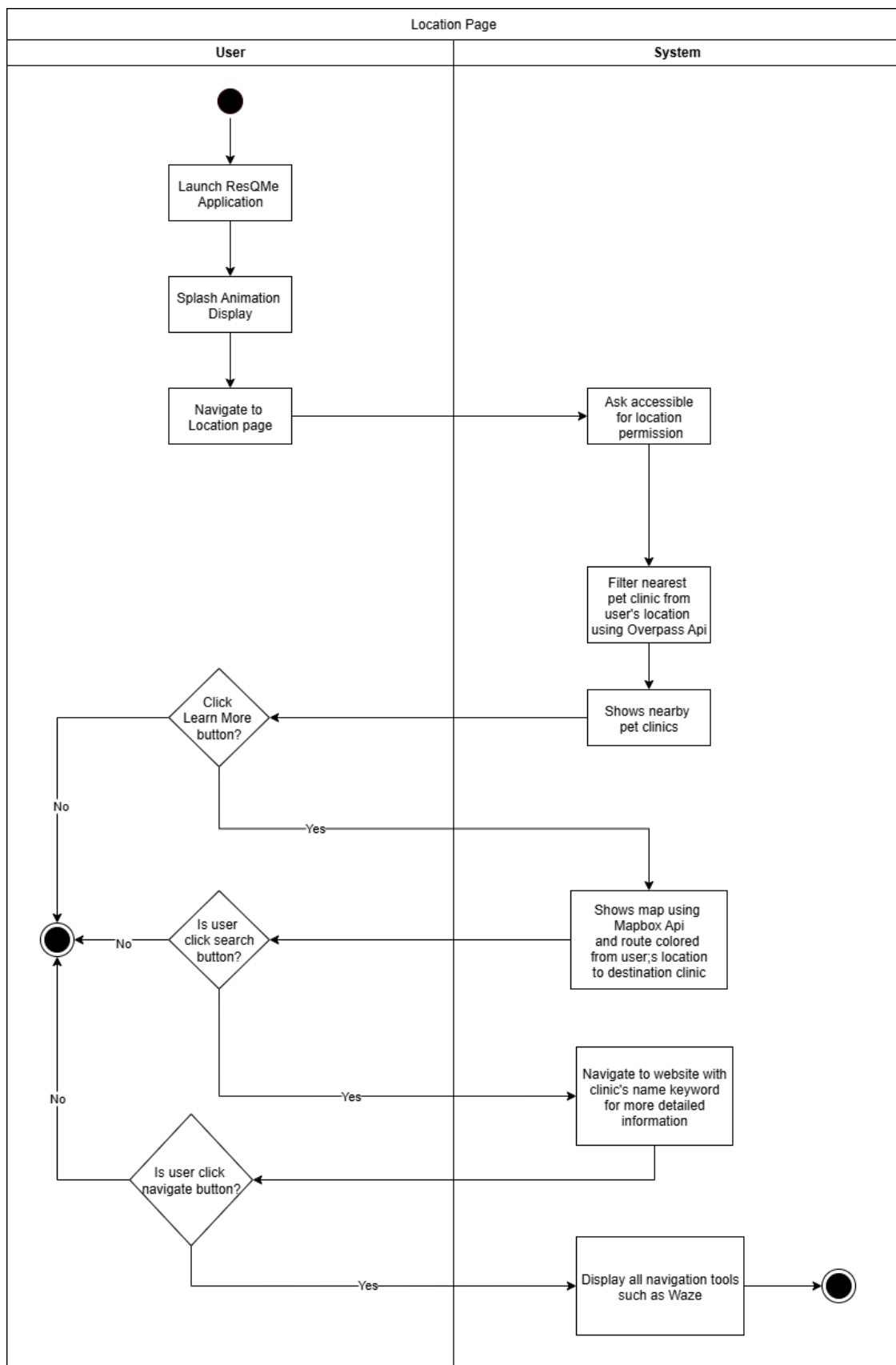


Figure 3.13 Location Page Activity Diagram

3.5.8 Saved Animal Page Activity Diagram

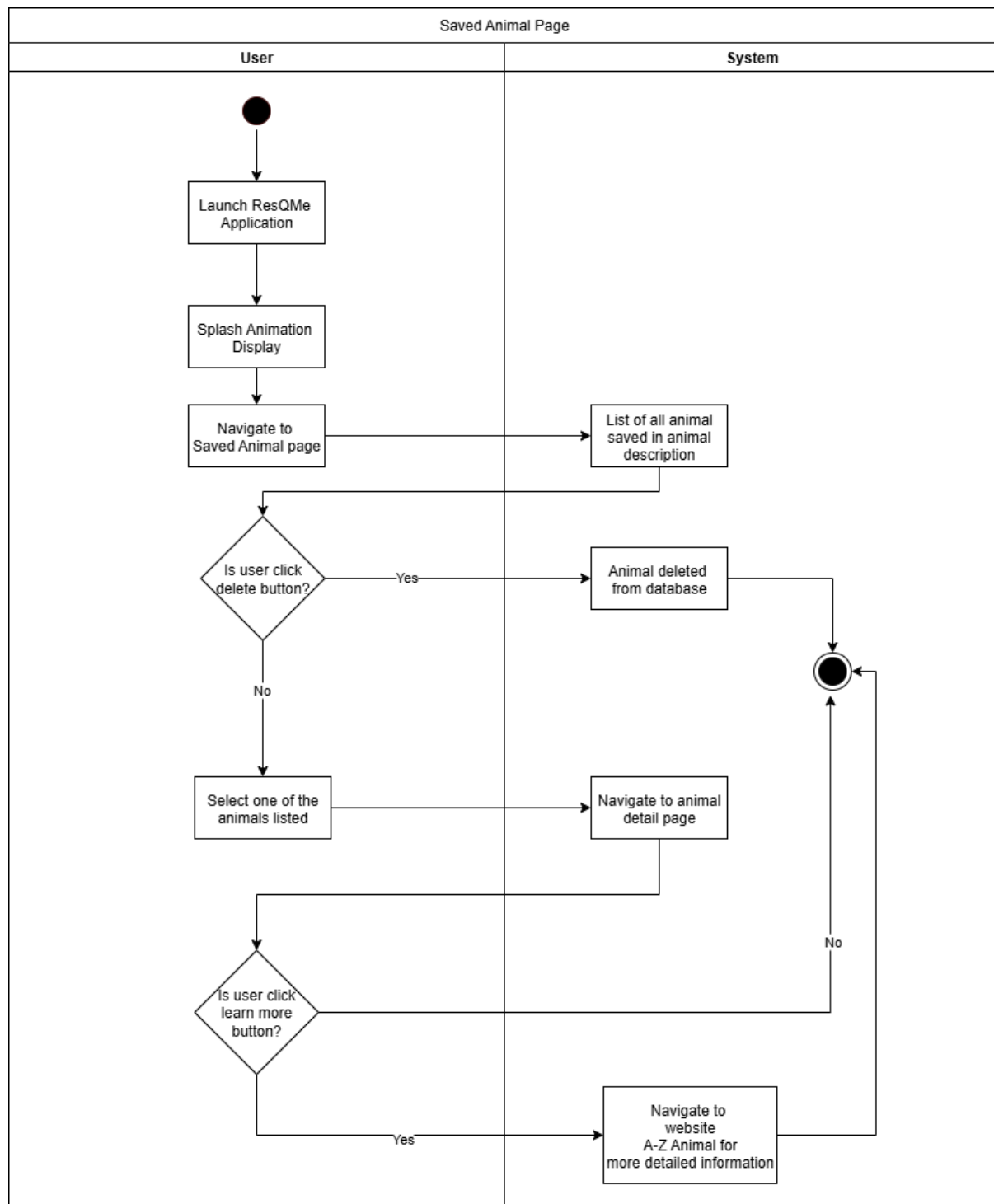


Figure 3.14 Saved Animal Page Activity Diagram

Chapter 4

System Design

4.1 System Block Diagram

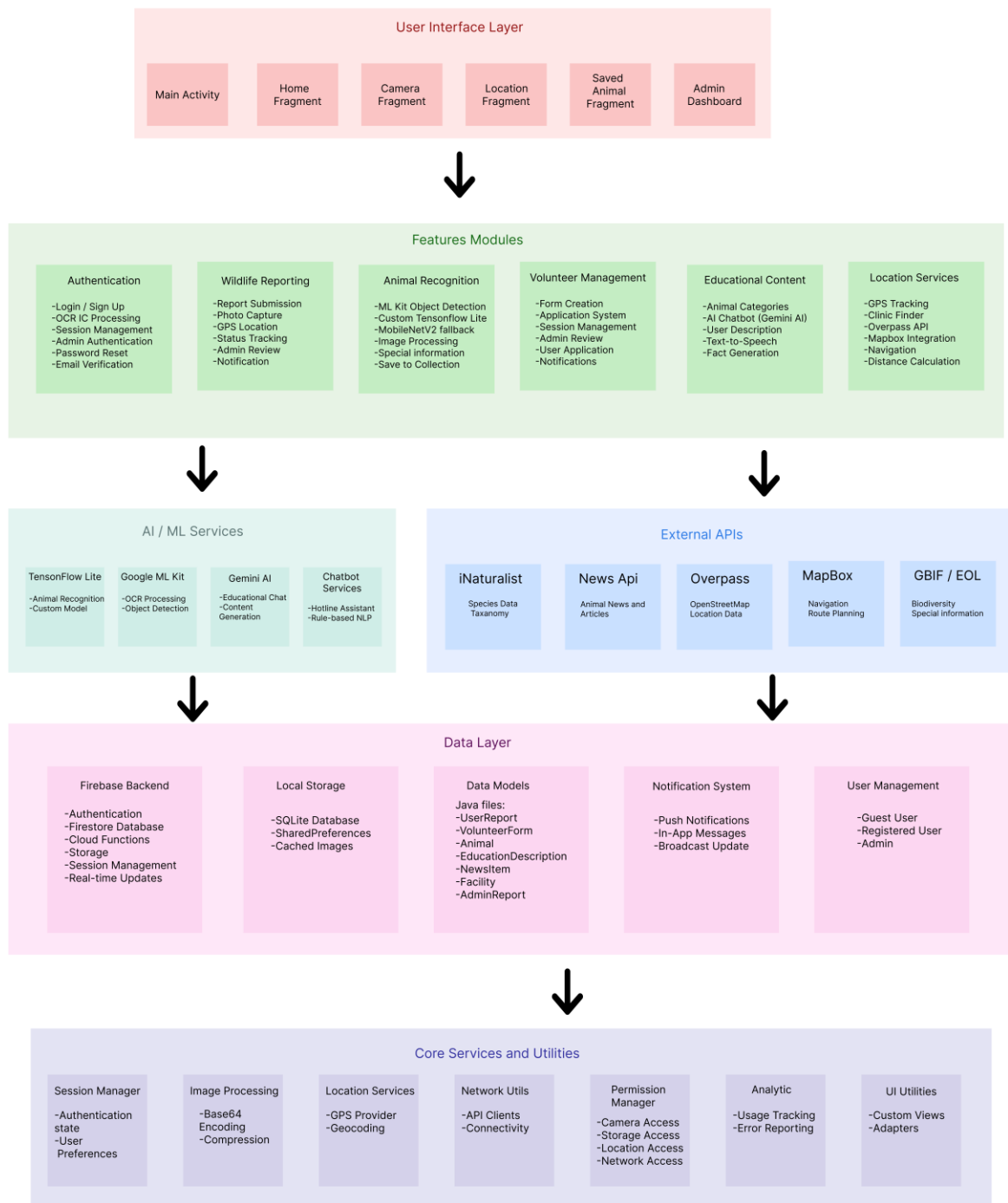


Figure 4.1 System Block Diagram

CHAPTER 4

Based on Figure 4.1, the system block diagram in ResQMe contains key layers such as the User Interface Layer, Feature Modules, AI and ML Services, External APIs, Data Layer, and Core Services.

User Interface Layer, including the Main Activity Java file, acts as the main navigational. There are 5 fragments, which are Home Fragment, Camera Fragment, Location Fragment, Saved Animal Fragment, and Admin Dashboard Fragment.

The **feature modules** include authentication, which makes sure users are always in secure mode, wildlife reporting for emergency cases, animal recognition for extra knowledge, volunteer management, which handles volunteer activities, educational content, and location services for real-time tracking.

The **AI Services and External APIs** make ResQMe content diversify, clean, and smoothly fetch the real-time data. Chatbot services overcome users' difficulties and problems instead of requiring extra knowledge. With extra APIs from the Internet, the system can unlock advanced features behind the scenes.

Firebase is the external storage and database inside the **data layer** for ResQMe animal rescue system, with faster speed, especially in online mode. Local storage, such as SQLite and SharedPreferences lets the system continue running even though it is in offline mode.

4.2 Storyboard

4.2.1 User Login and Registration Storyboard

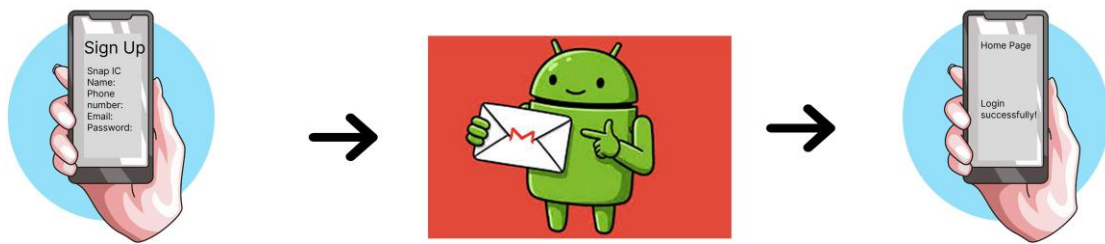


Figure 4.2a Sign Up Storyboard

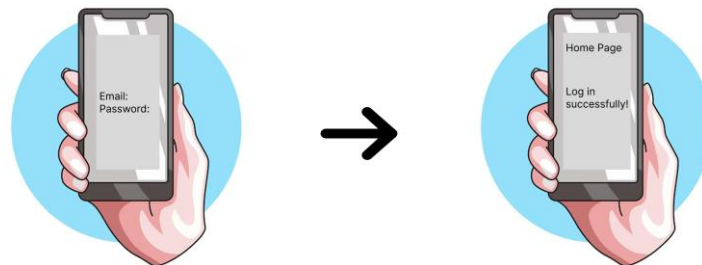


Figure 4.2b Login Storyboard

New guest user signs up with a real email, after completing email verification, return back to ResQMe app and logs in with the new email and password.

4.2.2 Report Submission Storyboard

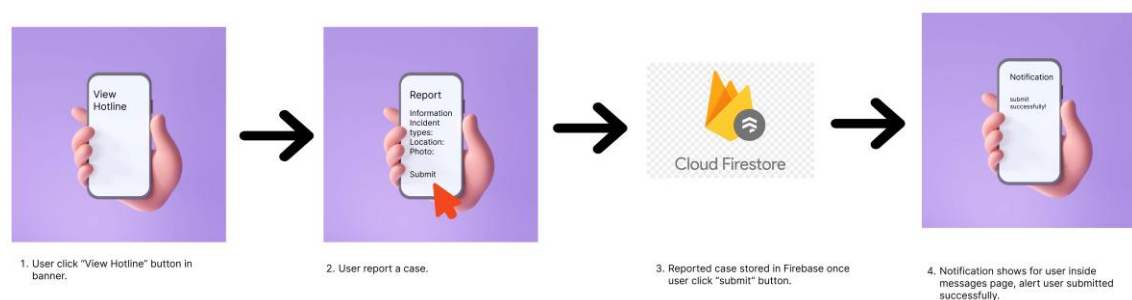


Figure 4.3 Report Submission Storyboard

4.2.3 Admin Approval Storyboard

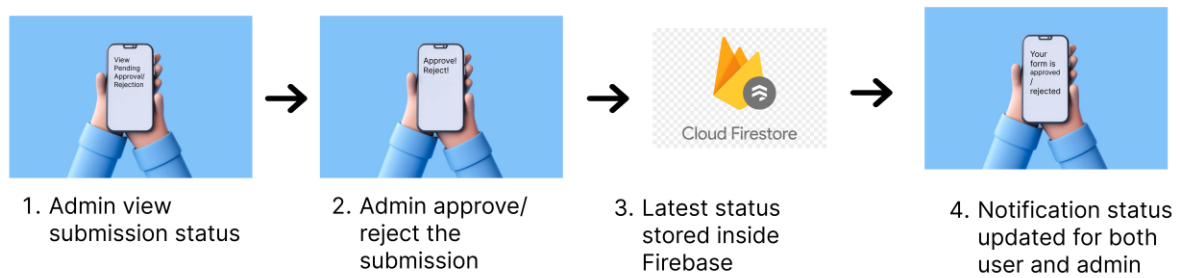


Figure 4.4 Admin Approval Storyboard

4.2.4 Camera Page Storyboard



Figure 4.5 Camera Page Storyboard

4.2.5 Educational Section Storyboard

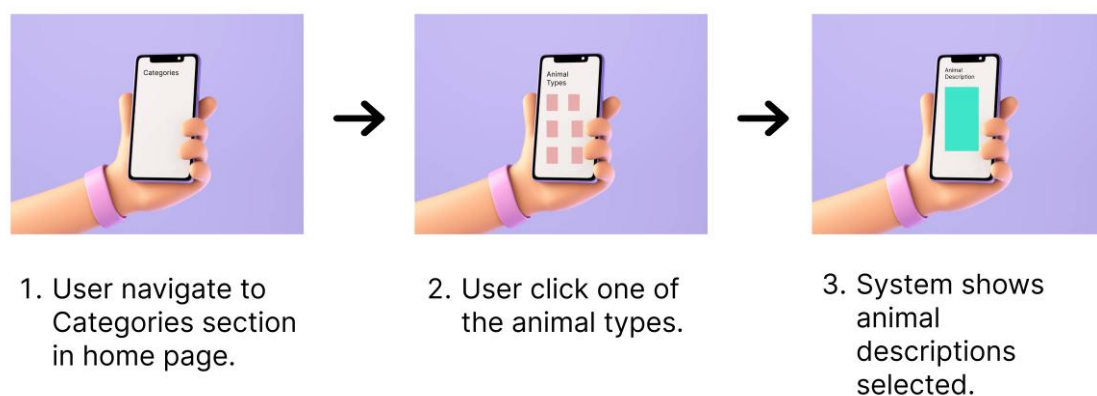


Figure 4.6 Educational Section Storyboard

4.2.6 Description Submission Storyboard

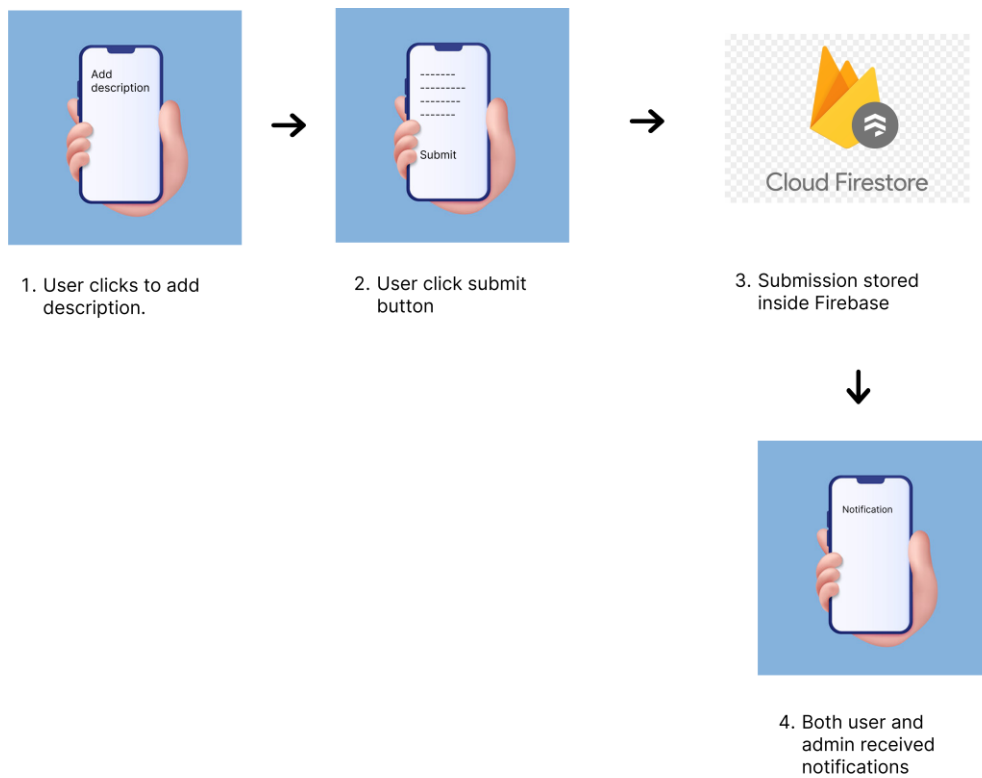


Figure 4.7 Description Submission Storyboard

4.2.7 Location Page Storyboard

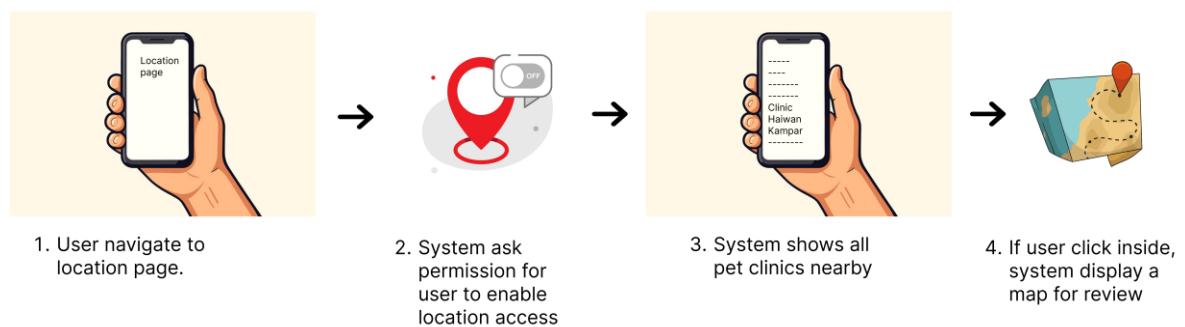


Figure 4.8 Location Page Storyboard

4.3 Flowchart

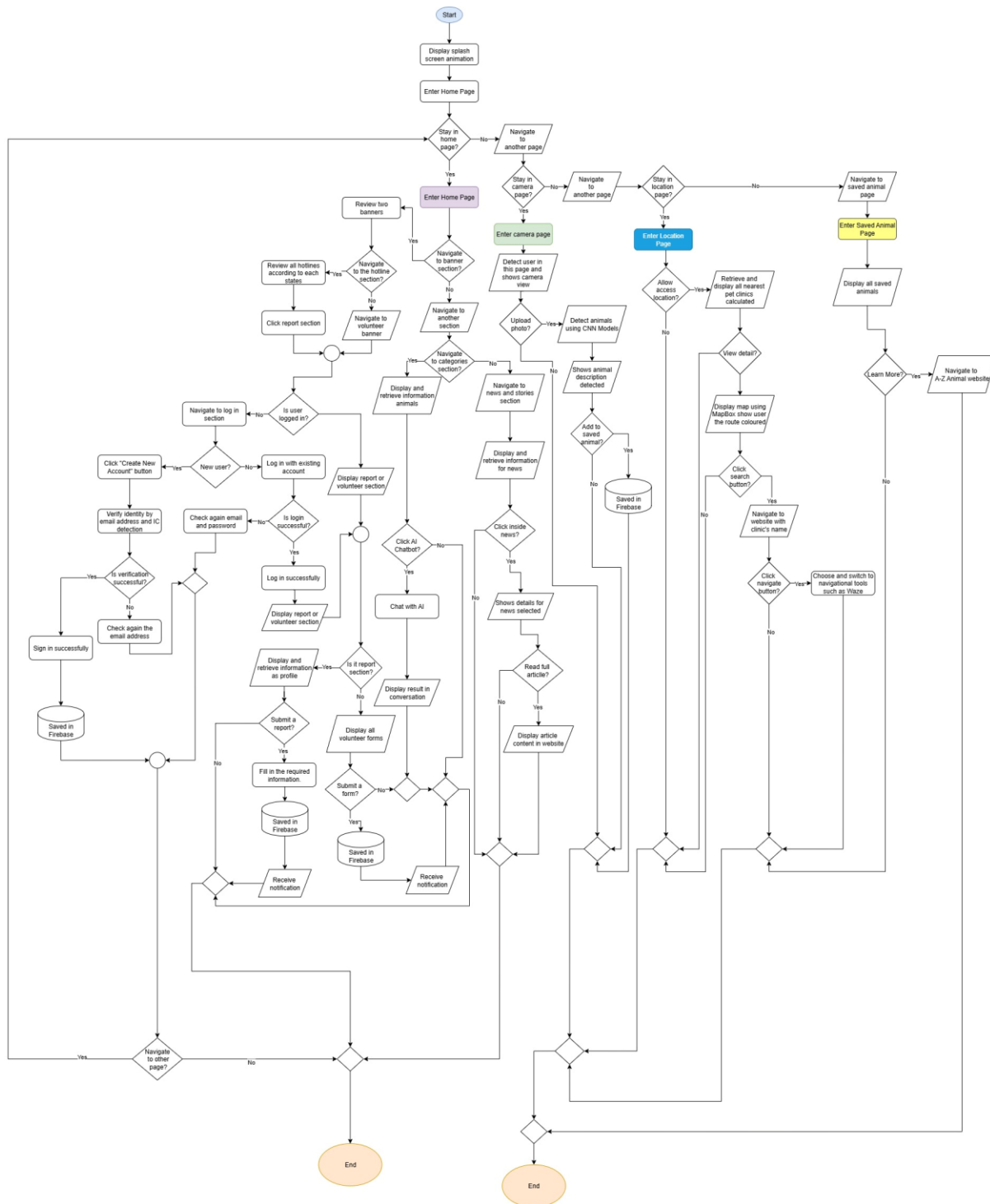


Figure 4.9 Overall Flowchart

4.4 System Components Specification

4.4.1 Registration and Authentication Module

This module enables both users and administrators to register or log in using their unique accounts safely. While users register for a new account, the system needs them to provide their IC photo and be detected by OCR, username, phone number, email, and password. After completing email verification to check if it's a real email, the user can return to ResQMe to log in with a new account. This step ensures authenticity and prevents duplicate accounts.

ResQMe supported role-based authentication, which included admin, guest user, and registered user. For admin login and sign up, additional information, which is the admin access code is required for the authorized admin's identity before being stored to Firebase, since admins have many powers to perform certain actions.

4.4.2 Wildlife Reporting Module

The wildlife reporting modules provide core features in rescue operations. It includes an inside hotline section, which, when user meet emergency situations, by inserting their basic information, the report will be submitted for approval after they click the "Submit" button. The photo captured as evidence is supported through camera view or gallery by storing inside Base64-encoded strings in Firebase. Image compression algorithm reduces file size while maintaining the photo's quality to decrease the delay operations for the system.

Admin acts as the approval character after the user submits the forms to ensure that the content is clear and reliable. Once actions are done, notifications will show in the messages section for both users and admins.

4.4.3 Animal Recognition and AI Module

A dual-AI model system used to ensure accuracy for animal detection, the primary model is a custom-trained TensorFlow Lite model called "animal_classification_model.tflite" by using its own combined datasets, and a pre-trained MobileNetV2 architecture.

Image processing with resizing to 224x224 pixels with RGB normalization, accompanied by object detection using Google ML Kit [13] for automatically cropping animals, purposefully detecting with higher probability and accuracy.

Besides, the AI system extends from Gemini AI applied in educational resources in the categories section to help users gain more understanding about that animal. The chatbot provides basic instructions to teach users learning how to use it, with species-by-species information provided, including habitats and facts. Template-based generated responses applied, such as a welcome message to enhance user experience.

4.4.4 Volunteer Management Module

Volunteer management system acts similarly to report submission, users apply the application form, which is announced by the admin, and the basic information, such as profile, will be sent back to the admin for reviewing without the need to fill in information again, enhancing convenience. By reviewing the difficulty levels such as “Beginner”, “Intermediate”, and “Experienced”, user can find their preferred volunteer activities rapidly. The application form supports rich information for users, such as activity descriptions, duration, location, and responsibilities. Once the application process is completed, real-time notifications allow interaction and communications between users and admins to enhance effectiveness.

4.4.5 Location Service Module

The location service modules provide comprehensive pet clinics nearby for better animal rescue operations. ResQMe queries OpenStreetMap through Overpass API [14] to quickly search the nearest rescue clinics within the user’s specified range, such as 10km, 25km, 50km, 100km, and 200km.

Besides, ResQMe also applied advanced distance calculations by using Mapbox Directions API to determine actual driving route situations, and the Haversine formula calculations calculate the straight-line distance. Rate-limited enhancement by using Nominatim reverse geocoding [15] made the whole process reliable. Moreover, integration with additional mapping using MapBox SDK [16] lets users have a better understanding of the situation of the route. ResQMe supported multiple navigation applications, such as Waze [17] and Google Maps [18], as well as users’ applications that have navigational functions.

4.4.6 Educational Content Module

Educational Content Management in ResQMe includes six different resource APIs to enrich the content, such as iNaturalist [19], GBIF [20], Open Tree of Life [21], and other scientific databases. Users can add their own additional descriptions by clicking the “Add Description” button. The descriptions submitted will be uploaded to the Description section after they add reliable content that has been approved by the admin. Besides, users can provide evidence of a photo to make other users have a better understanding of that kind of animal, since approved descriptions will be uploaded in real-time once the admin submits them for all users to have a look.

4.4.7 News and Stories Module

Animal-related news and stories by using NewsAPI.org and TheNewsAPI.com will show inside the News and Stories section on the page. Multiple APIs will refresh the news every day to provide users with information on events that have happened all the time. Once the user clicks the “View More” button, the system will only show 30 days of news to reduce the overload of information and lag.

4.5 Firebase Design

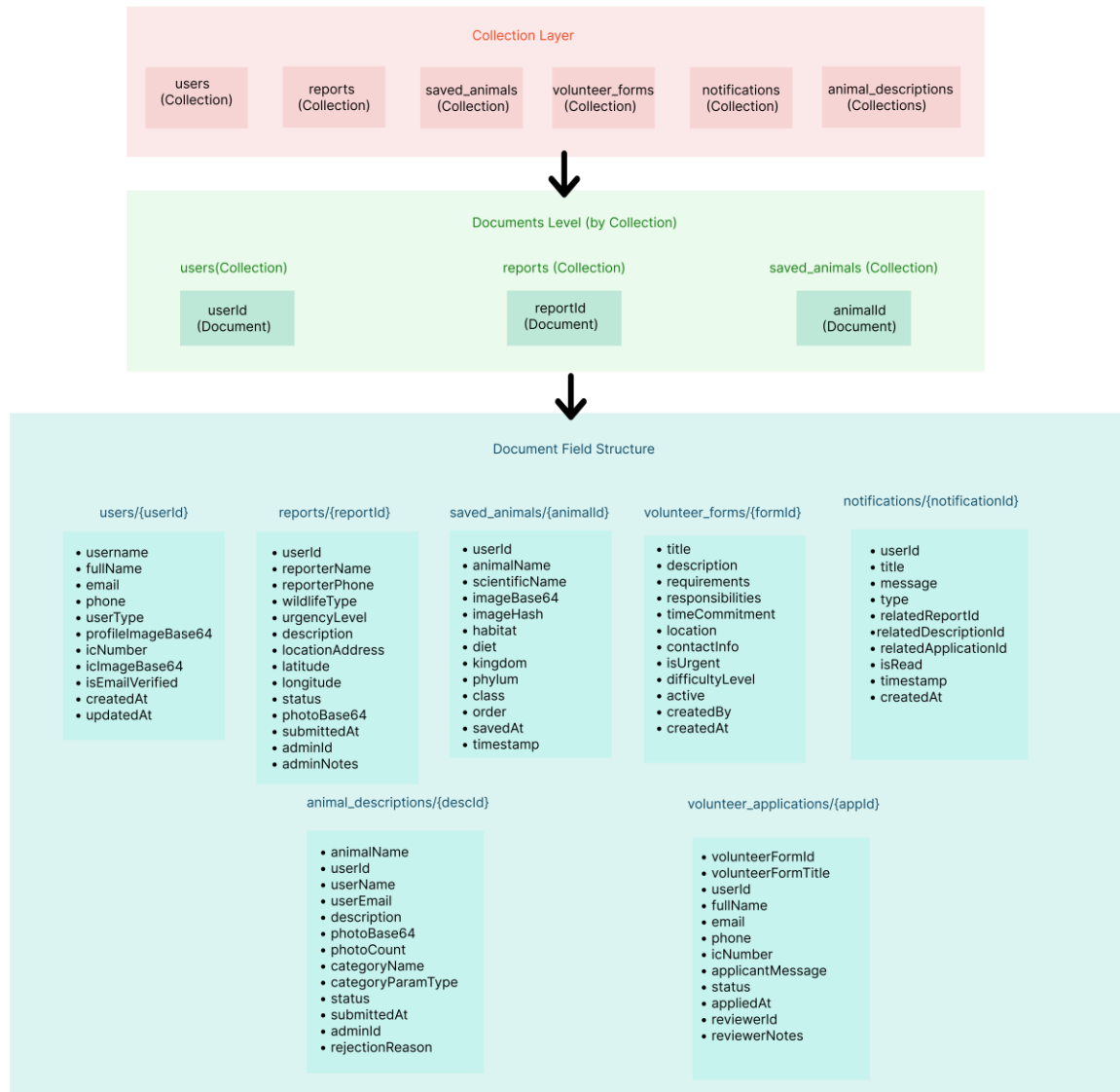


Figure 4.10 Firebase Design

Figure 4.4 shows the database architecture design. Firebase is the first choice not only for ease of use, but also for rapid real-time updates. The three layers include a collection layer, which contains 6 main collections to handle different aspects.

The users refer to users' profile and authentication, reports refer to wildlife reports where user submitted, saved_animal retrieve when user saved favorite animals from animal description page, volunteer_forms made by admins, which fill in forms for user submitting, notifications appear to both admin and user when action done, and animal_descriptions inside animal educational resources, which user can submit their description for approval.

CHAPTER 4

Here are the Firebase rules. These rules always streamline the data process of users' authentication, make sure everything is in secure mode:

```
rules_version = '2';

// Firestore Rules
service cloud.firestore {
  match /databases/{database}/documents {
    // Users can read/write their own document + allow querying for admins
    match /users/{userId} {
      // Users can read/write their own document
      allow read, write: if request.auth != null && request.auth.uid == userId;

      // Admins can read all user documents
      allow read: if request.auth != null &&
        getUserType(request.auth.uid) == 'admin';

      // Allow authenticated users to query users collection (needed for finding admins)
      allow read: if request.auth != null;

      // Allow email verification status updates
      allow update: if request.auth != null &&
        request.auth.uid == userId &&
        request.resource.data.diff(resource.data).affectedKeys()
          .hasOnly(['isEmailVerified', 'emailVerifiedAt']);

      // Allow admin profile updates (for IC verification, profile image, etc.)
      allow update: if request.auth != null &&
        request.auth.uid == userId &&
        getUserType(request.auth.uid) == 'admin';
    }

    // SAVED ANIMALS - OPTIMIZED FOR BASE64 STORAGE
    match /saved_animals/{animalId} {
```

```

// Users can read/write their own saved animals
allow read, write: if request.auth != null && request.auth.uid == resource.data.userId;

// Allow creating saved animals (make sure userId matches authenticated user)
allow create: if request.auth != null &&
    request.auth.uid == request.resource.data.userId;

// Allow deleting saved animals by owner
allow delete: if request.auth != null && request.auth.uid == resource.data.userId;

// Admins can read all saved animals
allow read: if request.auth != null &&
    getUserType(request.auth.uid) == 'admin';

// Allow querying saved animals by userId for current user
allow list: if request.auth != null &&
    request.auth.uid == resource.data.userId;

// IMPORTANT: Allow querying by animalName and imageHash for duplicate
detection
allow list: if request.auth != null &&
    request.auth.uid == resource.data.userId;
}

// Admin configuration
match /admin_config/{configId} {
    // Allow reading access_codes for login/signup validation (unauthenticated access)
    allow read: if configId == 'access_codes';
    // Allow authenticated admins to read other config
    allow read: if request.auth != null &&
        getUserType(request.auth.uid) == 'admin';
    // Only super admins can write
    allow write: if request.auth != null &&

```

```

        getUserType(request.auth.uid) == 'super_admin';
    }

    // Reports (Wildlife Reports)
    match /reports/{reportId} {
        // Users can read their own reports, admins can read all reports
        allow read: if request.auth != null && (
            request.auth.uid == resource.data.userId ||
            getUserType(request.auth.uid) == 'admin'
        );
        // Users can create reports (make sure userId matches authenticated user)
        allow create: if request.auth != null &&
            request.auth.uid == request.resource.data.userId;
        // Only admins can update reports (for approval/rejection)
        allow update: if request.auth != null &&
            getUserType(request.auth.uid) == 'admin';
        // Users can also update their own reports (for adding photoUrl after upload)
        allow update: if request.auth != null &&
            request.auth.uid == resource.data.userId;
        // Only admins can delete reports
        allow delete: if request.auth != null &&
            getUserType(request.auth.uid) == 'admin';
    }

    // Animal Descriptions - FIXED FOR BETTER ACCESS
    match /animal_descriptions/{descriptionId} {
        // Allow reading:
        // - Admins can read ALL descriptions (any status)
        // - Users can read their own descriptions (any status)
        // - ANY AUTHENTICATED USER can read approved descriptions (for public
display)
        // - Allow querying by animalId and userId for approved descriptions
        allow read: if request.auth != null && (

```

```

    getUserType(request.auth.uid) == 'admin' ||
    request.auth.uid == resource.data.userId ||
    resource.data.status == 'approved'
);

// Allow querying approved descriptions by any authenticated user
allow list: if request.auth != null &&
    request.query.where('status', '==', 'approved');

// Users can create descriptions (make sure userId matches authenticated user)
allow create: if request.auth != null &&
    request.auth.uid == request.resource.data.userId;

// Users can update their own descriptions (only if still pending)
// Admins can update any description (for approval/rejection)
allow update: if request.auth != null && (
    (request.auth.uid == resource.data.userId && resource.data.status == 'pending') ||
    getUserType(request.auth.uid) == 'admin'
);

// Only admins can delete descriptions
allow delete: if request.auth != null &&
    getUserType(request.auth.uid) == 'admin';
}

// VOLUNTEER FORMS - NEW COLLECTION
match /volunteer_forms/{formId} {
    // Any authenticated user can read active volunteer forms (for public display)
    allow read: if request.auth != null && resource.data.active == true;

    // Admins can read all volunteer forms (active and inactive)
    allow read: if request.auth != null &&
        getUserType(request.auth.uid) == 'admin';
}

```

```

// Allow querying active volunteer forms by any authenticated user
allow list: if request.auth != null &&
    request.query.where('active', '==', true);

// Allow querying all volunteer forms by admins
allow list: if request.auth != null &&
    getUserType(request.auth.uid) == 'admin';

// Allow querying by title for form resolution (needed for applications)
allow list: if request.auth != null;

// Only admins can create volunteer forms
allow create: if request.auth != null &&
    getUserType(request.auth.uid) == 'admin' &&
    request.resource.data.createdBy == request.auth.uid;

// Only admins can update volunteer forms
allow update: if request.auth != null &&
    getUserType(request.auth.uid) == 'admin';

// Only admins can delete volunteer forms
allow delete: if request.auth != null &&
    getUserType(request.auth.uid) == 'admin';
}

// VOLUNTEER APPLICATIONS - NEW COLLECTION
match /volunteer_applications/{applicationId} {
    // Users can read their own applications
    allow read: if request.auth != null &&
        request.auth.uid == resource.data.userId;

    // Admins can read all applications

```

```

allow read: if request.auth != null &&
    getUserType(request.auth.uid) == 'admin';

// Allow querying applications by userId for current user
allow list: if request.auth != null &&
    request.auth.uid == resource.data.userId;

// Allow querying applications by volunteer form ID for admins
allow list: if request.auth != null &&
    getUserType(request.auth.uid) == 'admin' &&
    'volunteerFormId' in request.query.where;

// Allow querying by userId and volunteerFormId for duplicate checking
allow list: if request.auth != null &&
    request.auth.uid == resource.data.userId &&
    'volunteerFormId' in request.query.where;

// Allow querying by userId and volunteerFormTitle for duplicate checking (fallback)
allow list: if request.auth != null &&
    request.auth.uid == resource.data.userId &&
    'volunteerFormTitle' in request.query.where;

// Allow admins to query all applications without restrictions
allow list: if request.auth != null &&
    getUserType(request.auth.uid) == 'admin';

// Users can create applications (make sure userId matches authenticated user)
allow create: if request.auth != null &&
    request.auth.uid == request.resource.data.userId;

// Only admins can update applications (for approval/rejection)
allow update: if request.auth != null &&
    getUserType(request.auth.uid) == 'admin';

```



```

// Users can update their own applications (for status tracking)
allow update: if request.auth != null &&
    request.auth.uid == resource.data.userId;

// Only admins can delete applications
allow delete: if request.auth != null &&
    getUserType(request.auth.uid) == 'admin';
}

// Notifications - FIXED TO ALLOW SYSTEM CREATION
match /notifications/{notificationId} {
    // Users can read their own notifications, admins can read all
    allow read: if request.auth != null && (
        request.auth.uid == resource.data.userId ||
        getUserType(request.auth.uid) == 'admin'
    );

    // Allow querying notifications by userId
    allow list: if request.auth != null &&
        request.auth.uid == resource.data.userId;

    // Allow admins to query all notifications
    allow list: if request.auth != null &&
        getUserType(request.auth.uid) == 'admin';

    // Allow creating notifications
    // - Any authenticated user can create notifications (for system notifications)
    // - This is needed for NotificationHelper.createAdminNewDescriptionNotification()
    allow create: if request.auth != null;

    // Users can update their own notifications (mark as read)
    // Admins can update any notification

```

```

allow update: if request.auth != null && (
    request.auth.uid == resource.data.userId ||
    getUserType(request.auth.uid) == 'admin'
);

// Only admins can delete notifications
allow delete: if request.auth != null &&
    getUserType(request.auth.uid) == 'admin';
}

// Reported cases
match /reported_cases/{caseId} {
    allow read: if request.auth != null;
    allow create: if request.auth != null &&
        request.auth.uid == resource.data.userId;
    allow update: if request.auth != null && (
        // User can update their own case
        (request.auth.uid == resource.data.userId) ||
        // Admin can approve/reject
        (getUserType(request.auth.uid) == 'admin')
    );
}

// Messages
match /messages/{messageId} {
    allow read: if request.auth != null && (
        request.auth.uid == resource.data.senderId ||
        request.auth.uid == resource.data.recipientId
    );
    allow create: if request.auth != null;
    allow update: if request.auth != null &&
        request.auth.uid == resource.data.recipientId;
}

```

```

// User activities (for tracking user actions - optional)
match /user_activities/{activityId} {
  allow read, write: if request.auth != null;
}

// Email verification tracking (optional)
match /email_verification_logs/{logId} {
  allow read, write: if request.auth != null &&
    request.auth.uid == resource.data.userId;
  allow read: if request.auth != null &&
    getUserType(request.auth.uid) == 'admin';
}

// Helper function to get user type
function getUserType(userId) {
  return get(/databases/$(database)/documents/users/$(userId)).data.userType;
}
}

// Firebase Storage Rules - MINIMAL FOR FREE TIER (Base64 reduces storage usage)
service firebase.storage {
  match /b/{bucket}/o {
    // Only essential storage for non-animal images (profile pics, reports, etc.)
    match /{allPaths=**} {
      // Allow read access to authenticated users
      allow read: if request.auth != null;

      // Allow write access to authenticated users with strict size limit
      // Since animal images are now base64, this is mainly for other app assets
      allow write: if request.auth != null &&
        request.resource.size < 2 * 1024 * 1024 && // 2MB max
    }
  }
}

```

```

        request.resource.contentType.matches('image/.*'); // Only images
    }

    // Separate rule for IC images (if stored in storage instead of base64)
    match /ic_images/{userId}/{fileName} {
        allow read, write: if request.auth != null &&
            request.auth.uid == userId &&
            request.resource.size < 5 * 1024 * 1024; // 5MB for IC images
    }

    // Profile images
    match /profile_images/{userId}/{fileName} {
        allow read: if request.auth != null;
        allow write: if request.auth != null &&
            request.auth.uid == userId &&
            request.resource.size < 3 * 1024 * 1024; // 3MB for profile images
    }
}
}
}

```

These Firebase rules establish a role-based authentication system, where users can upload their own data, such as profile settings and submissions. Admins have broad access across the collections, including the ability to make approvals. The main features include the submission of a wildlife report in the hotline section, verification of email addresses, participation in the application process, and an animal description with an approval workflow, as well as a notification system. Implementation of data validation ensures user authorization, which user can only access data using their own account, improve the security and belief in ResQMe. For Firebase Storage, ResQMe optimized for minimal usage instead of “Get Started” in Firebase storage, these storage data such as profile images and IC verification documents, are used.

4.6 Prototype Design

Wireframe Prototype Design

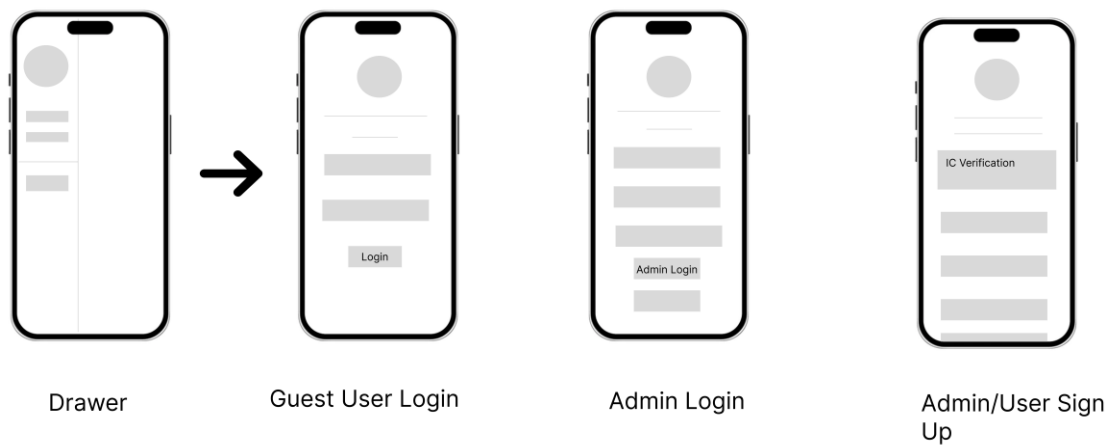


Figure 4.11 Login and Sign-Up Prototype

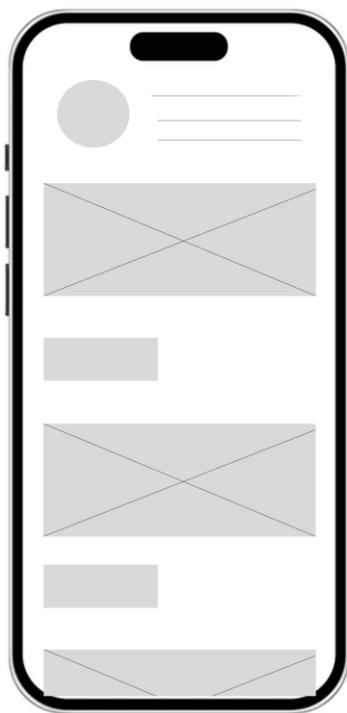


Figure 4.12 Home Page Prototype

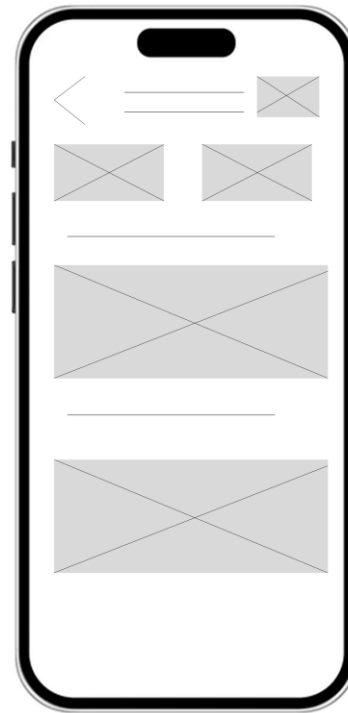


Figure 4.13 Hotline Prototype

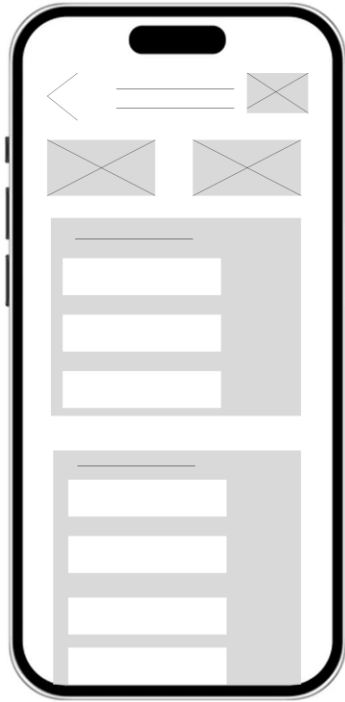


Figure 4.14 Report Prototype

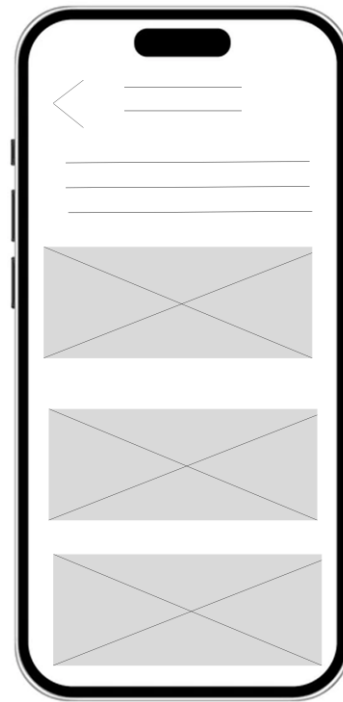


Figure 4.15 Volunteer Prototype

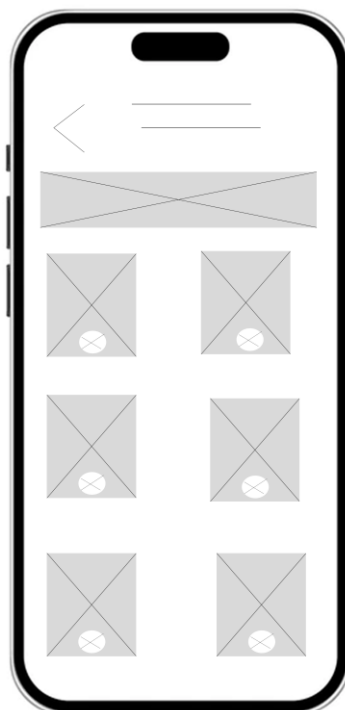


Figure 4.16 Categories Prototype

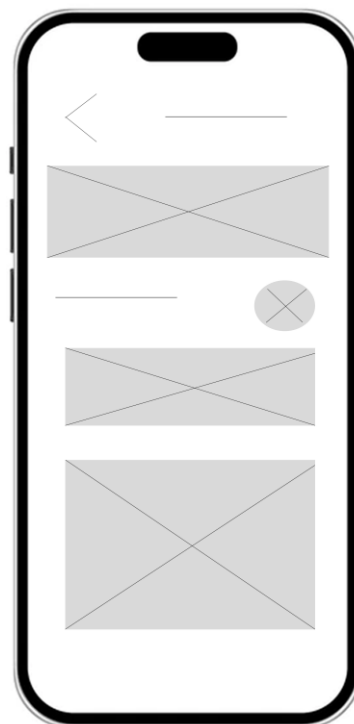


Figure 4.17 Animal Details Prototype

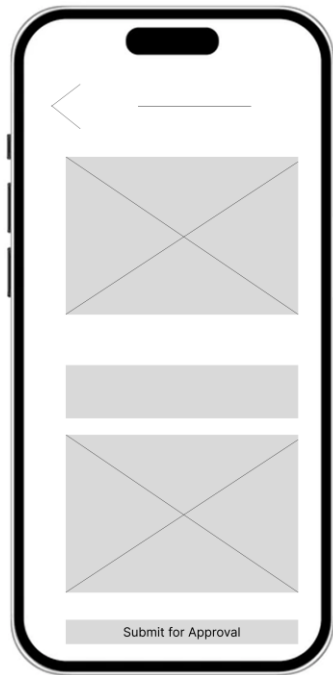


Figure 4.18 Add Description Prototype

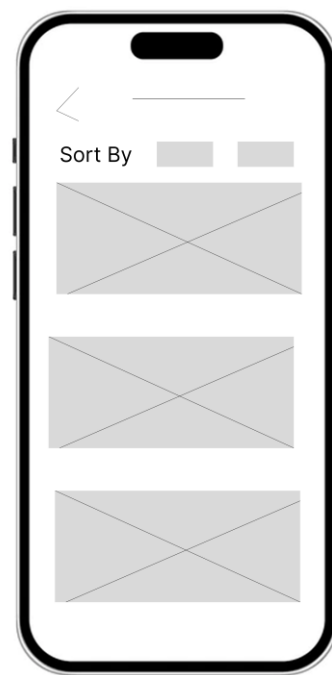


Figure 4.19 News and Stories Prototype

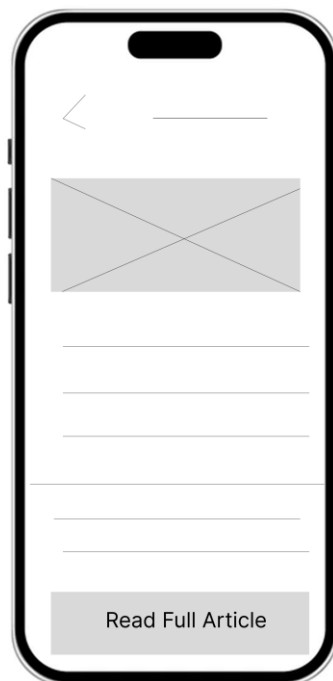


Figure 4.20 News Article Prototype

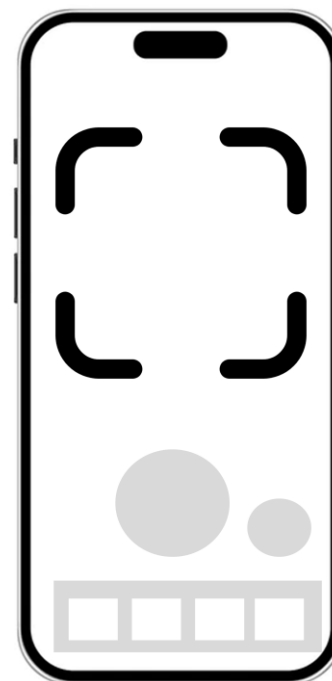


Figure 4.21 Camera Page Prototype

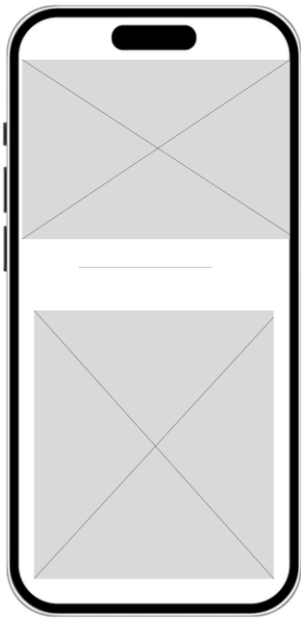


Figure 4.22 Animal Description Prototype

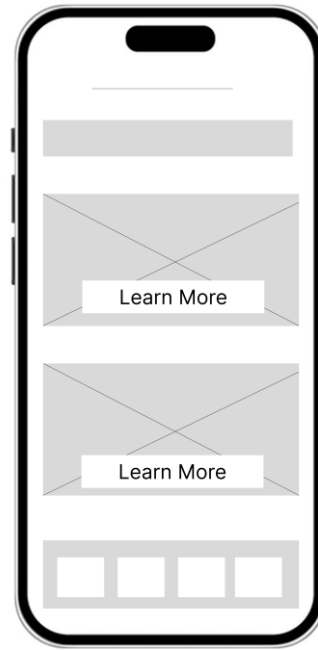


Figure 4.23 Location Page Prototype



Figure 4.24 Clinic Detail Prototype

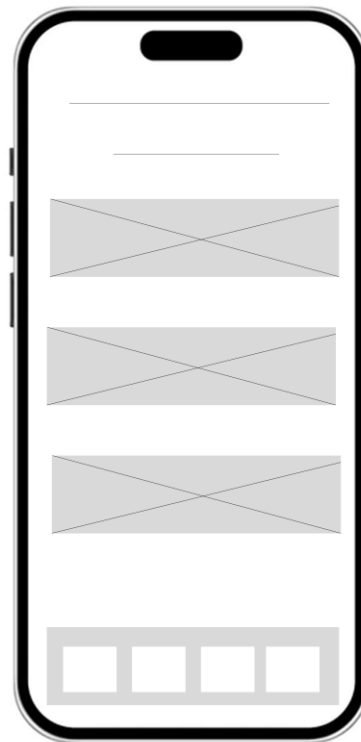


Figure 4.25 Saved Animal Page Prototype

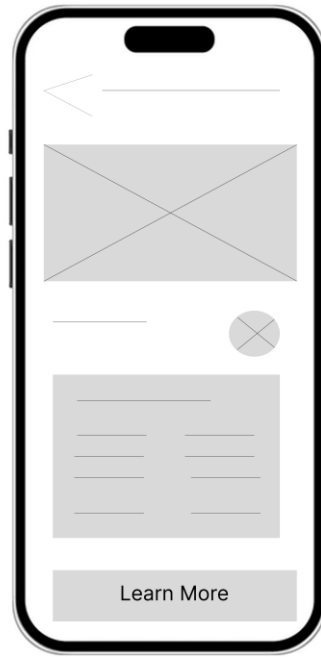


Figure 4.26 Saved Animal Description Prototype

CHAPTER 4

4.7 Timeline

4.7.1 Timeline for FYP1

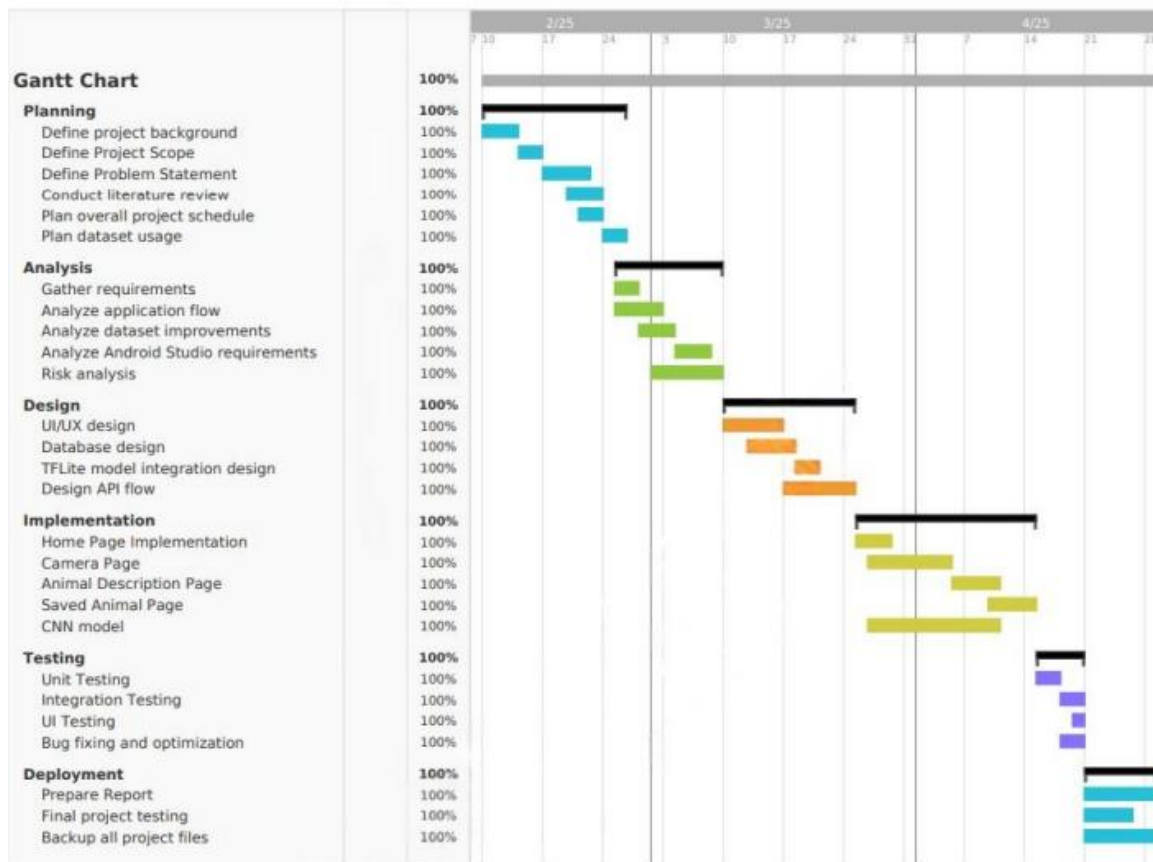


Figure 4.27a Gantt Chart for FYP1

4.7.2 Timeline for FYP2

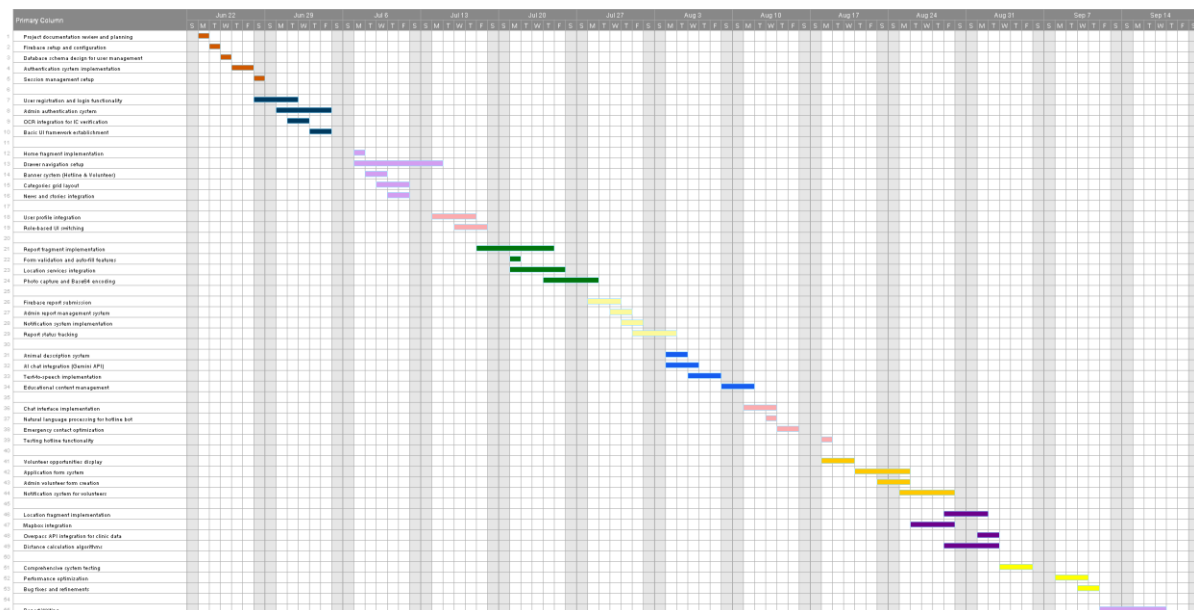


Figure 4.27b Gantt Chart for FYP2

Chapter 5

SYSTEM IMPLEMENTATION

5.1 Hardware Setup

The hardware components primarily include a computer system and mobile devices. The computer acts as a development and training machine learning model, implementing the CNN architecture, and optimizing the algorithm for route planning. Mobile devices will be used to deploy the application and provide end-users with access to the system features.

Table 5.1 Specifications of Laptop

Description	Specifications
Model	Asus Tuf Gaming A15 FA507NU
Processor	AMD Ryzen 5 7535HS with Radeon Graphics 3.30 GHz
Operating System	Windows 11
Graphic	NVIDIA GeForce RTX 4050 Laptop GPU
Memory	16GB RAM
Storage	449 GB NVMe SSD

Table 5.2 Specifications of Smartphone

Description	Specifications
Model	OPPO Reno12 Pro 5G
Processor	MediaTek Dimensity 7300-Energy (4 nm)
Operating System	Android 14, ColorOS 14.1
Memory (RAM)	12 GB
Storage	512 GB (UFS 3.1)
Camera (Front)	50 MP, f/2.0
Camera (Rear)	Triple: 50 MP (wide, OIS) + 50 MP (telephoto, 2× optical zoom) + 8 MP ultra-wide
GPU	Mali-G615 MC2
Max CPU Clock Speed	Up to 2.5 GHz (Cortex-A78 cores)

5.2 Software Setup

Table 5.2.1 shows the main software tools used; Table 5.2.2 represents the programming language, and Table 5.2.3 shows packages and libraries. Table 5.2.4 describes packages of java used in Android Studio. Table 5.2.5 shows extra tools from Internet resources.

Table 5.3 Main Software Tools




Description	Specifications
<p>Model Training Platform</p>  <p>Figure 5.1a Kaggle [22]</p>	<p>Kaggle</p> <ul style="list-style-type: none"> • Provides Various animal databases provided especially for deep learning tasks. • Pre-configured ML libraries and powerful GPU resources are provided for animal recognition.
<p>Mobile Development IDE</p>  <p>Figure 5.1b Android Studio [23]</p>	<p>Android Studio</p> <ul style="list-style-type: none"> • Environment for mobile application development. • Supported various dependencies and libraries. • Provides comprehensive coding, writing, and testing procedures. • Supported built-in emulator. • Integrated with Cloud Firestore.
<p>Backend Service</p>  <p>Figure 5.1c Firebase [8]</p>	<p>Firebase</p> <ul style="list-style-type: none"> • Easy to use and clean format. • No need for a SQL database for authentication and storing data. • For secure user authentication. • Provides session management activities.

Table 5.4 Programming Language





Description	Specifications
<p>Python</p>  <p>Figure 5.2a Jupyter Notebook [24]</p>	<p>Jupyter Notebook</p> <ul style="list-style-type: none"> • Tested step-by-step, reduce waiting time. • Primary language for deep learning and animal recognition. • Data pre-processing and augmentation. • Model conversation with TensonFlow Lite.
<p>Java</p>  <p>Figure 5.2b Java [25]</p>	<ul style="list-style-type: none"> • Cross-platform programming language. • Powerful language in debugging, coding, and testing. • Compatibility with various Android APIs.

Table 5.5 Libraries and Packages

Python Libraries	Version	Purpose
<p>TensonFlow</p>  <p>Figure 5.2c TensonFlow [26]</p>	v2.17.0	Deep learning platform for training neural network models. Provides tools for model conversation with Android mobile application development.
<p>Keras</p>  <p>Figure 5.2d Keras [27]</p>	v3.6.0	High-level neural network APIs for model development.





<p>OpenCV</p>  <p>Figure 5.2e OpenCV [28]</p>	v4.10.0	A library of computer vision for resizing, normalization, and preprocessing for animal recognition.
<p>NumPy</p>  <p>Figure 5.2f NumPy [29]</p>	v1.26.4	Supported multi-dimensional arrays and mathematical calculations.
ImageHash	v4.3.1	A hashing library for detecting and removing duplicate images to enhance effectiveness.


Table 5.6 Java Packages

Package Name	Purpose
com.google.firebase.auth	Firebase authentication, handle user and admin log in, sign-up, or submission forms.
com.google.firebase.firestore	Firestore, no need for SQL code for report submission, volunteer submission, description submission, and notifications.
com.google.firebase.Timestamp	Used for time tracking for reports, applications, and notifications.
androidx.appcompat.app	Support backward-compatible UI components for Android.
androidx.recyclerview.widget	RecyclerView+Adapter to display lists of reports, volunteer forms, and applications.
androidx.cardview.widget	CardView layout for UI.

androidx.fragment.app	Admin Dashboard fragment, Home Fragment, Camera Fragment, Location Fragment, and Saved Animal Fragment
androidx.core.app	Notifications and permissions, such as camera permission, location permission, and network permission, are provided by the utilities.
com.google.android.material	Material Component Design.
android.location	Access GPS and geocoding services in the report section.
com.google.android.gms.location	FusedLocationProviderClient for location service.
android.util.Base64	Storing images encoding/decoding for Firebase.
android.graphics	Display images in Base64.
com.airbnb.lottie	Lottie animation used to enhance user interest.
com.google.mlkit.vision.text	OCR, Google ML Kit for text recognition.
java.util	Collections, dates, and timestamp.

Table 5.7 Extra Tools from Network

Tool	Purpose	Usage in ResQMe
 Figure 5.3a Draw.io [30]	Sketching and designing tool for drawing diagrams.	Used to design case diagrams, activity diagrams, and flow charts.
 Figure 5.3b Figma [31]	Collaborative UI/UX design tool.	Used to design UI for all pages in ResQMe.

<p>Canva</p>  <p>Figure 5.3c Canva [32]</p>	Graphic design tool.	Used to design a poster.
--	----------------------	--------------------------

5.3 Setting and Configuration

5.3.1 Android Studio Setup

1. Download Android Studio [9] from the official website (<https://developer.android.com>).
2. Install and configure Android Studio.

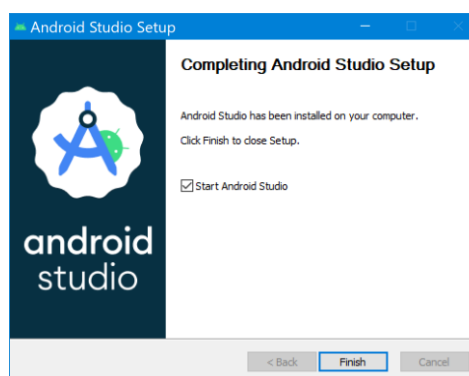


Figure 5.4a Android Studio Setup

3. Set up SDK with minimum API level 24 (Android 7.0).

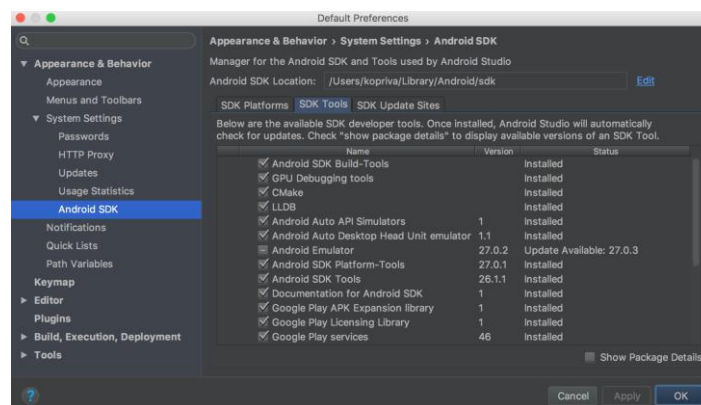


Figure 5.4b Setup Android SDK

4. Configure the emulator for device debugging.

5.3.2 Firebase Setup

1. Log in or sign in on the Firebase official website [8] (<https://console.firebase.google.com/>).

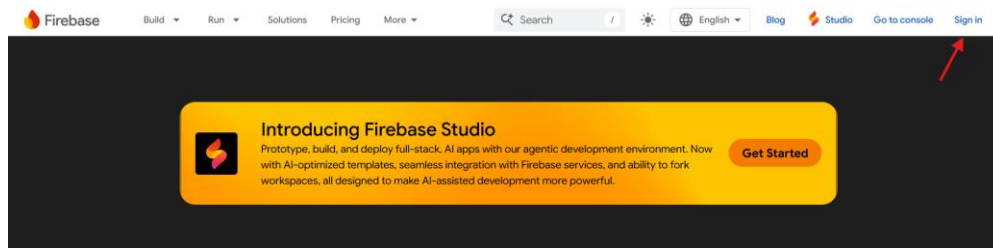


Figure 5.5a Sign in Firebase

2. Create a new project in Firebase after logging.
3. Add the ResQMe Android application to the Firebase project.
4. Download the google-services.json file and process integration.

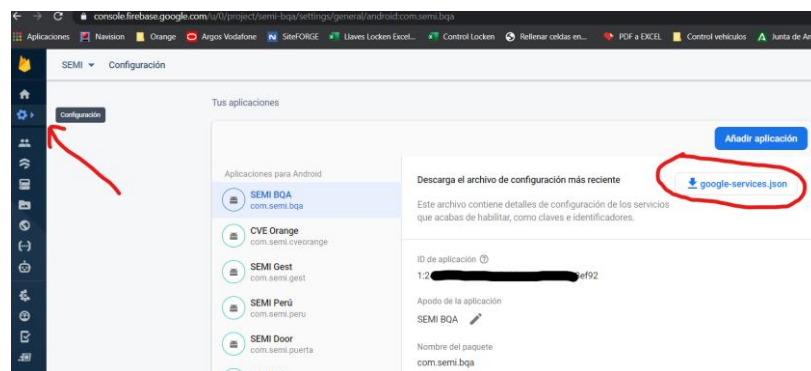


Figure 5.5b Download google-services.json

5. Enable authentication and Firestore Database.

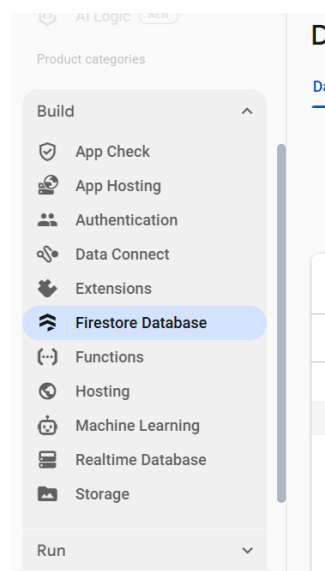


Figure 5.5c Enable components in build

5.3.3 TensorFlow Lite Setup

1. In build.gradle, add TensorFlow Lite [26] [20] dependencies.

- Integrate the custom animal classification model (animal_classification_model.tflite) inside the ml folder.

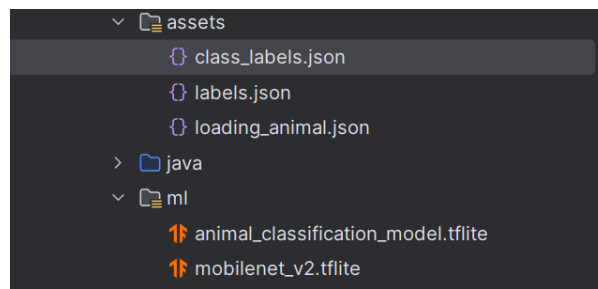


Figure 5.6 Integrate JSON files and TFLite files

- Add a fallback MobileNetV2 model for enhanced accuracy.
- Add a JSON file (class_labels.json and labels.json) that includes all custom and trained animal datasets.

5.3.4 Third-Party APIs Setup

Table 5.8 Third-Party APIs Setup

API Name	Purpose	Usage in Project
Animal Data APIs		
GBIF API [20]	Global Biodiversity information on animals	<ul style="list-style-type: none"> Fetch taxonomic information of animals according to different categories. Used in ApiAnimalDetailActivity.java for species identification.
API Ninjas [33]	Various animal information provided.	<ul style="list-style-type: none"> Get animal basic information and characteristics. Used in ApiAnimalDetailActivity.java to show animal information. Primary source in animal detail page.
The Dog API [34]	Breed-specific dog information.	<ul style="list-style-type: none"> Specific dog official name in animal description.
Cat API [35]	Breed-specific cat information.	<ul style="list-style-type: none"> Specific cat official name in animal description.
Encyclopedia of Life (EOL) API [36]	Comprehensive special animal types	<ul style="list-style-type: none"> Rich taxonomic and biodiversity animal information. Secondary source in animal detail page.

iNaturalist API [19]	Citizen scientific animal. official species	<ul style="list-style-type: none"> Additional biological information.
Open Tree of Life API [21]	Various tree databases.	<ul style="list-style-type: none"> Provides an evolutionary relationship with each animal. Provide additional information in taxonomic terms, such as habitat environment.
Animal API [37]	General animal information	<ul style="list-style-type: none"> Provides basic information for animal species.
Zoo Animal API [38]	Zoo Animal database provided.	<ul style="list-style-type: none"> Enhance the diversity of different animals. Familiar animal species provided.
Location APIs		
Overpass Api [14]	Data query for OpenStreetMap.	<ul style="list-style-type: none"> Find veterinary and pet clinics. Geographic coordinates used to query facilities.
Mapbox Api [11]	Provides map overview, location, and navigational tracking.	<ul style="list-style-type: none"> Provides real-time driving routes and location tracking. Used in MapActivity.java and LocationFragment.java for distance calculations.
Nominatim API [39]	OpenStreetMap geocoding service.	<ul style="list-style-type: none"> Convert location address to coordinates. LocationInputActivity.java when the user inputs an address manually.
News and Stories APIs		
NewsAPI.org [40]	Fetch animal-related articles.	<ul style="list-style-type: none"> Primary news source in NewsApiService.java.
TheNewsAPI.com [41]	Fetch animal-related articles.	<ul style="list-style-type: none"> Backup news source. Fallback if the primary news API loading failed.

AI and Machine Learning APIs		
Google Gemini AI [42]	A large language model is basic for conversation.	<ul style="list-style-type: none"> Acts as a chatbot in the educational section for extra knowledge. GeminiAIService.java and AIChatActivity.java for contextual animal discussions
Hugging Face API [43]	A machine learning model.	<ul style="list-style-type: none"> Used DialoGPT to generate animal facts. SimpleAIFactsGenerator.java for AI-generated animal information.
OpenAI GPT API [44]	Advanced language model.	<ul style="list-style-type: none"> Secondary AI service in SimpleAIFactsGenerator.java. Backup AI tools if the Hugging Face API loading failed.
Google ML Kit [13]	On-device machine learning.	<ul style="list-style-type: none"> Object detection and OCR text extraction. OCRHelper.java for IC scanning and CameraFragment.java for animal detection.

5.3.5 Algorithms Setup

Table 5.9 Algorithms Setup

Algorithm Name	Purpose	Setup
OCR Algorithm (Optical Character Recognition) [12]	Extracts text such as real name on IC.	<ul style="list-style-type: none"> Implements by using Google ML Kit Text Recognition API. Convert image to bitmap format, then identify and extracts text blocks, a regular expression (regex) pattern used to detect valid Malaysian IC name.
NLP Algorithm (Rule-Based Chatbot) [45]	Provides emergency hotlines in chatbot	<ul style="list-style-type: none"> Using keywords detection and string matching. Simple NLP preprocessing

	for quick rescue operations.	
Image Compression Algorithm	Compresses image size before uploading.	<ul style="list-style-type: none"> • Using Android Bitmap Compression. • Resize to 300 x 300 px. • JPEG quality reduced by 40%. • For storage, encoded to Base64.
Geocoding Algorithm [46]	Convert GPS coordinates to readable address in report section.	<ul style="list-style-type: none"> • Using Android's Geocoder API. • Coordinates are taken from FusedLocationProviderClient.
CNN (Convolutional Neural Networks) [1]	Animal detection and recognition.	<ul style="list-style-type: none"> • Collects combined animal datasets. • Reuse pre-trained MobileNetV2 architecture. • Preprocessing and data augmentation. • Produce TFLite and JSON files to import into Android Studio.
TTS Algorithm (Text-to-Speech) [47]	Convert text to speech of animal name pronunciation.	<ul style="list-style-type: none"> • Implemented with Android TextToSpeech API.

5.3.6 Draw.io Setup

1. Navigate to <https://app.diagrams.net> and log in or sign up for an account.
2. Choose storage options.
3. Create a new diagram, then enter a file name.
4. After completing the diagram, click "Export" and choose a file format.

5.3.7 Figma Setup

1. Navigate to <https://www.figma.com> and log in or sign up for an account.
2. Create a new file.
3. Add plug-in components.
4. After completing the design, click the "Save" button or "CTRL + S".

5.4 System Operation

This section outlines the overall workflows for ResQMe, which is the animal rescue mobile application. It demonstrates the interactions between different types of users, such as guest users, registered users, and administrators, with the system. The goal of the system operation is to showcase how ResQMe functions in a world scenario. Screenshots and explanations illustrate the interaction between the frontend and backend provided.

5.4.1 Splash Screen

The initial loading splash screen is downloaded from LottieFiles, which appears once the application is launched. ResQMe uses Lottie JSON animations to perform an interesting greeting animation to enhance the user experience. The splash screen shows a fox that matches the “animal” theme in ResQMe as Figure 5.7.



Figure 5.7 Splash Animation

5.4.2 Sign Up and Log in

5.4.2.1 User

User can navigate to the drawer on the top left of the home page by clicking the hamburger menu icon, which shows the options log in / sign up. After clicking, the user will navigate to the login page, as shown in Figure 5.8a. If the user already has an existing account, then they can log in immediately; if the user does not have an account before, clicking the “Create New Account” button will take the user to the sign-up page, as shown in Figure 5.8b. Then, the user needs to fill in the basic information, such as username, phone number, email, and password. In Figure 5.8e, the email must be verified first to make sure that it is real. In Figure 5.8c and Figure 5.8d, the full name from the IC will be extracted once the user uploads the IC photo by using OCR and shows an error when the user enters an invalid photo.

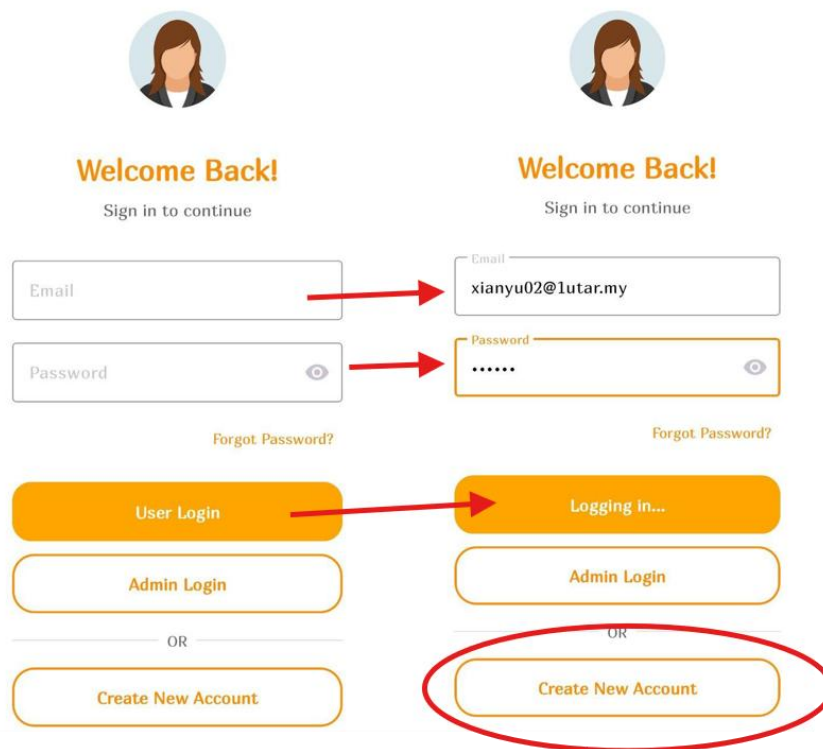
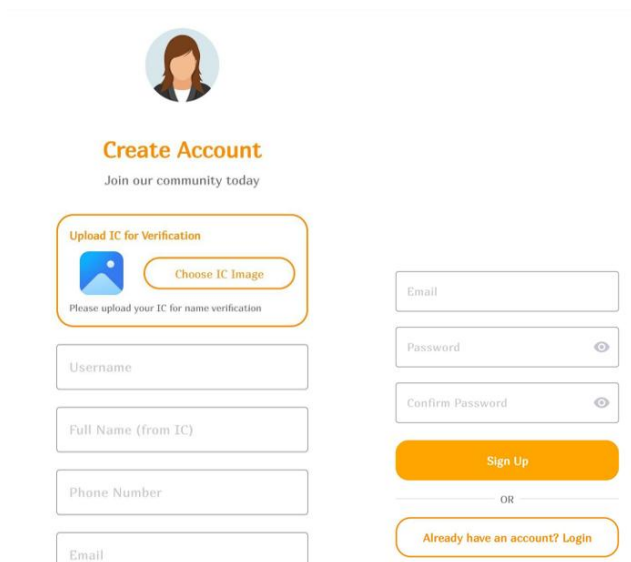


Figure 5.8a User Login



The 'Create Account' form features a header with a user icon and the text 'Create Account' and 'Join our community today'. It is divided into two main sections. The left section, titled 'Upload IC for Verification', includes a 'Choose IC Image' button and a note: 'Please upload your IC for name verification'. Below this are input fields for 'Username', 'Full Name (from IC)', 'Phone Number', and 'Email'. The right section contains input fields for 'Email', 'Password', and 'Confirm Password', each with a toggle icon. An orange 'Sign Up' button is positioned below the password fields. A horizontal line with 'OR' in the center separates the 'Sign Up' button from a 'Login' button, which is labeled 'Already have an account? Login'.

Figure 5.8b User Sign Up



This form shows the 'Upload IC for Verification' section with a successful upload. The 'Choose IC Image' button is present, and below the image, it states '✓ Name extracted: Eng Xian Yu'. The input fields are filled with the following information: 'Username' is 'xianyu', 'Full Name (from IC)' is 'Eng Xian Yu', 'Phone Number' is '0123456789', and 'Email' is '0123456789'. The 'Password' and 'Confirm Password' fields are filled with six dots. An orange 'Sign Up' button is at the bottom.

Figure 5.8c User Sign Up Information



This form shows the 'Upload IC for Verification' section with an invalid upload. The 'Choose IC Image' button is present, and below the image, it states '✗ Could not extract name or IC number from image. Please ensure the IC is clear and well-lit.' The input fields for 'Username', 'Full Name (from IC)', 'Phone Number', and 'Email' are empty. The 'Password' field is empty, and the 'Confirm Password' field is partially filled with six dots. A dark grey error message box is displayed above the 'Confirm Password' field, containing the text: 'Could not extract name or IC number from image. Please ensure the IC is cle...'. An orange 'Sign Up' button is at the bottom.

Figure 5.8d Invalid IC photo

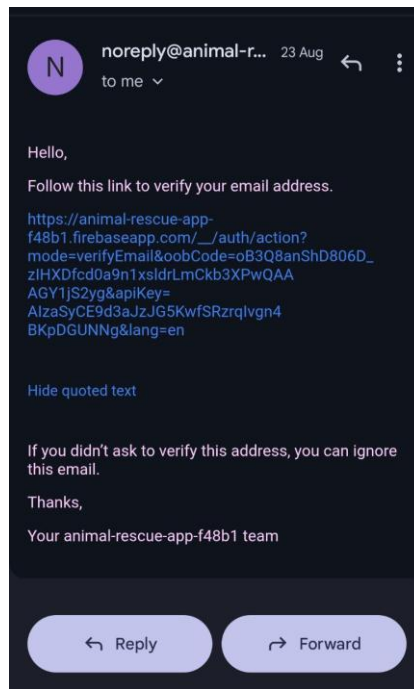


Figure 5.8e Email Address Verification

5.4.2.2 Admin

A security validation feature was added inside the admin login and sign up, and other flows like user login and sign up, as shown in Figure 5.9. After entering the unique code for admin, the system verifies the account by querying the userType for the user collection in Firestore. For admin, the needs of the admin access code are validated against the admin_config/access_codes document in Firestore using AdminCodeManager. Once the admin verifies authentication, they will open access to the approval of submission and publishing volunteer forms.

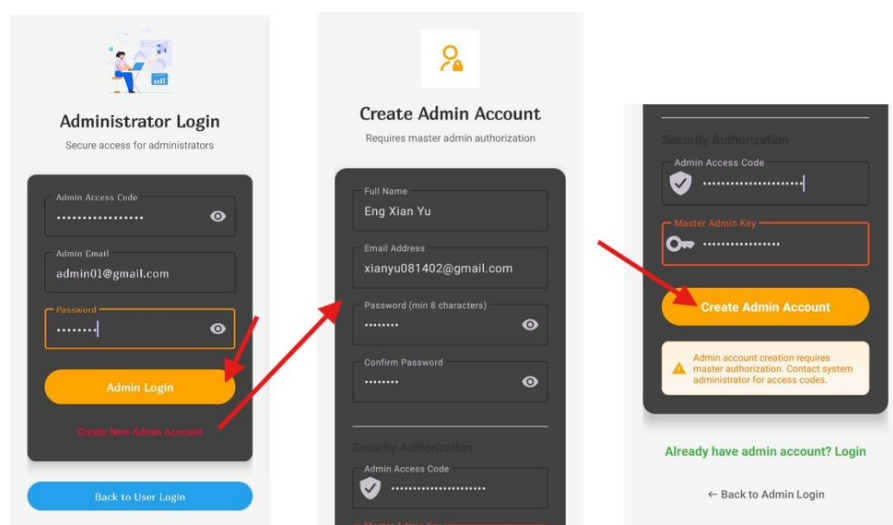


Figure 5.9 Admin Login and Sign Up

5.4.3 Forgot Password

If users forgot their password, they could find it back by navigating to profile settings and clicking the “Change Password” button, as shown in Figure 5.10a. A dialog box shows confirmation. After the user clicks “send reset password”, a reset email is sent to the user’s email address, and the user can change the password via email verification, such as Figure 5.10b.

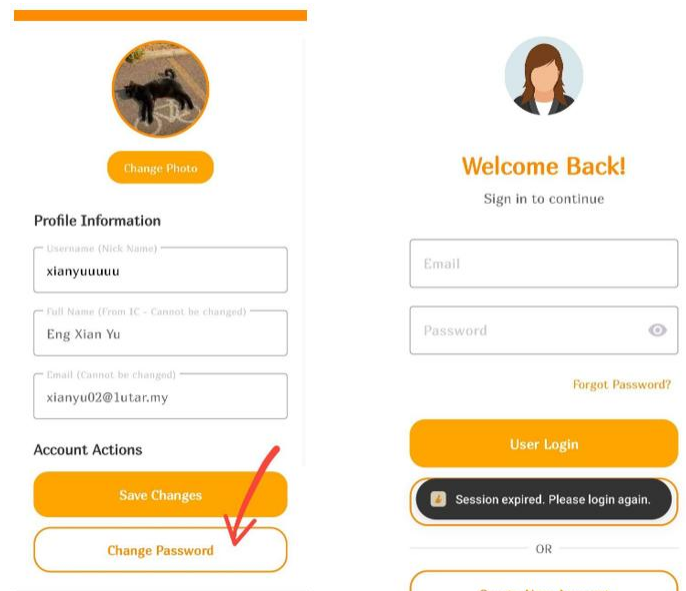


Figure 5.10a Change Password

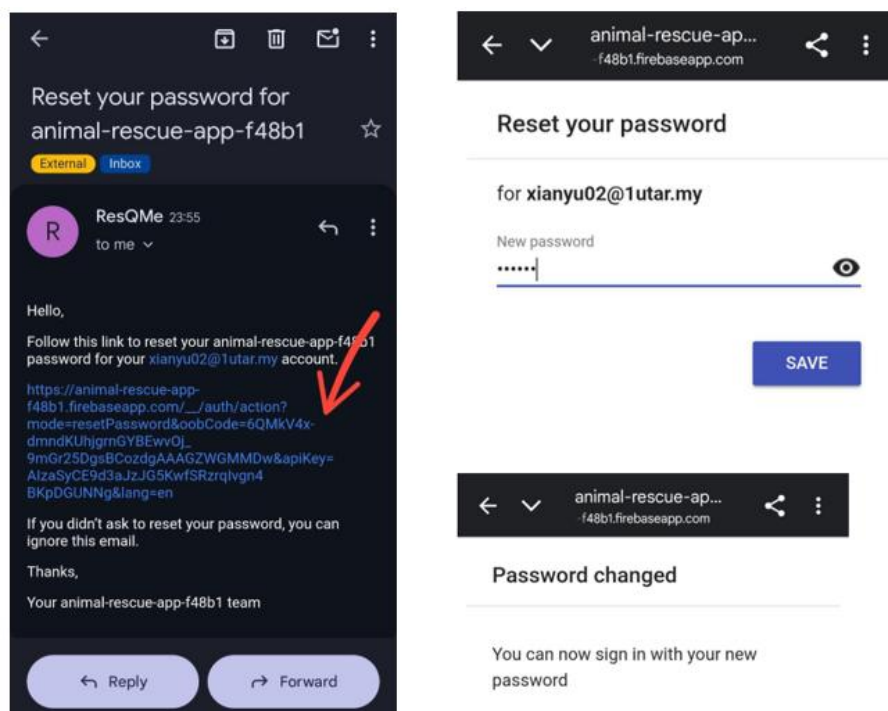


Figure 5.10b Reset Password

5.4.4 Home page

The home page acts as a default main fragment, as shown in Figure 5.11a and Figure 5.11b, and when the user first launches ResQMe, they will navigate to the home page by default. The home page provides quick access to main features such as the drawer, categories, and news and stories. BannerAdapter.java displays dynamic banners that capture and attract the user's eye, such as the hotline and volunteer section. The system will check and retrieve data from Firebase. If the user is new and not registered yet, the system will greet with “Hi Guest!”, if the user is registered, the system will greet with “Hi ‘username’!”.

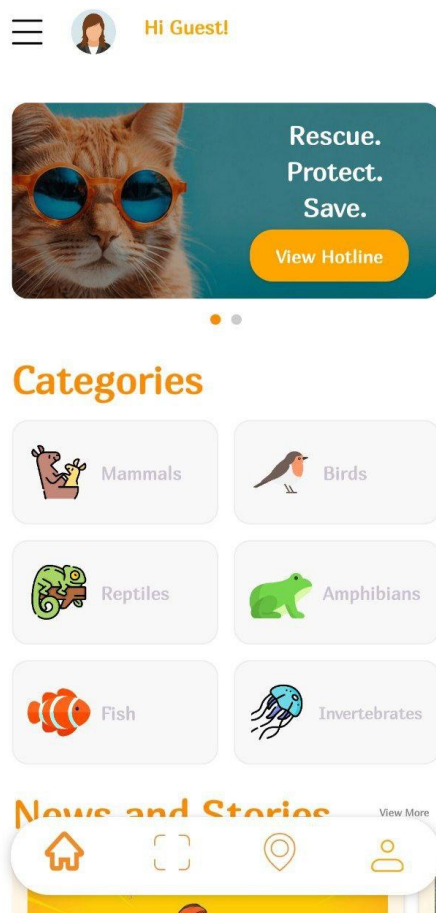


Figure 5.11a Home Page Part 1

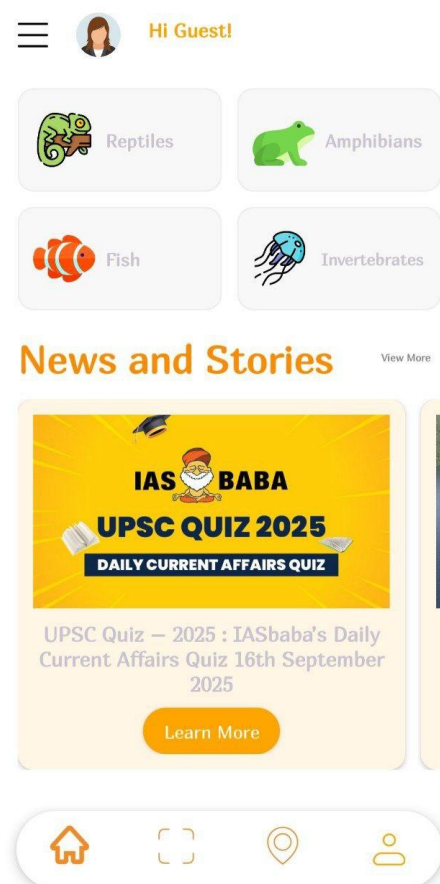


Figure 5.11b Home Page Part 2

5.4.4.1 Drawer

The navigational drawer is a side menu panel; it triggers when the user clicks the hamburger menu icon on the top left of the home page. It provides quick and consistent access for users to reach the page they want to, especially for an extra page, such as messages.

System check authentication for both users and admins. If users and admins are not registered or have not signed up yet, then the drawer only shows the “Login / Sign Up” button, and the identity text is “Guest User” and “Not logged in”. If the system detects a registered user, then it shows the navigational sections for users such as “My Reported Cases”, “Messages”, “Profile Settings”, “My Volunteer Applications”, “My Descriptions”, and “Logout, the text on top of the drawer changes to the user’s profile picture, username, and the user type. If the system detects an admin, the navigational sections will be “All Reports”, “Messages”, “All Descriptions”, “Volunteer Management”, and “Logout”. Figure 5.12a and Figure 5.12b show the logic. Once the system detects users logging in, the profile picture and username will change accordingly in the top left corner beside the hamburger menu icon.

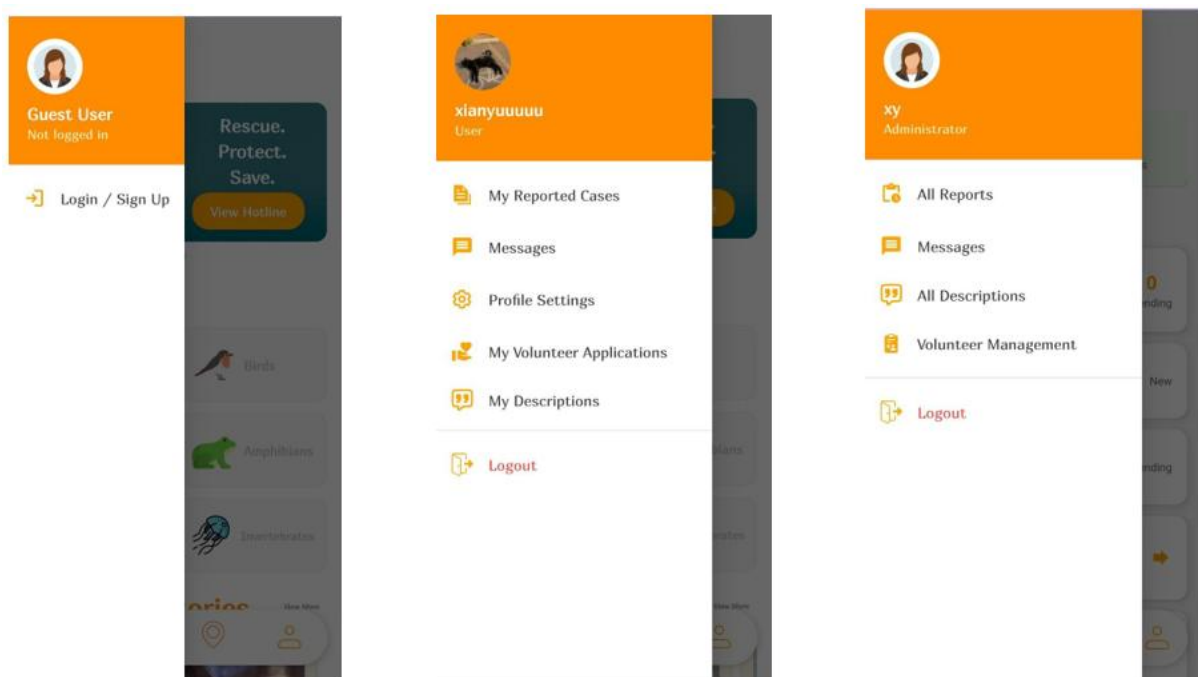


Figure 5.12a Drawer User Type

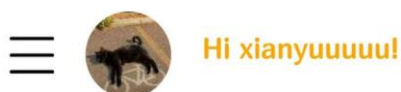


Figure 5.12b Update Profile and Username

Besides, if both users and admins do not act on submitting forms or reports, the approval or rejection actions, empty alerts are shown to users and admins, as shown in Figure 5.12c to Figure 5.12g.

For users:

Bachelor of Computer Science (Honours)

Faculty of Information and Communication Technology (Kampar Campus), UTAR

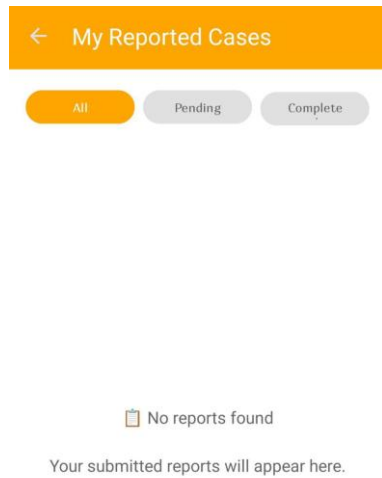


Figure 5.12c Empty Report



Figure 5.12d Empty Messages

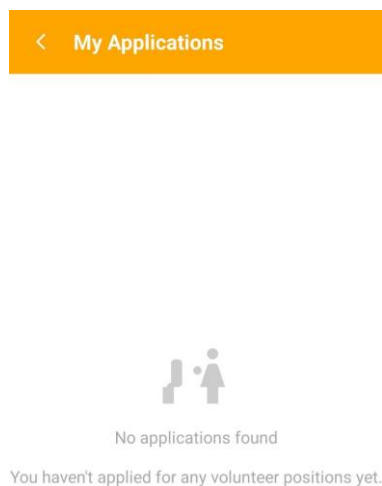


Figure 5.12e Empty Application

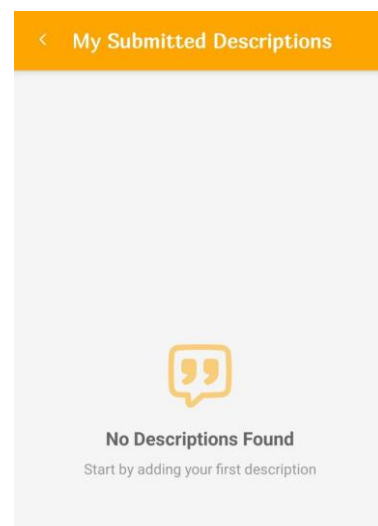


Figure 5.12f Empty Description

For admins, reports, messages, and descriptions are the same as those of users; the empty state in volunteer management is as follows:

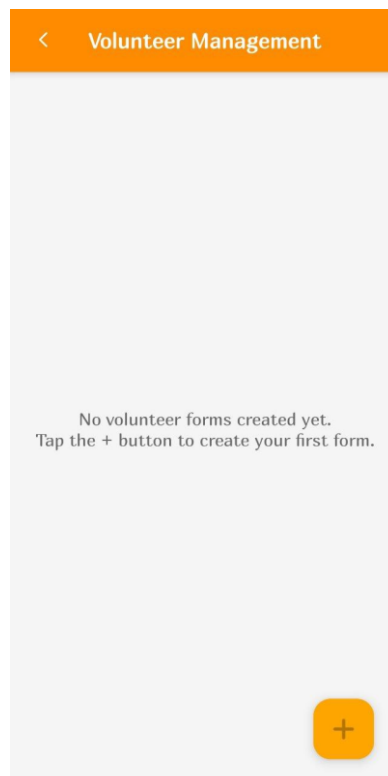


Figure 5.12g Empty Volunteer Management

5.4.4.2 Hotline Section

Based on Figure 5.13a, the hotline page provides all hotline numbers for emergency rescue teams according to each state. The top one is National Wildlife Rescue & Reporting. If users do not know their exact state they are, they can either call with phone numbers or email. A rule-based chatbot is embedded to help users locate the hotlines quickly. An AI chatbot also provided a conversation to find the hotline number quickly, as shown in Figure 5.13b.

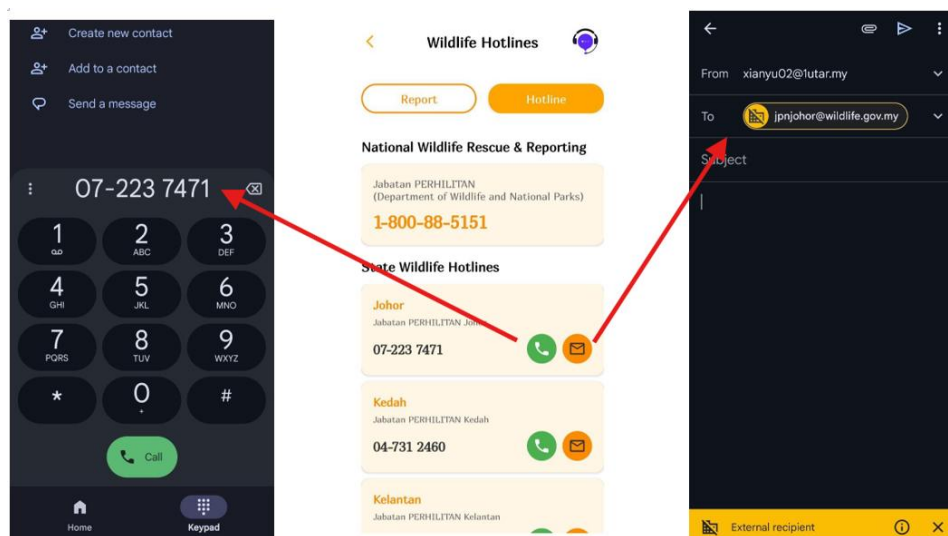


Figure 5.13a Hotline Call and Email

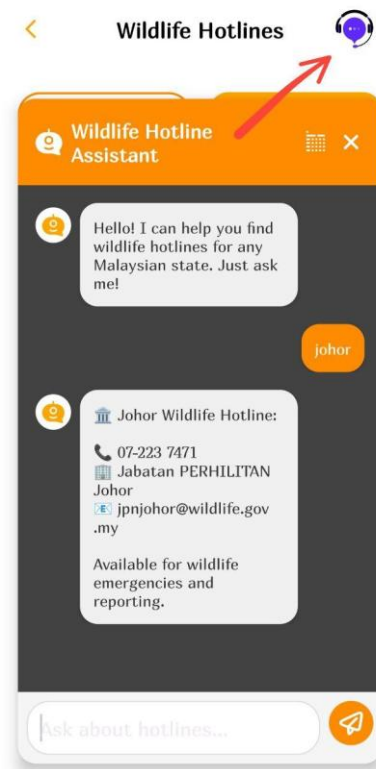


Figure 5.13b Rule-Based Chatbot in Hotline

5.4.4.3 Report Section

This section enables users to submit wildlife rescue reports in cases of emergency. The form includes reporter information, as users and admins need to register before launching the report section. Therefore, the system will automatically retrieve the data while the user signs up and display it in the reporter's information. Users can also select incident information, location details, and photo evidence for additional information. If users do not know where they are, they can click "Get Current Location", and the system will automatically perform location tracking and fill in the location address. Figures 5.14a to 5.14c show the logic.

After the user submits the report, notifications will be sent to both the user and the admin, as shown in Figure 5.14d to Figure 5.14f. Admin can either approve or reject the report.

Report Wildlife Case

Report Date & Time: 18/09/2025 00:06

Reporter Information

Full Name *
Eng Xian Yu

Phone Number *
+60182685851

Email Address (Optional)
xianyu02@lutar.my

Incident Information

Wildlife Type *
Select Wildlife Type

Urgency Level *
Select Urgency Level

Figure 5.14a Auto-Filled Reporter Information

Incident Information

Wildlife Type *

- Select Wildlife Type
- Injured Wildlife
- Trapped Wildlife
- Orphaned Wildlife
- Dead Wildlife
- Wildlife in Human Area
- Illegal Wildlife Trade
- Other

Urgency Level *

- Select Urgency Level
- Critical - Immediate Action Required
- High - Action Required Today
- Medium - Action Required This Week
- Low - Non-Urgent

Figure 5.14b Dropdown for Wildlife Type and Urgency Level

Detailed Description *

A dead rabbit found beside Taman Putih.


Location Information

Location Address *

Bandar Barat, 31900 Kampar, Perak


Get Current Location

Photo Evidence (Optional)



Get Current Location

Photo Evidence (Optional)



Take Photo Select Photo

Submit Report

For immediate emergency situations, call the national hotline: 1-800-88-5151

Figure 5.14c Location Information and Photo Evidence

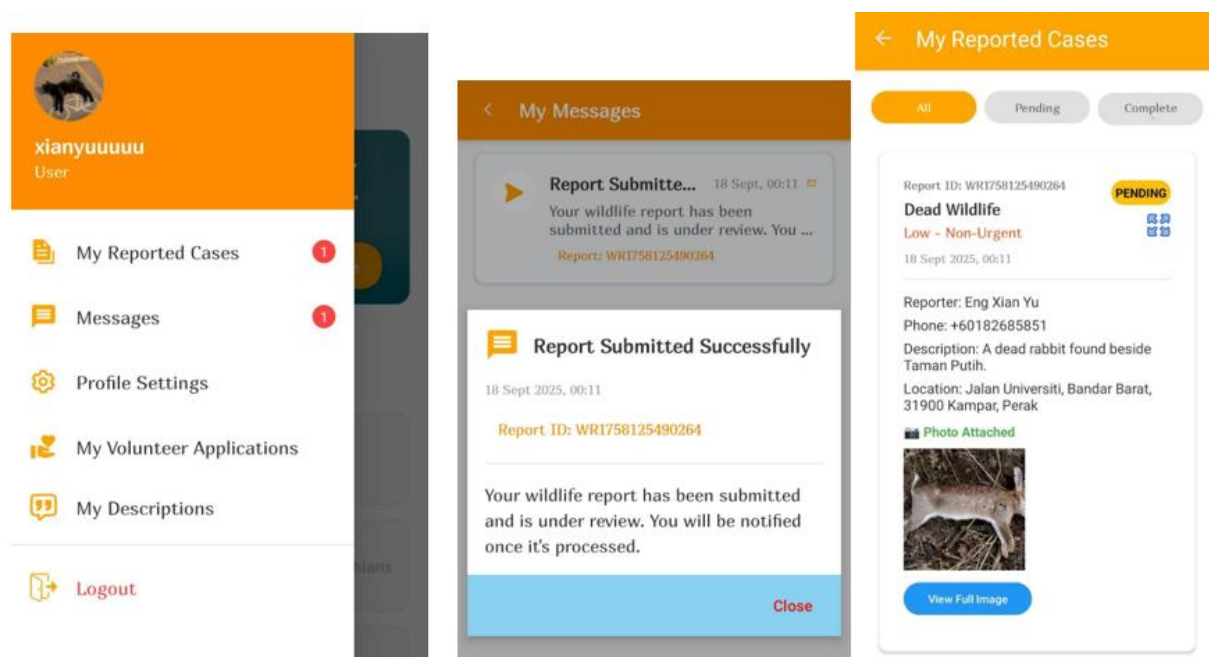


Figure 5.14d Notifications for User

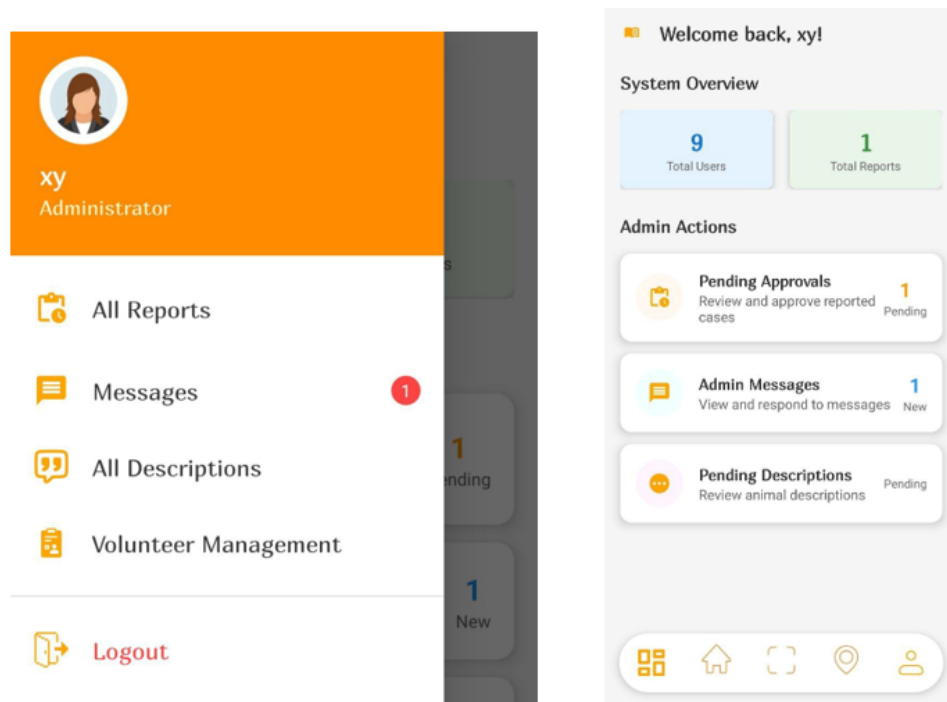


Figure 5.14e Notifications from Both Drawer and Admin Dashboard

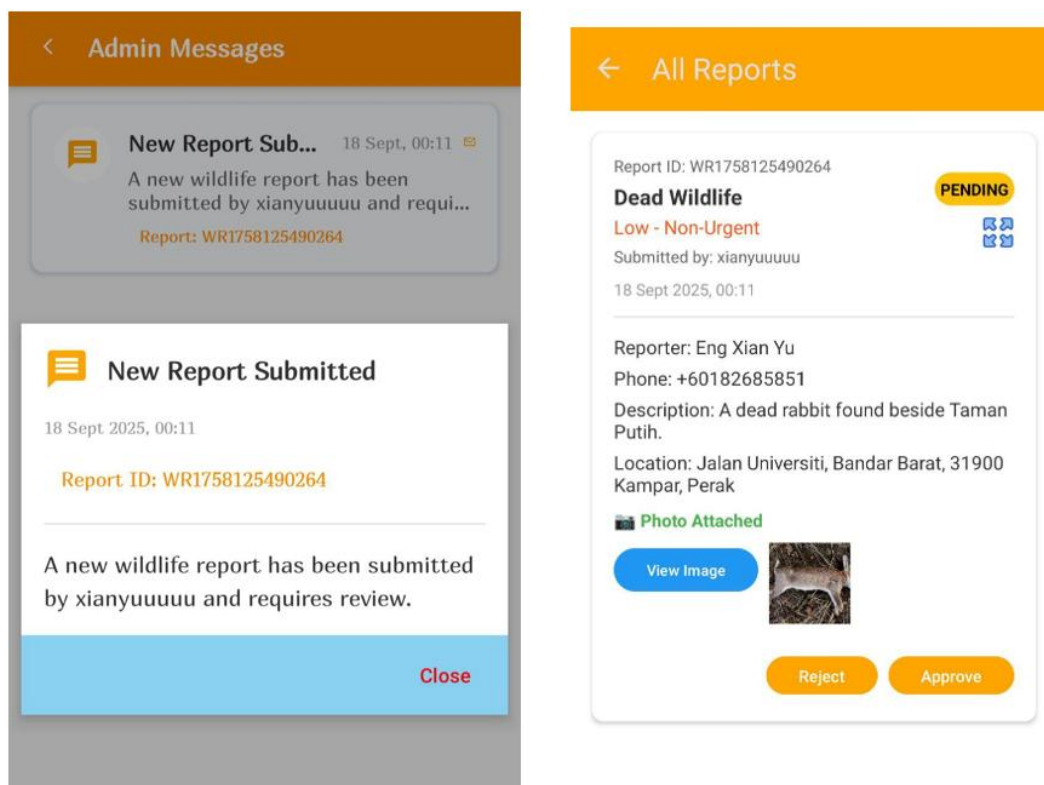


Figure 5.14f Notification and Submission Approval

5.4.4.4 Volunteer Section

The volunteer page listed all volunteer opportunities, which were created by the admin and stored inside the `volunteer_forms` collection in Firestore, as shown in Figure 5.15a and Figure 5.15b. After the user clicks “Volunteer Now” from the volunteer banner, it shows all volunteers created following the starting level, such as beginner, intermediate, advanced, and expert. The user can choose one of the forms and apply; this application will be sent back to the admin for approval. Notifications are received from both the user and admin once the submission is successful. Admin can edit the description and information of the form.

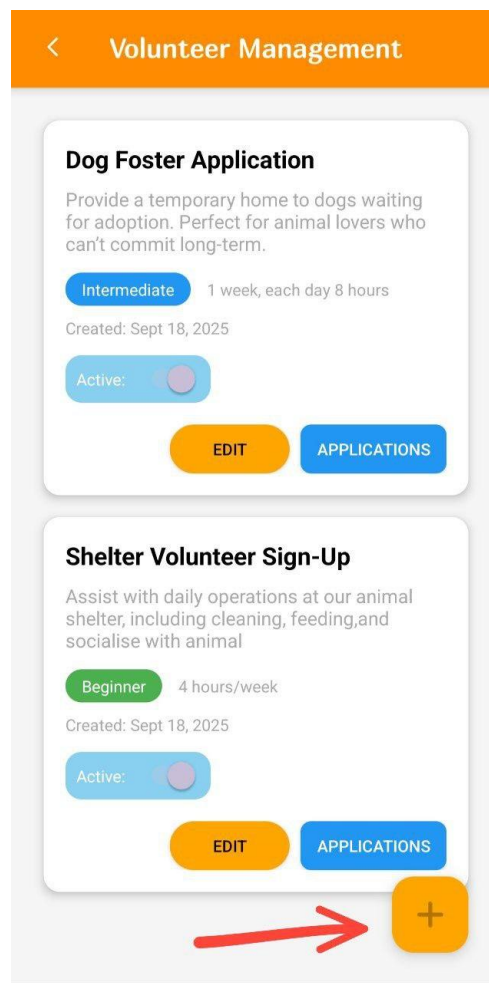


Figure 5.15a Admin Create Volunteer Form

Figure 5.15b Volunteer Form

On the home page, the user can click the “Volunteer Now” button to review existing volunteer opportunities created by the admin, as shown in Figure 5.15c and Figure 5.15d.

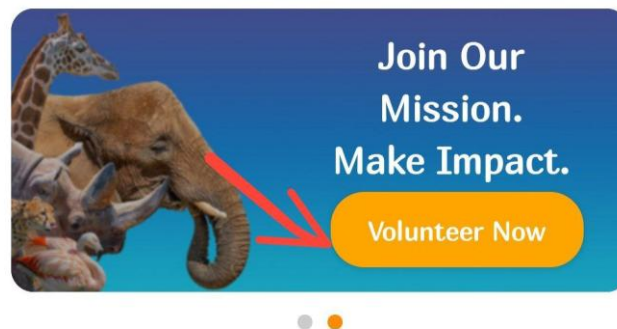


Figure 5.15c Button Volunteer Now

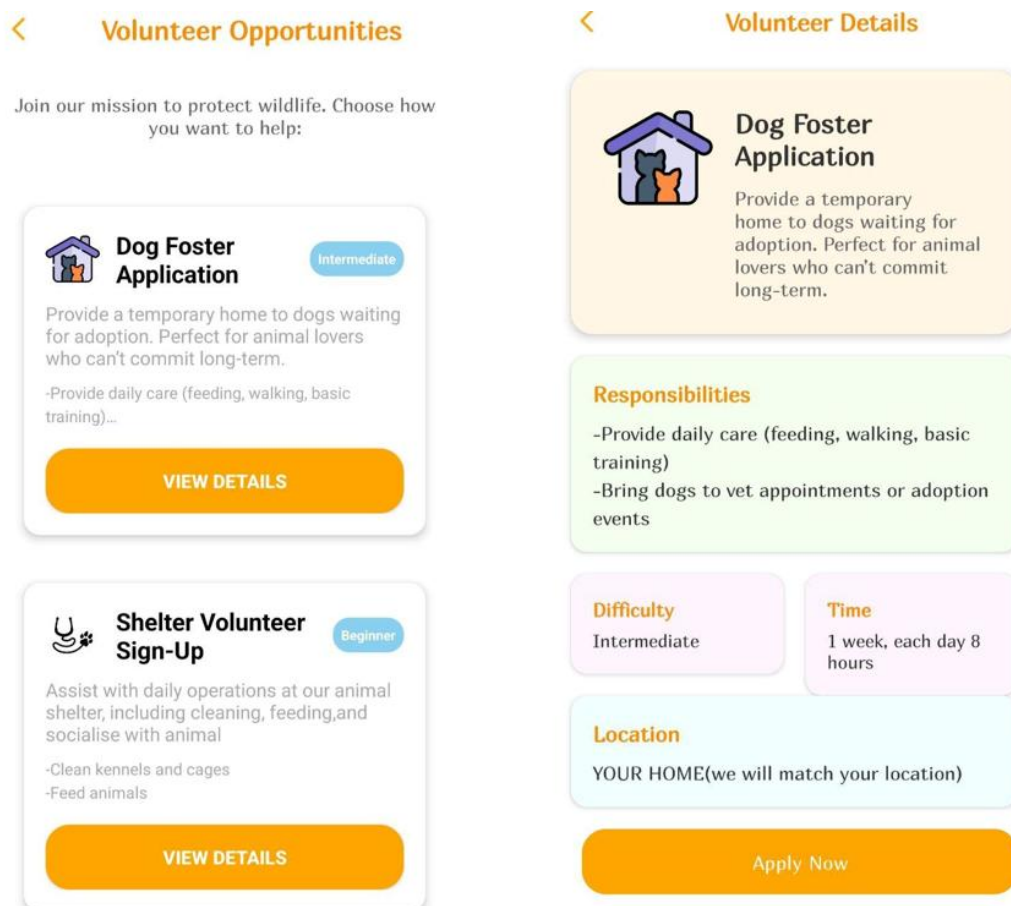


Figure 5.15d Volunteer Opportunities and Details

In Figures 5.15e to 5.15g, the user can apply the desired volunteer activity, and the application will be sent back to the admin for review via notification reminder. Admin can edit the form if they find something mistaken, as shown in Figure 5.15h.

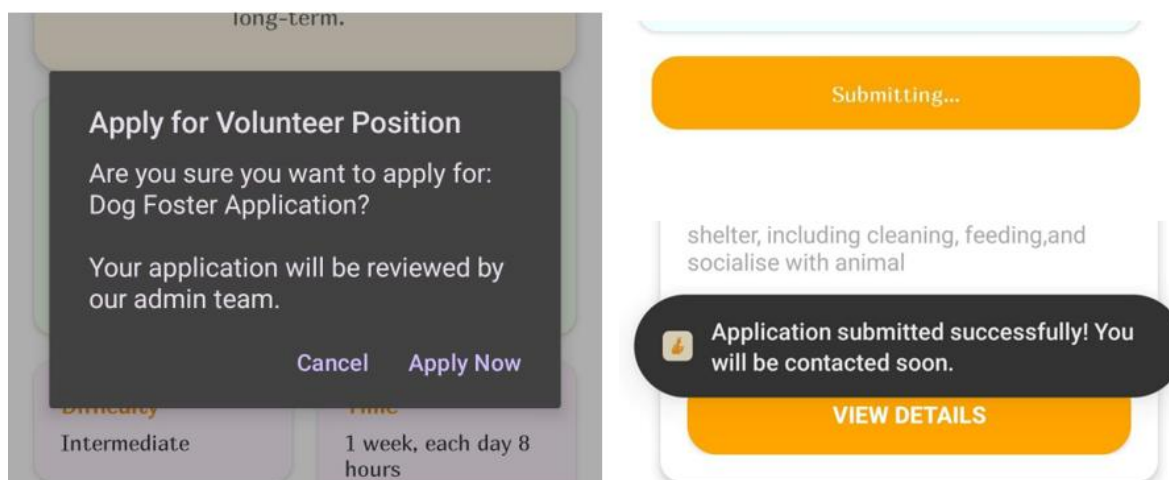


Figure 5.15e Application Submitted Successfully

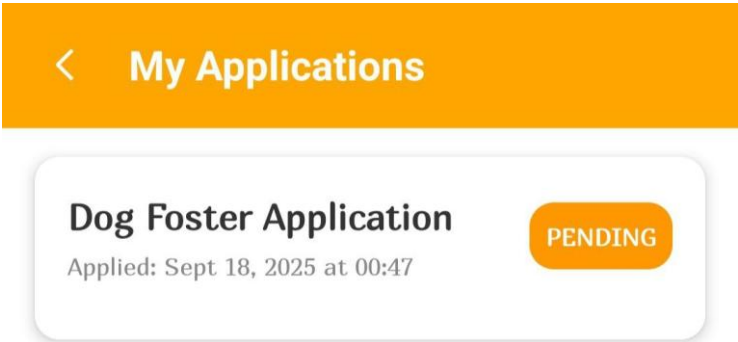


Figure 5.15f Notification for User

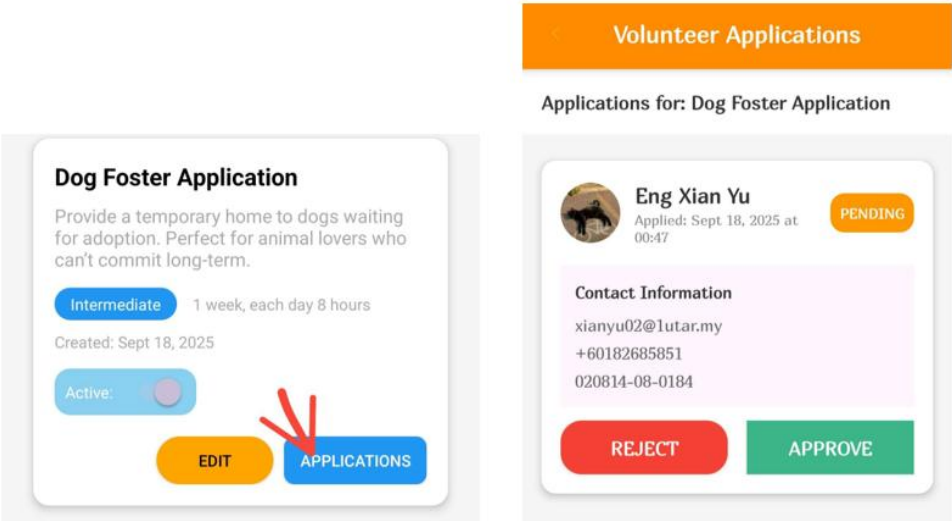


Figure 5.15g Information Shown

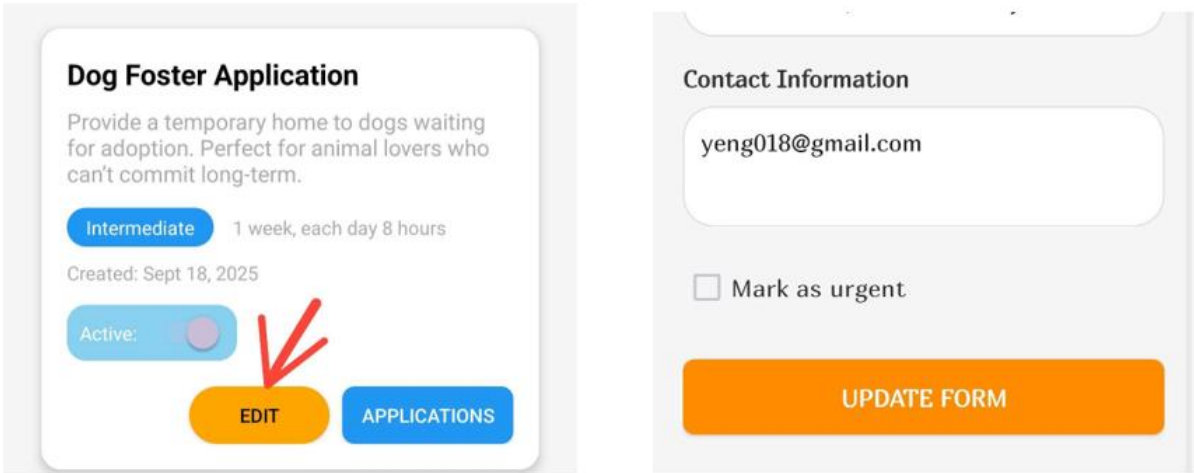


Figure 5.15h Admin Update the Form

5.4.4.5 Categories Section

From Figure 5.16a, the categories section provides different types of animals such as mammals, birds, reptiles, amphibians, fish, and invertebrates. After clicking inside one of the categories, it will show all animals in that category, as shown in Figure 5.16b. Search functionality is tested as shown in Figure 5.16c. The Animal Detail Page provides various features, including an AI chatbot, taxonomic classification, and user description.

If the user wants to submit a description, clicks the “Add My Description” button, it will navigate the user to the Add Description page, which has guidelines, a category, and a description edit box. After the user submits the description, a notification will be sent to both the admin and user, waiting for approval. Figure 5.16d to Figure 5.16j shows all the processes.

Categories

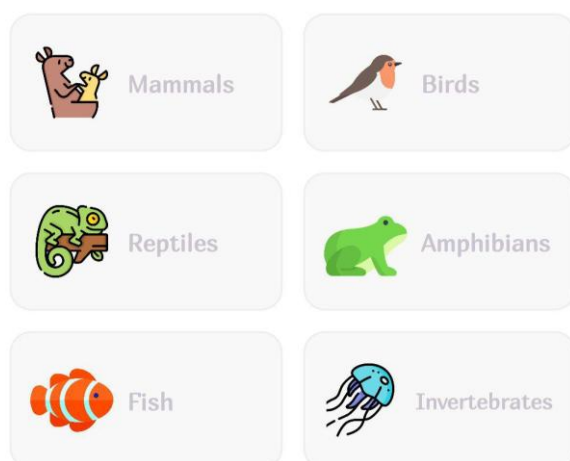


Figure 5.16a Categories Section

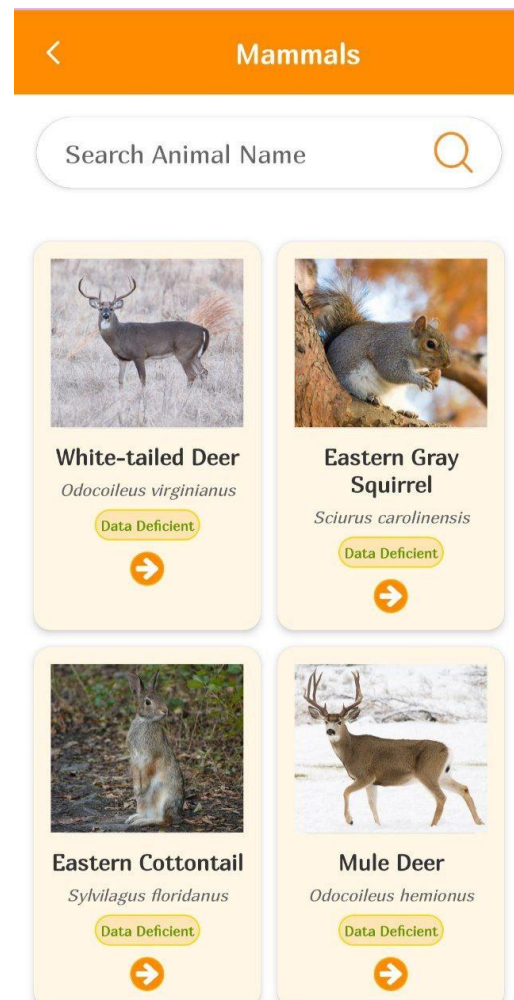


Figure 5.16b Animal Categories Detail Page

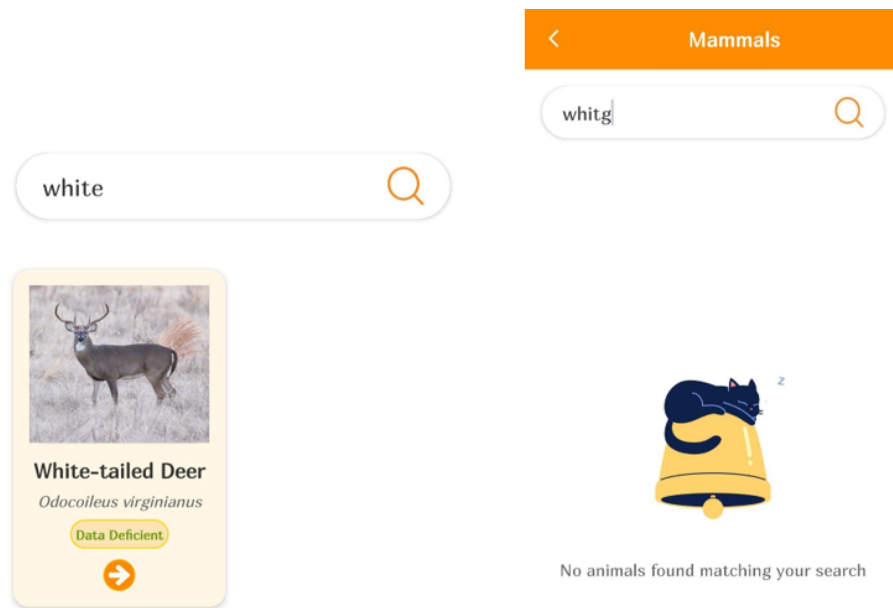


Figure 5.16c Search Functionality for Categories Section

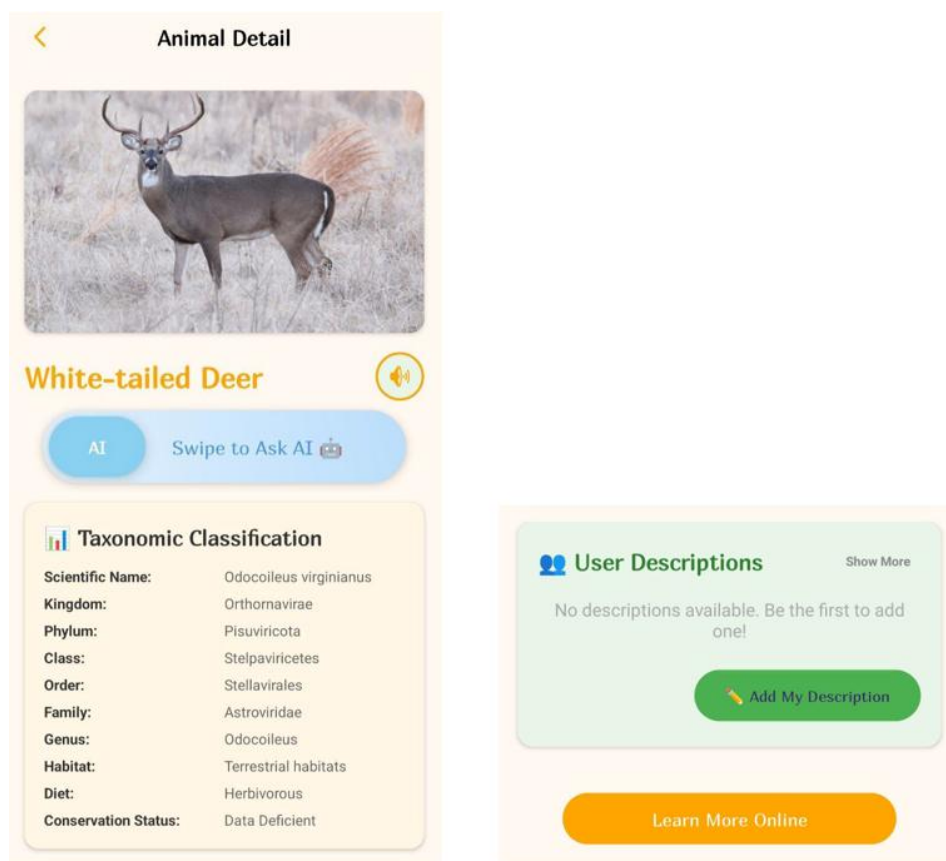


Figure 5.16d Animal Detail Page

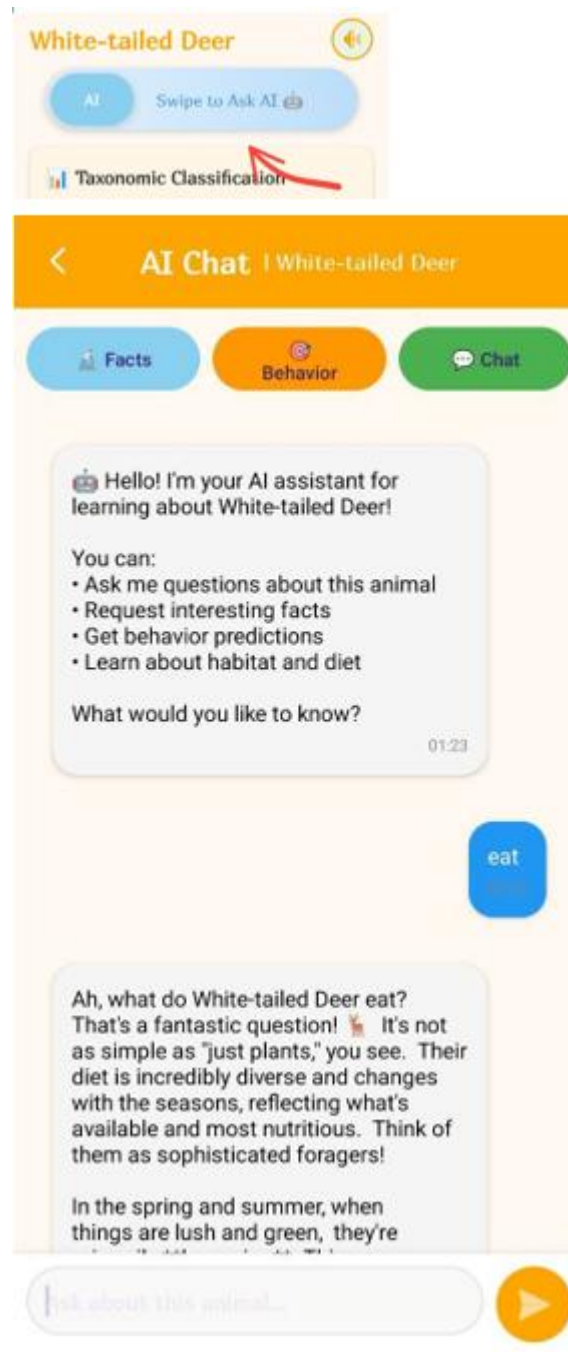


Figure 5.16e AI Chatbot

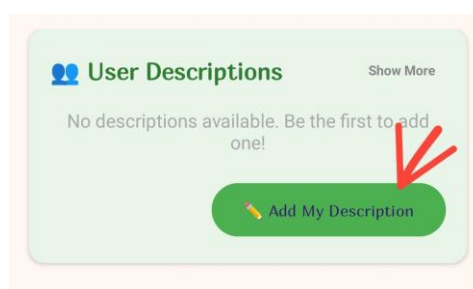


Figure 5.16f Add Description Button

< Add Description for White-tailed Deer

Guidelines

- Minimum 20 characters
- Be descriptive and informative
- Include behavior observations
- Optional: Add a photo
- Admin approval required

Category: Mammals

Your Description

Your Description

white underside of their tail, which they raise when alarmed. They play a crucial role in forest ecosystems but can also impact vegetation and agriculture when populations are high.

Add Photo (Optional)

Add Another Photo (1 selected)

Submit for Approval

Figure 5.16g Add Description Page

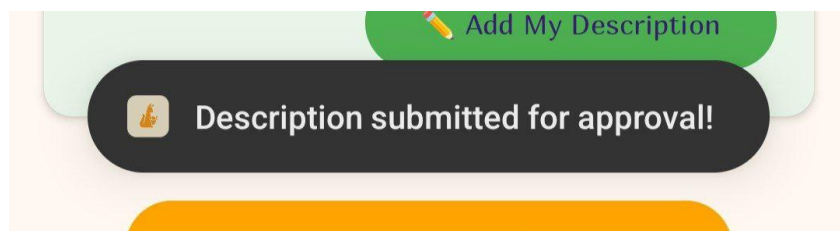


Figure 5.16h Description Submitted Successfully

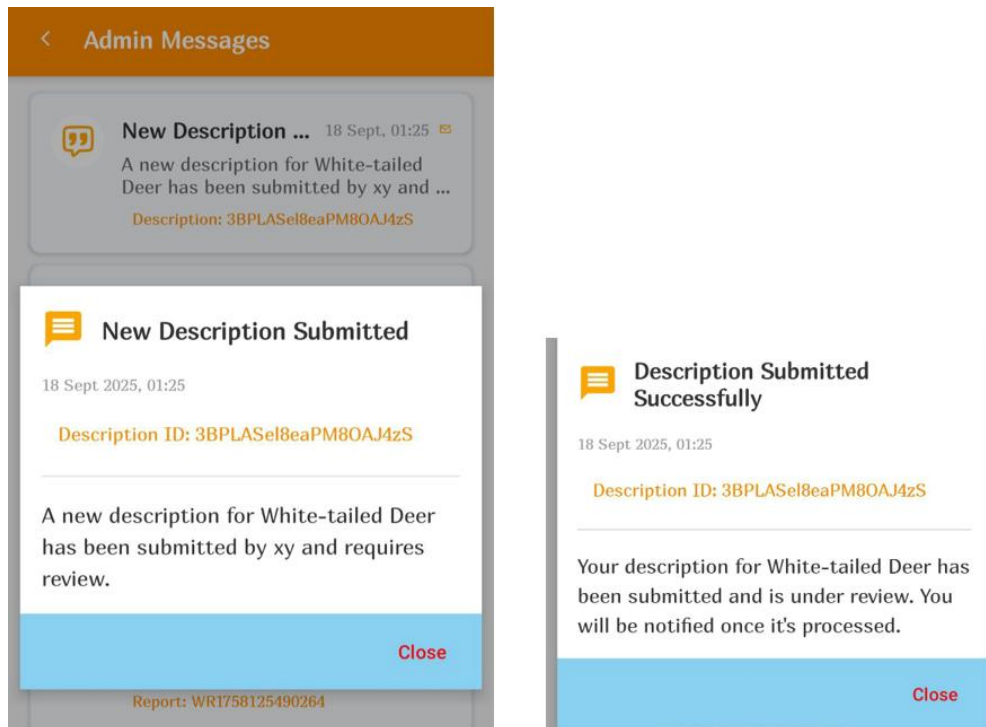


Figure 5.16i Notification for Both Admin and User

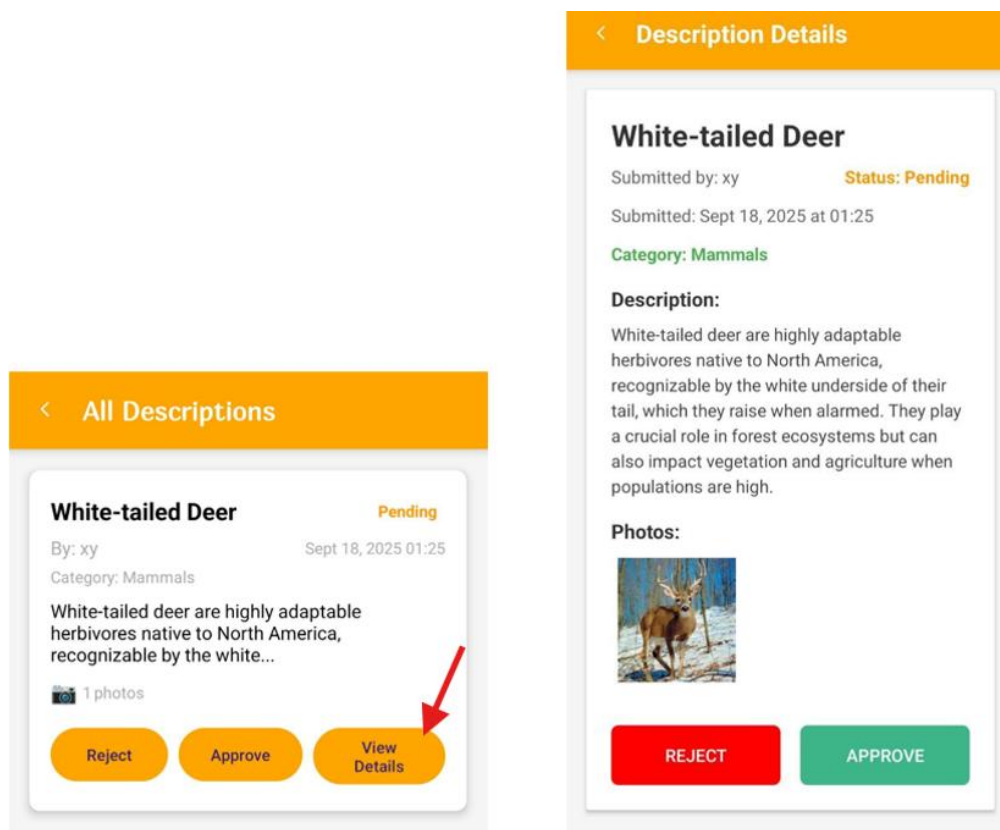


Figure 5.16j Description Waiting for Approval

5.4.4.6 Admin Approval Process

The submitted forms, such as wildlife rescue reports, volunteer forms, and description forms from users, will be sent back to the admin for approval. Once the admin approves or rejects, the notification will show in the messages section for both the admin and users, and the status of that form will be changed. All submission processes show from Figure 5.17a to Figure 5.17i.

Wildlife Report:

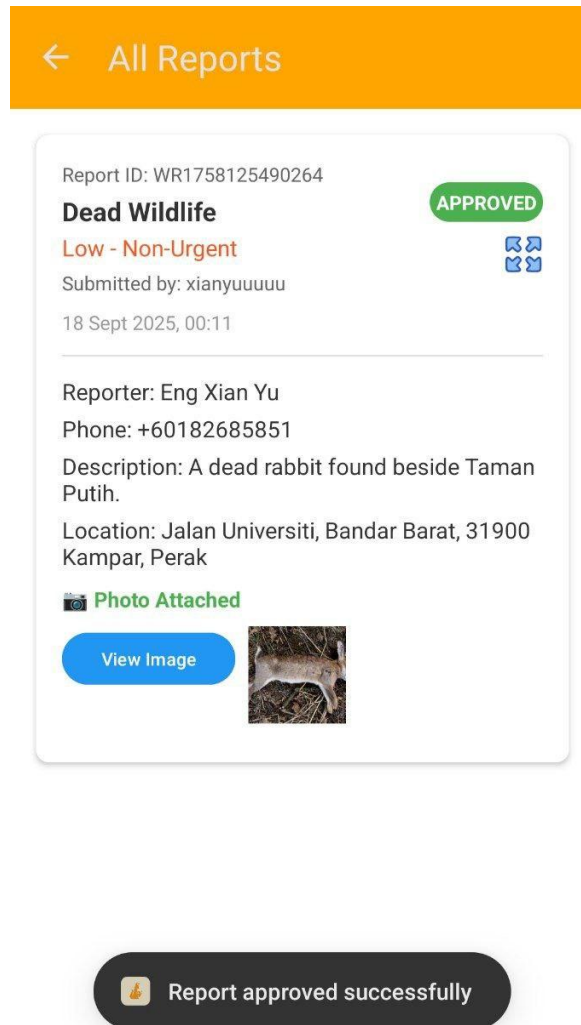


Figure 5.17a Admin Approved Wildlife Report

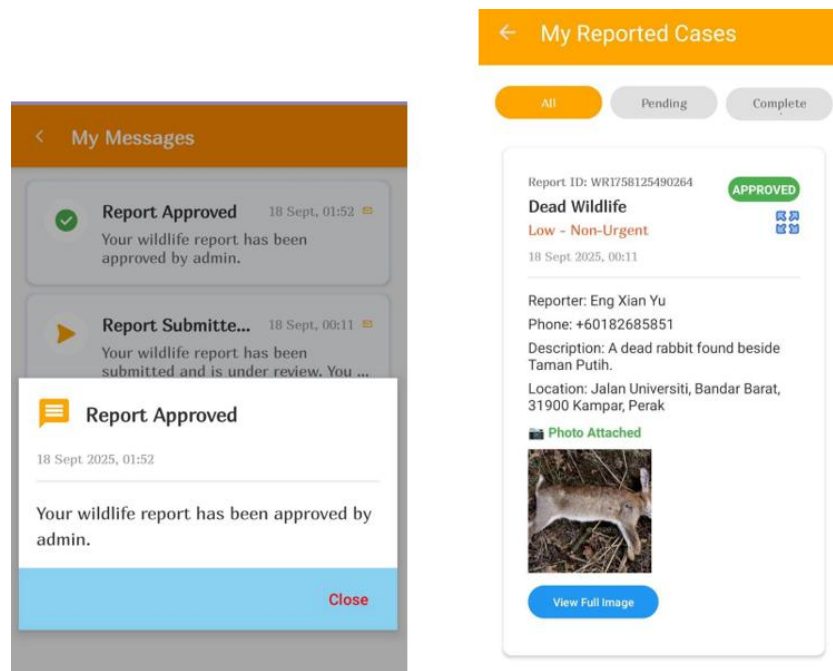


Figure 5.17b Notification for User and Status Updated

Volunteer Application:

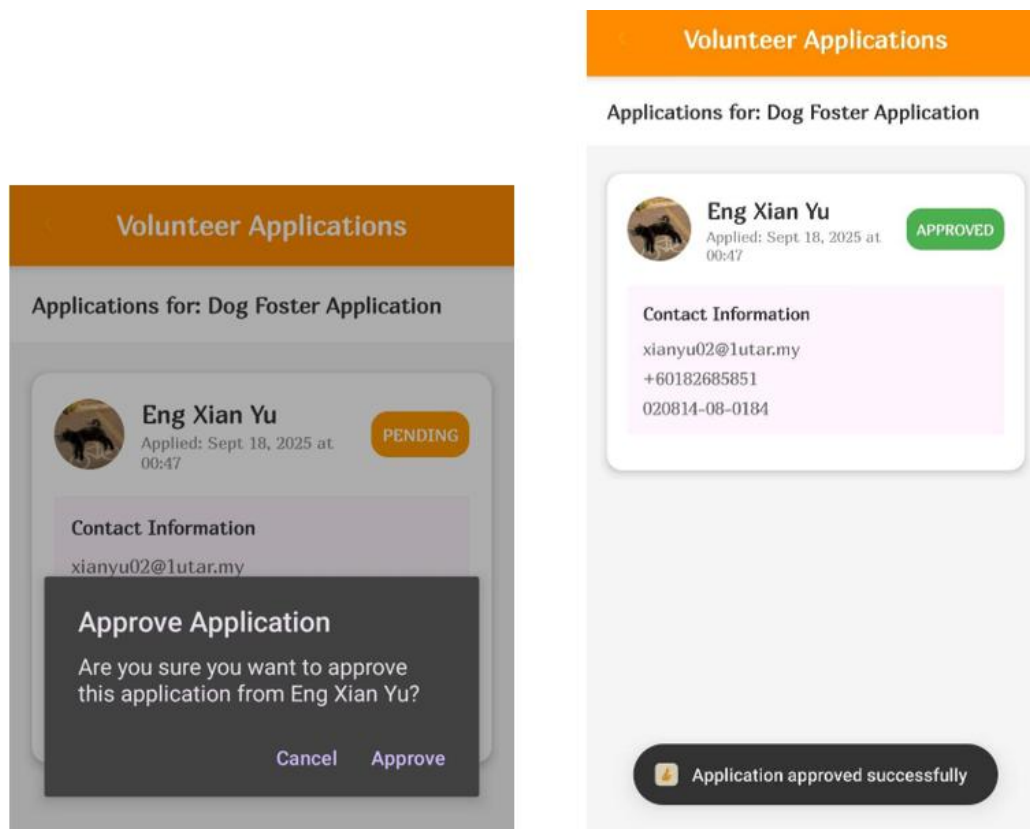


Figure 5.17c Application Approved and Status Updated

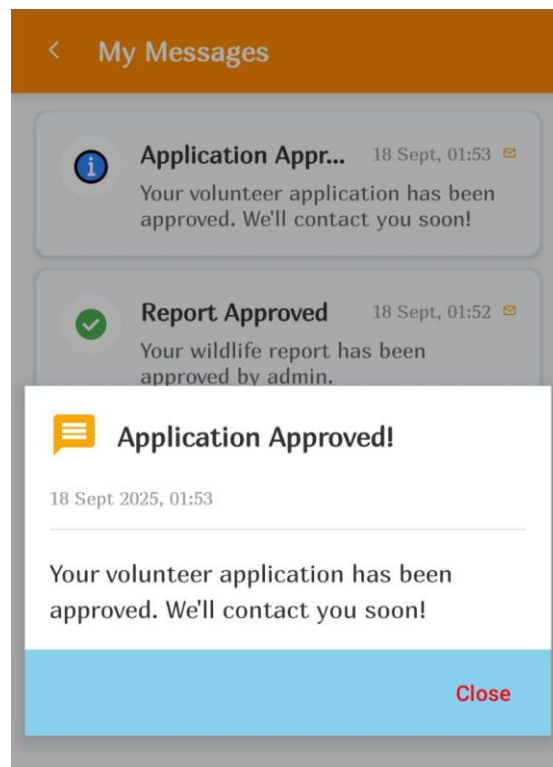


Figure 5.17d Notification for User

Description submission:

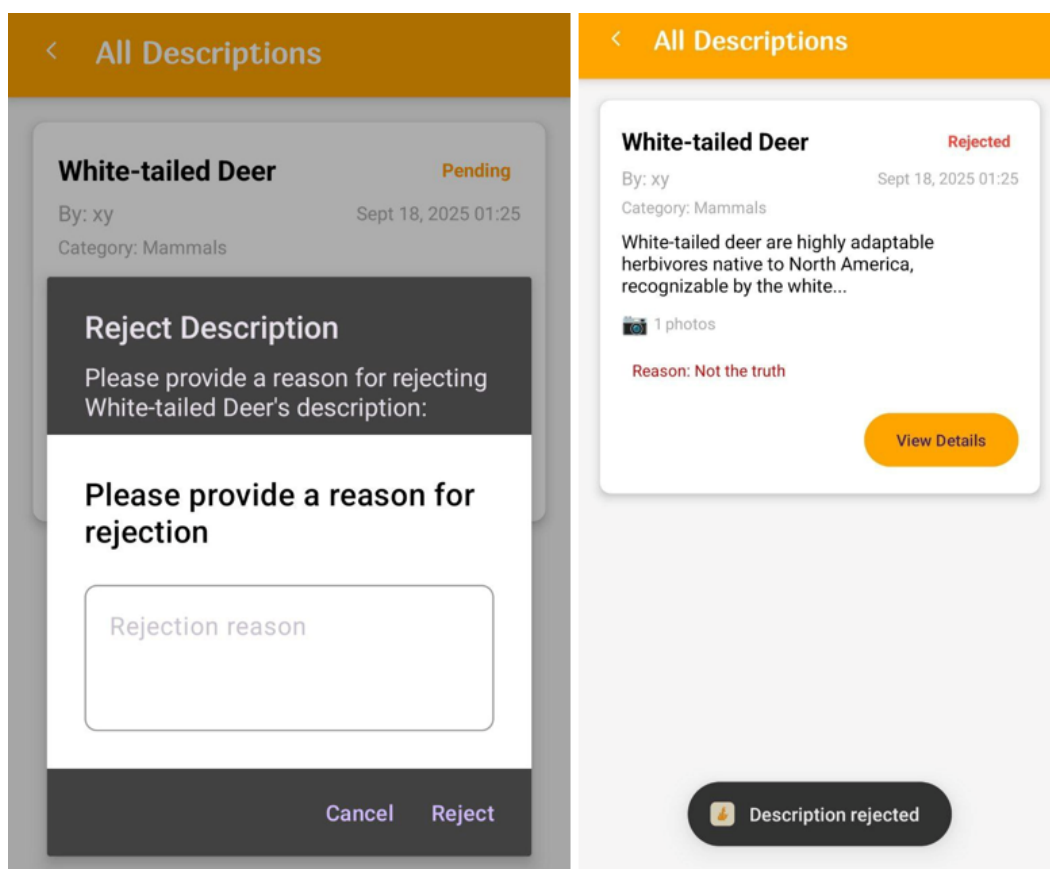


Figure 5.17e Description Rejected and Reason Given

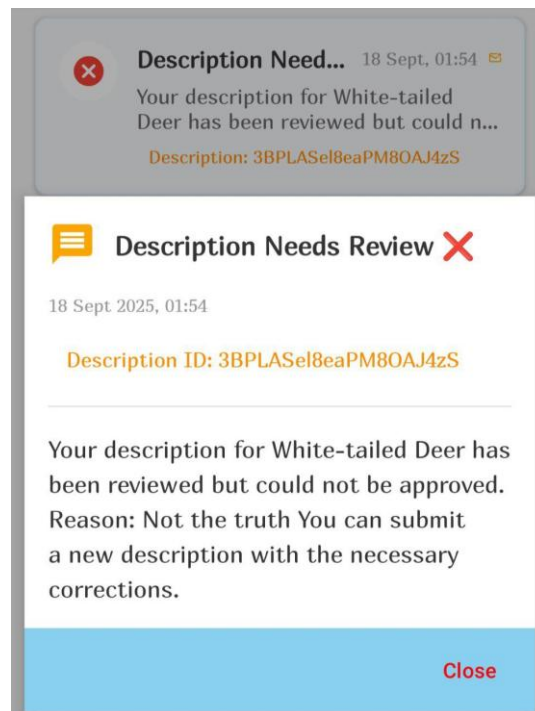


Figure 5.17f Notification for User

Description rejected by the admin will not show inside the animal detail page. If description accepted:

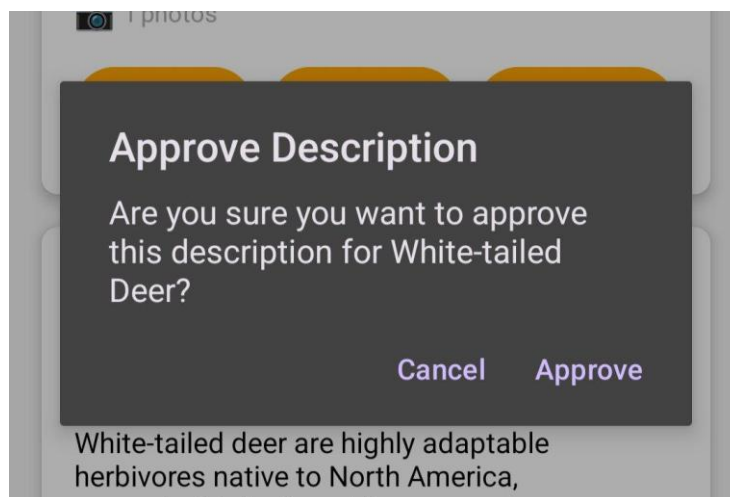


Figure 5.17g Description Approve Confirmation

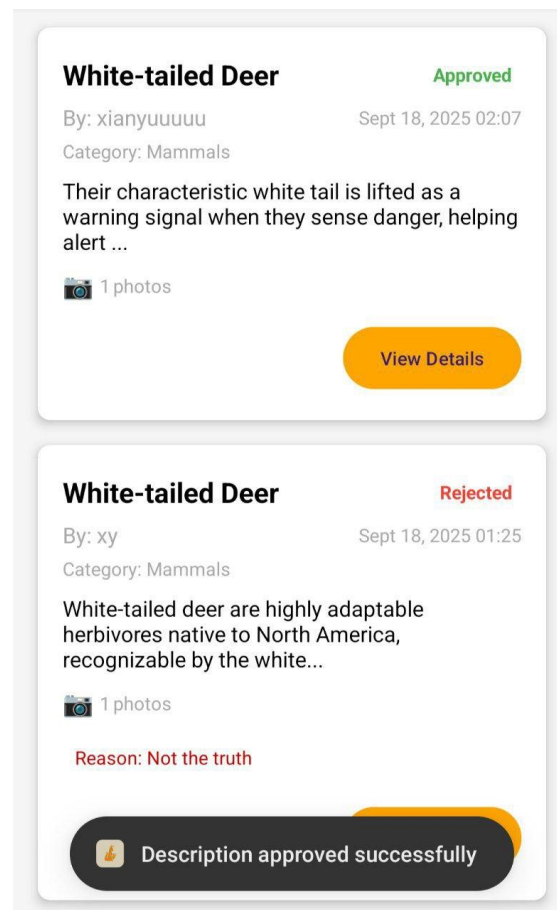


Figure 5.17h Description Status Changed

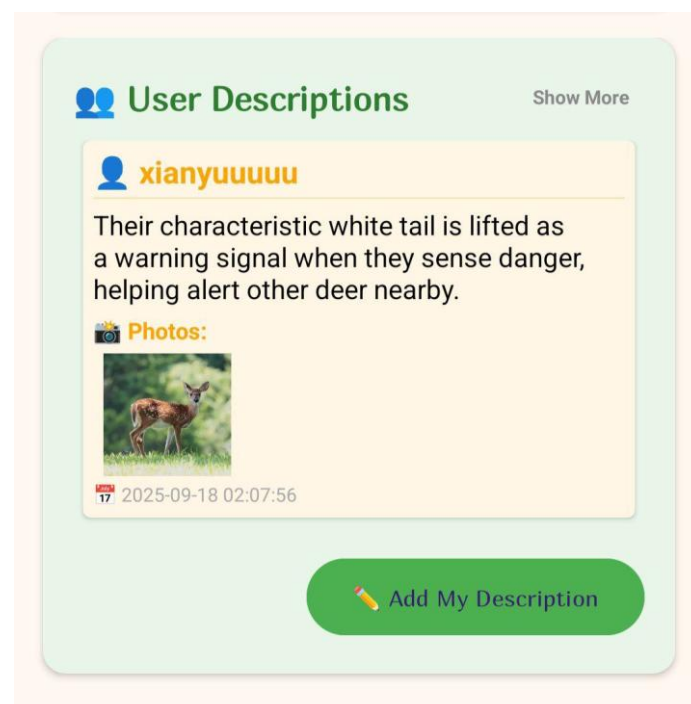


Figure 5.17i Description Updated for All Users

5.4.4.7 News and Stories

From Figures 5.18a to 5.18c, the News and Stories Section provides articles to enhance users' understanding and awareness. The user can click the “View More” button to review all news. Each article includes an image, title, source, date, description, and a “Read Full Article” button.

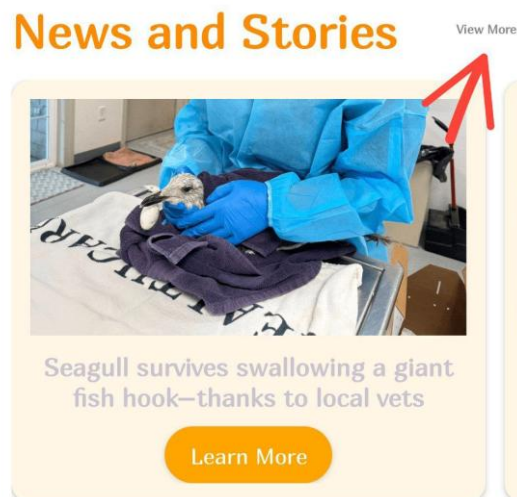


Figure 5.18a View More Button in News and Stories

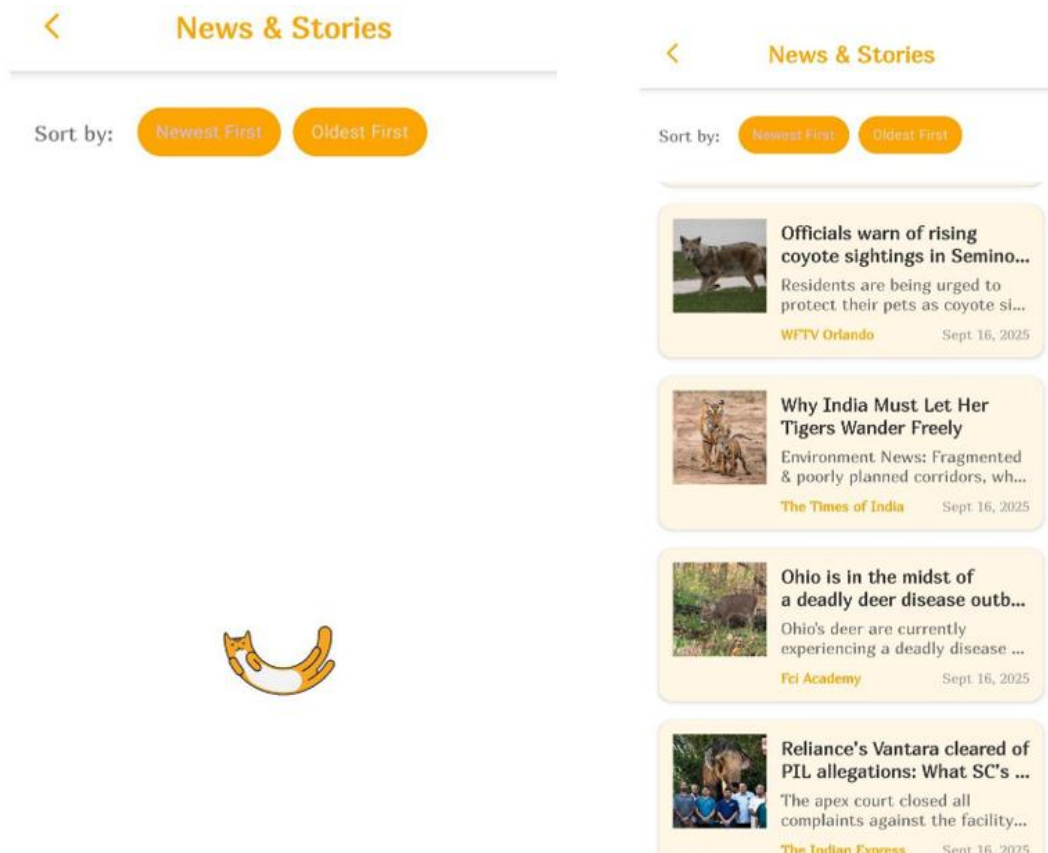


Figure 5.18b News Loading and Article Shows

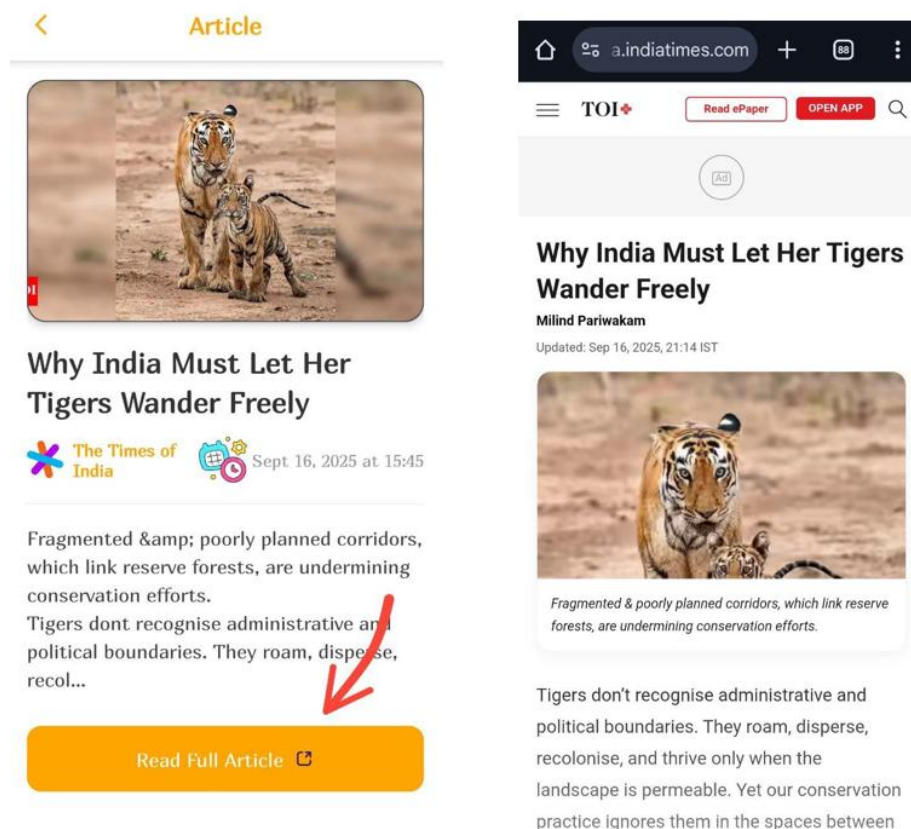


Figure 5.18c Read Full Article and Navigate to Website

5.4.5 Camera Page

The camera page enables users to capture a photo from ResQMe directly, like Figure 5.19a, and users can also choose to select a photo from the gallery, like Figure 5.19b. Once the photo is uploaded, the user navigates to the animal description page to review the information of the animal uploaded after the animal recognition process by using CNN models, as shown in Figure 5.19c. Figures 5.19d and 5.19e show that the user clicks the love icon to save the animal inside Firebase, and on the saved animal page, the “Learn More” button will navigate to the A-Z Animal website for extra knowledge.

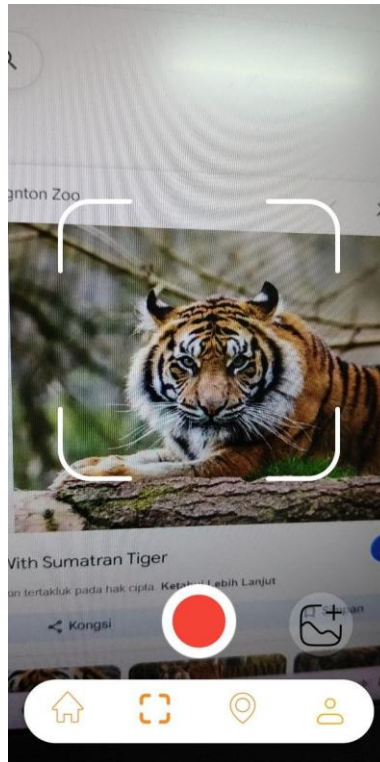


Figure 5.19a Capture Image in Camera View

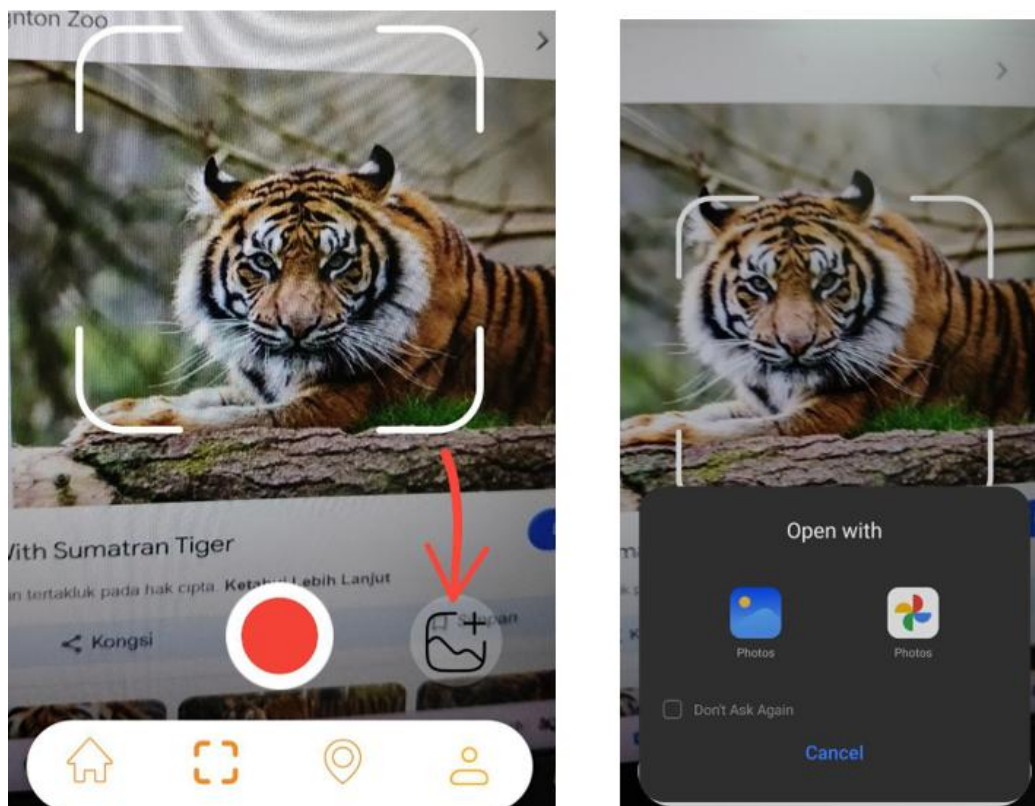


Figure 5.19b Choose Image from Gallery

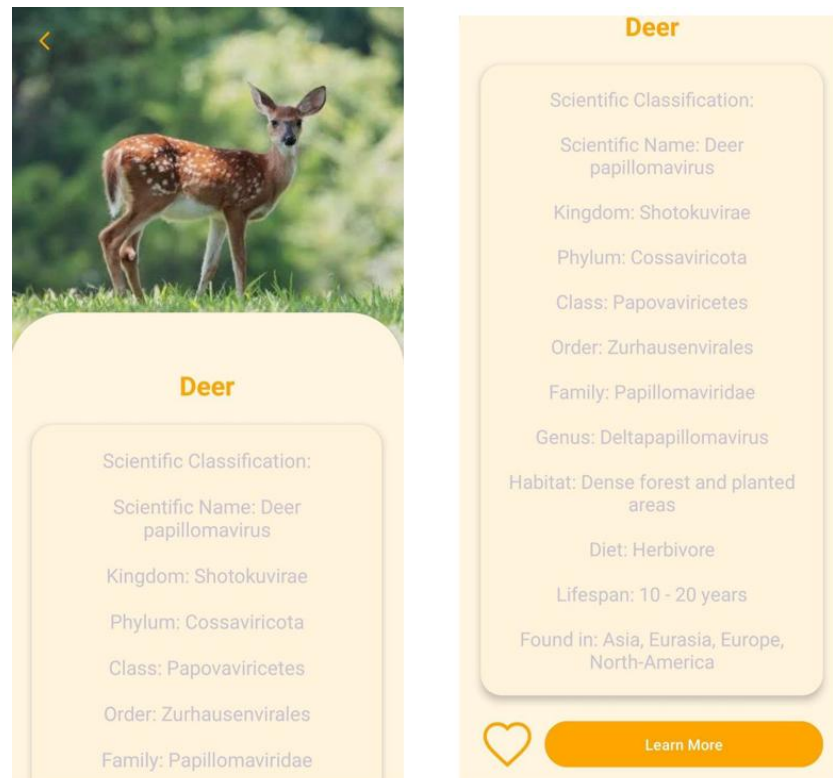


Figure 5.19c Animal Description Page

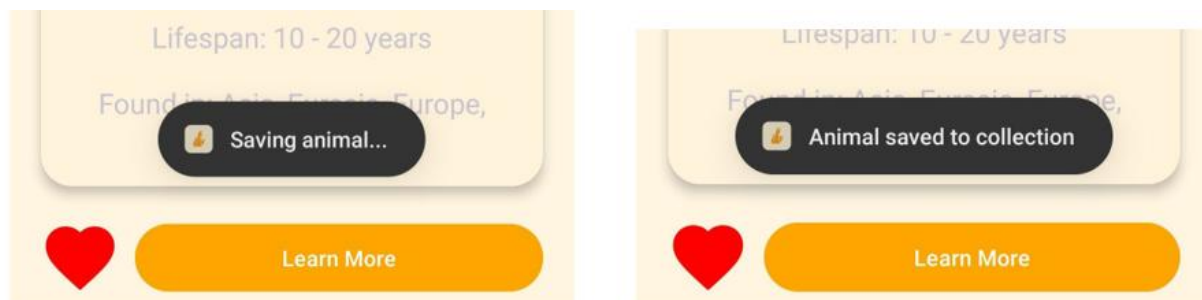


Figure 5.19d Save Animal

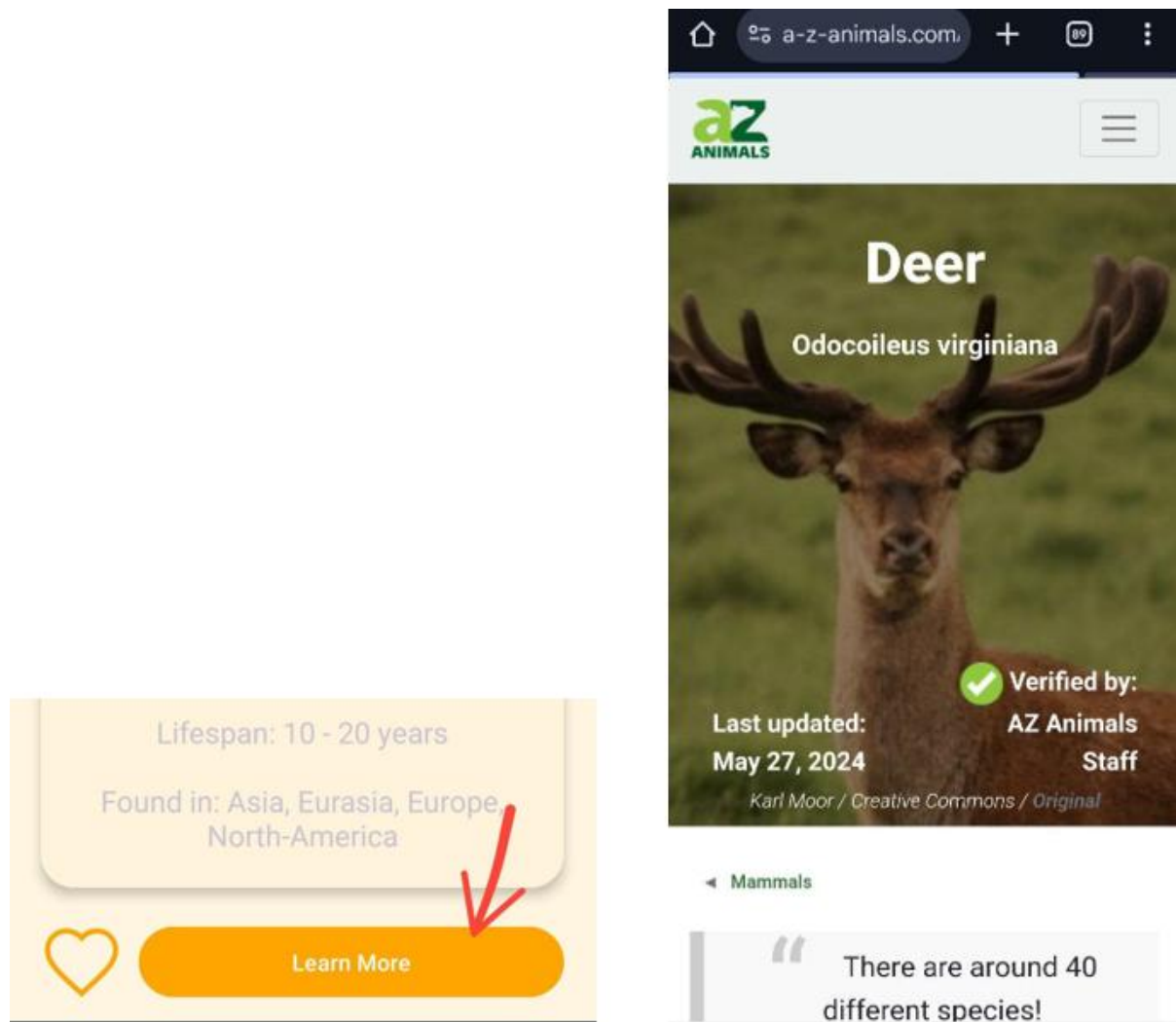


Figure 5.19e Learn More Button to A-Z Animal Website

5.4.6 Location Page

ResQMe applies GPS via FusedLocationProviderClient for fetching real-time coordinates and converting them into a common address by using the Geocoder algorithm. ResQMe system will ask permission for location and network before using the app. The user can choose to use real-time location tracking or manually type the address. Then, after calculating the distance, a list of clinics will be shown for review. If the user clicks the “Learn More” button, then they will navigate to the map page, which shows the route colored using Mapbox. Two buttons are provided, which are used to search for keywords on the website and navigate to use an external navigational app. Figures 5.20a to 5.20g show the processes mentioned.

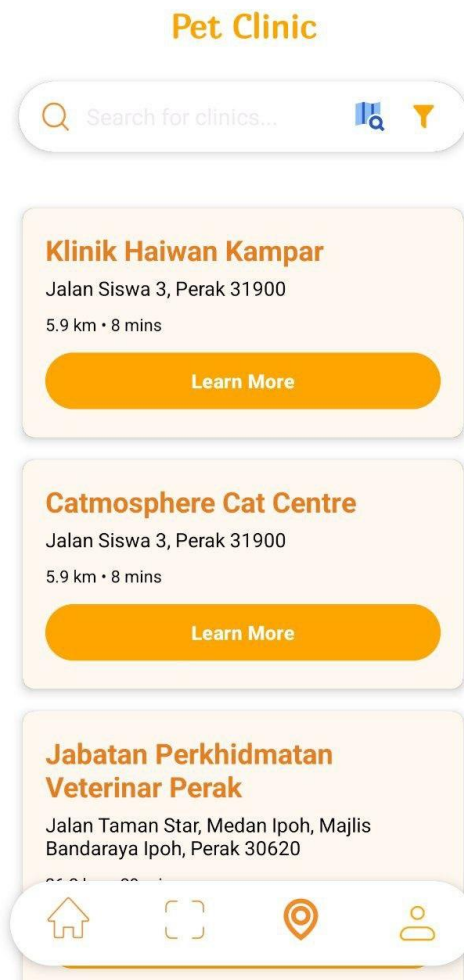


Figure 5.20a Location Page

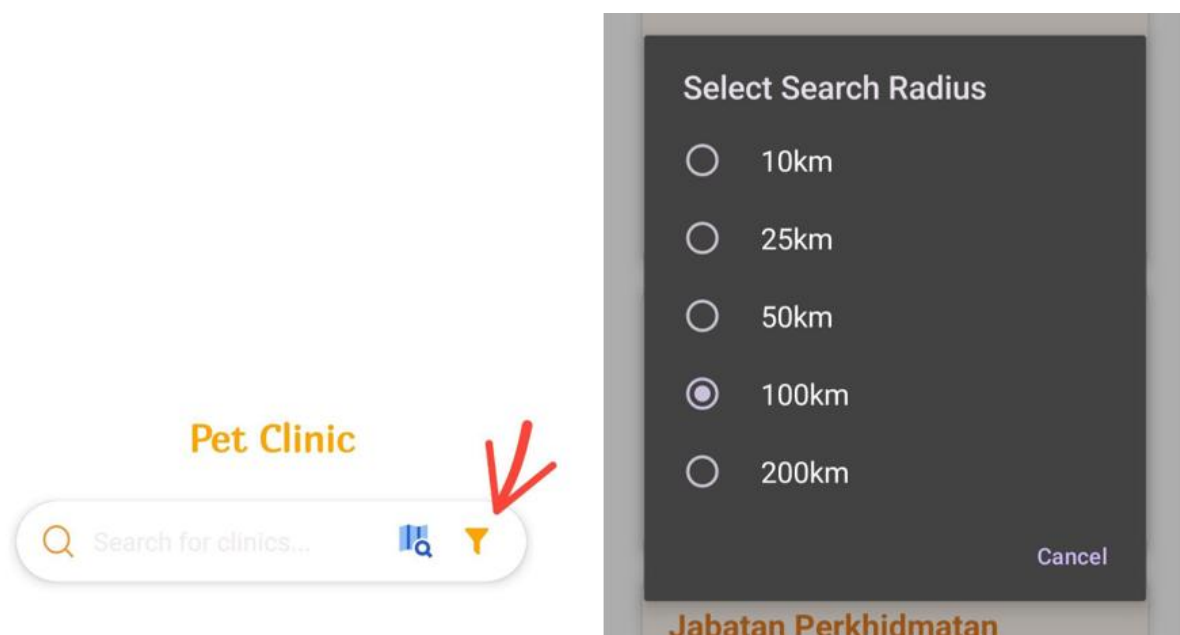


Figure 5.20b Filter Search Radius

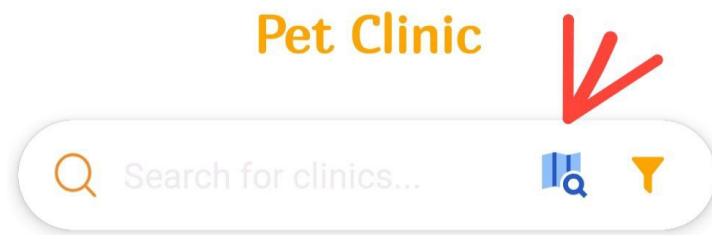


Figure 5.20c Input Location Manually

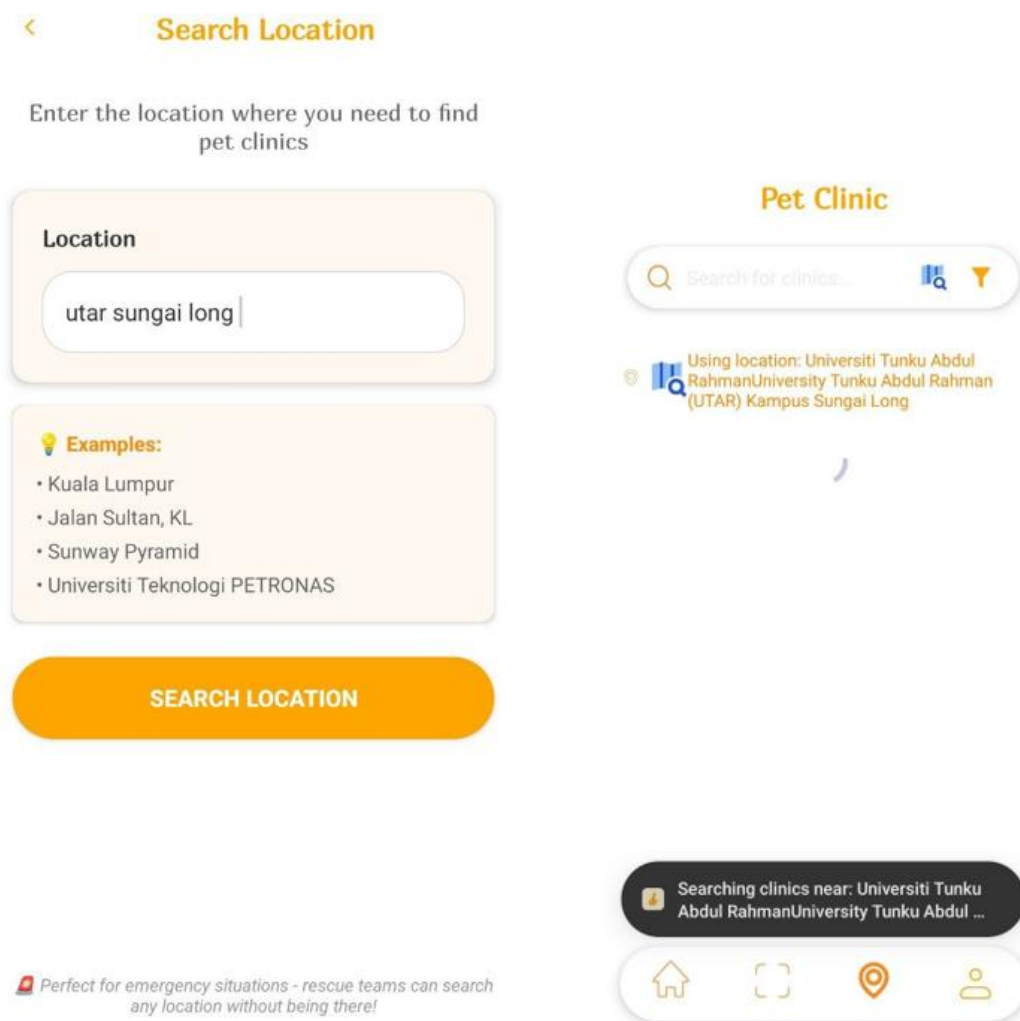


Figure 5.20d Location Searching via Input

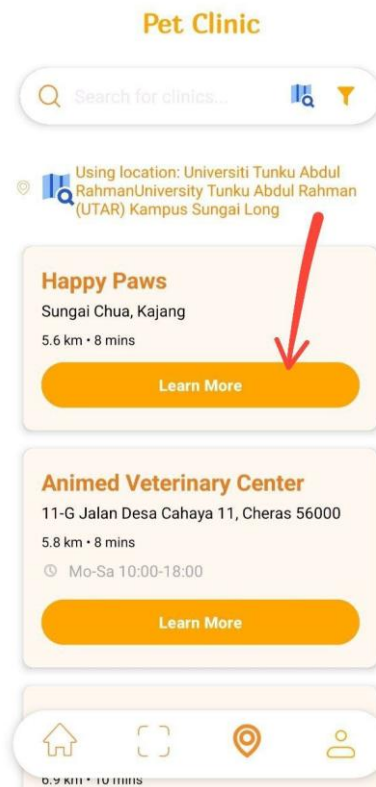


Figure 5.20e Shows All Clinics Nearby

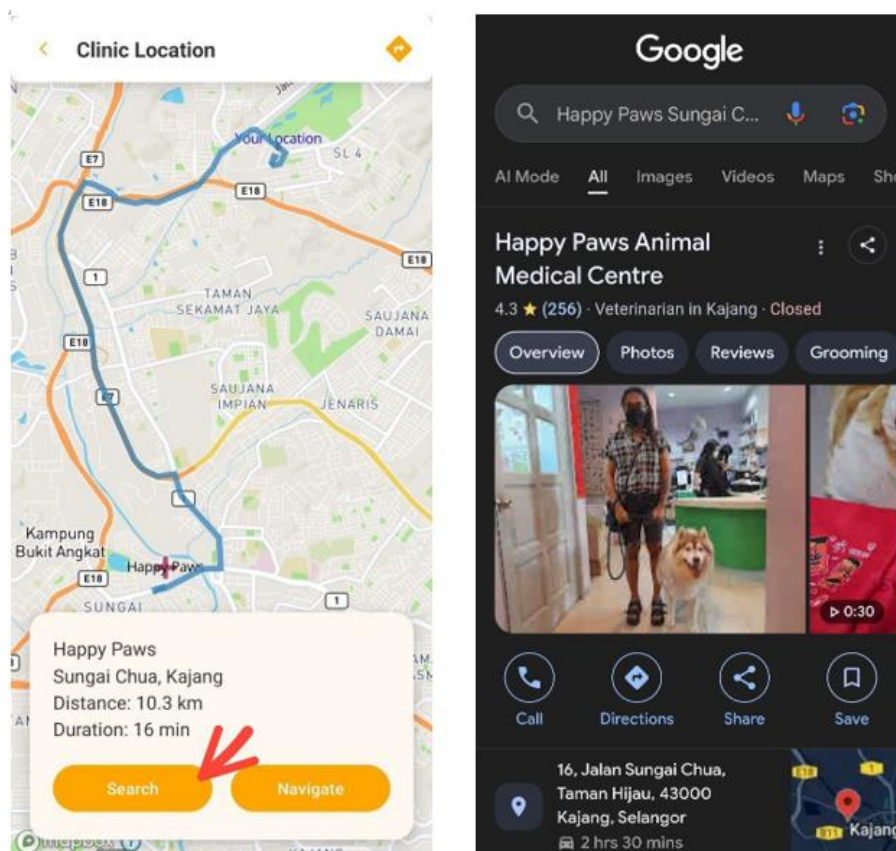


Figure 5.20f Search Button with Clinic's Name

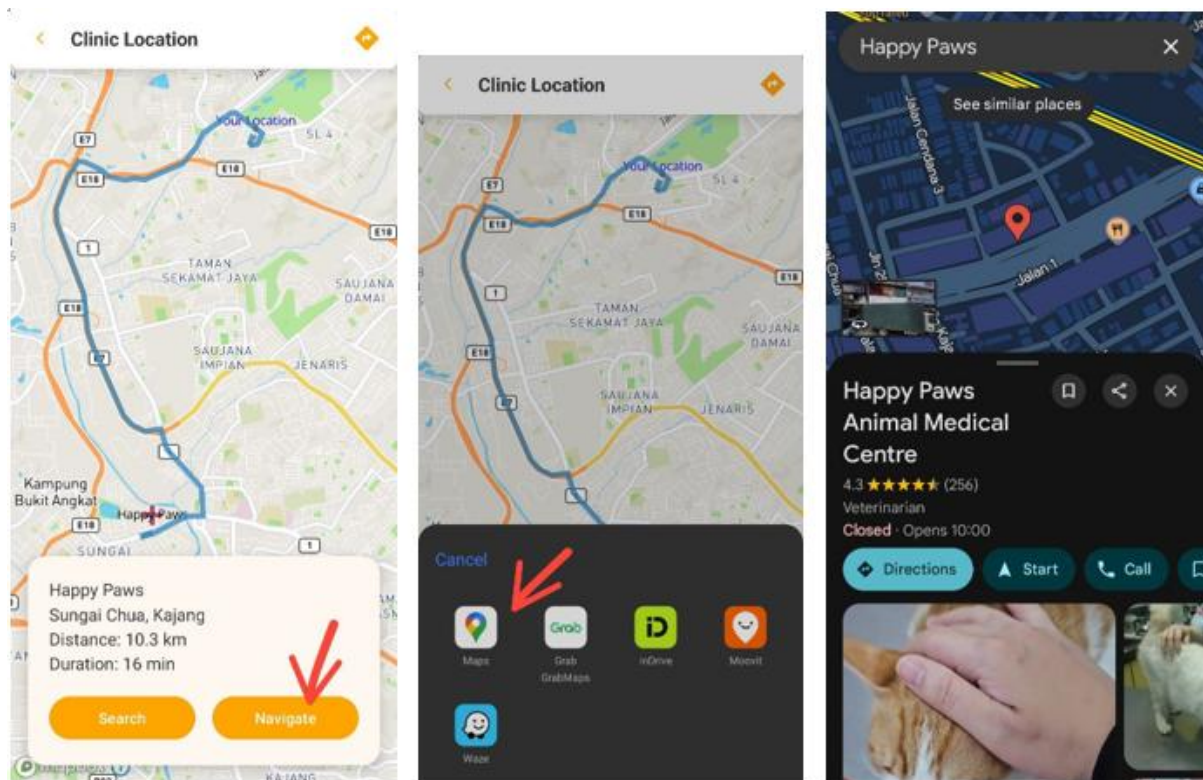


Figure 5.20g Navigate to Navigational Tool

5.4.7 Saved Animal Page

The saved animal page allows users to bookmark interesting animals, with information such as the animal's common name, scientific name, and the timestamp saved. The user can choose to delete the animal or enter the animal description to have a look. If the user enters the animal description page, the information will be provided similarly to the camera page. The “Learn More” button will navigate to the A-Z Animal website.

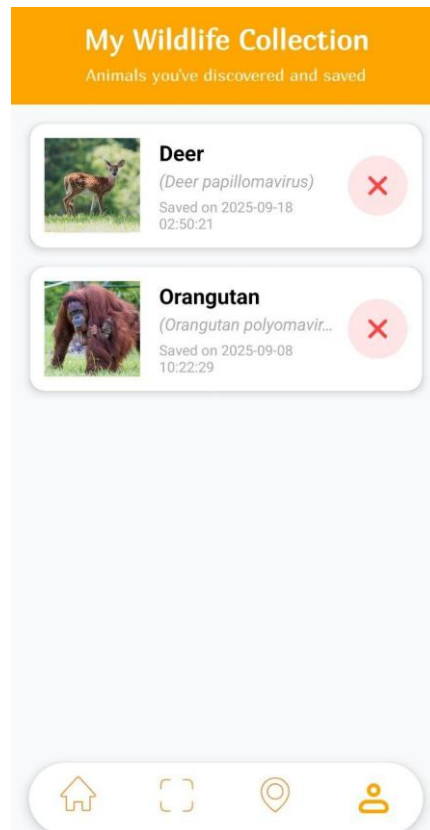


Figure 5.21a Saved Animal Page

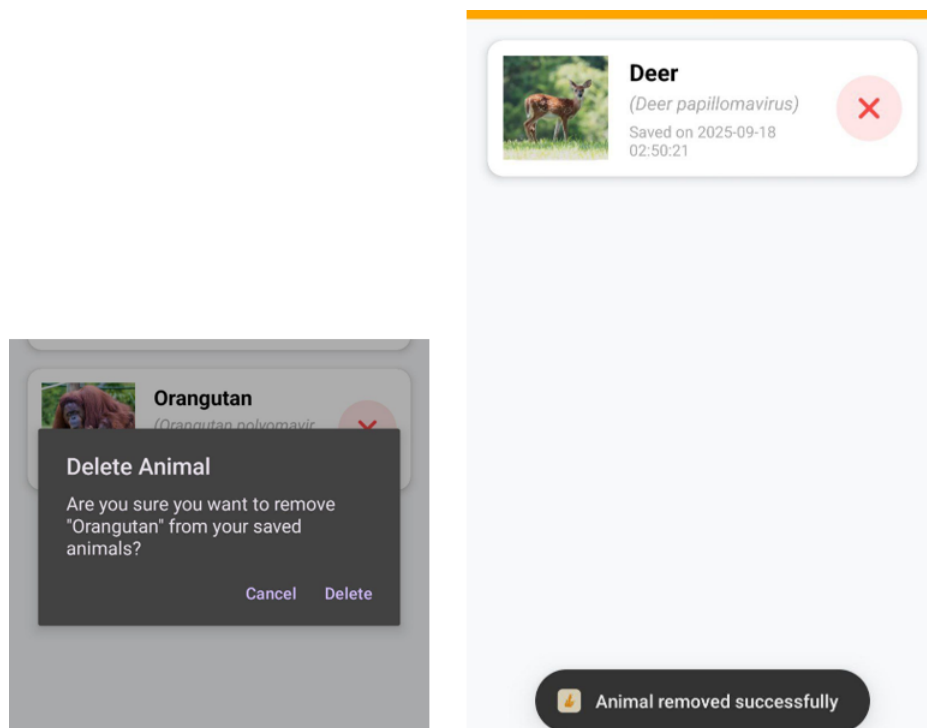


Figure 5.21b Deletion of Animal List

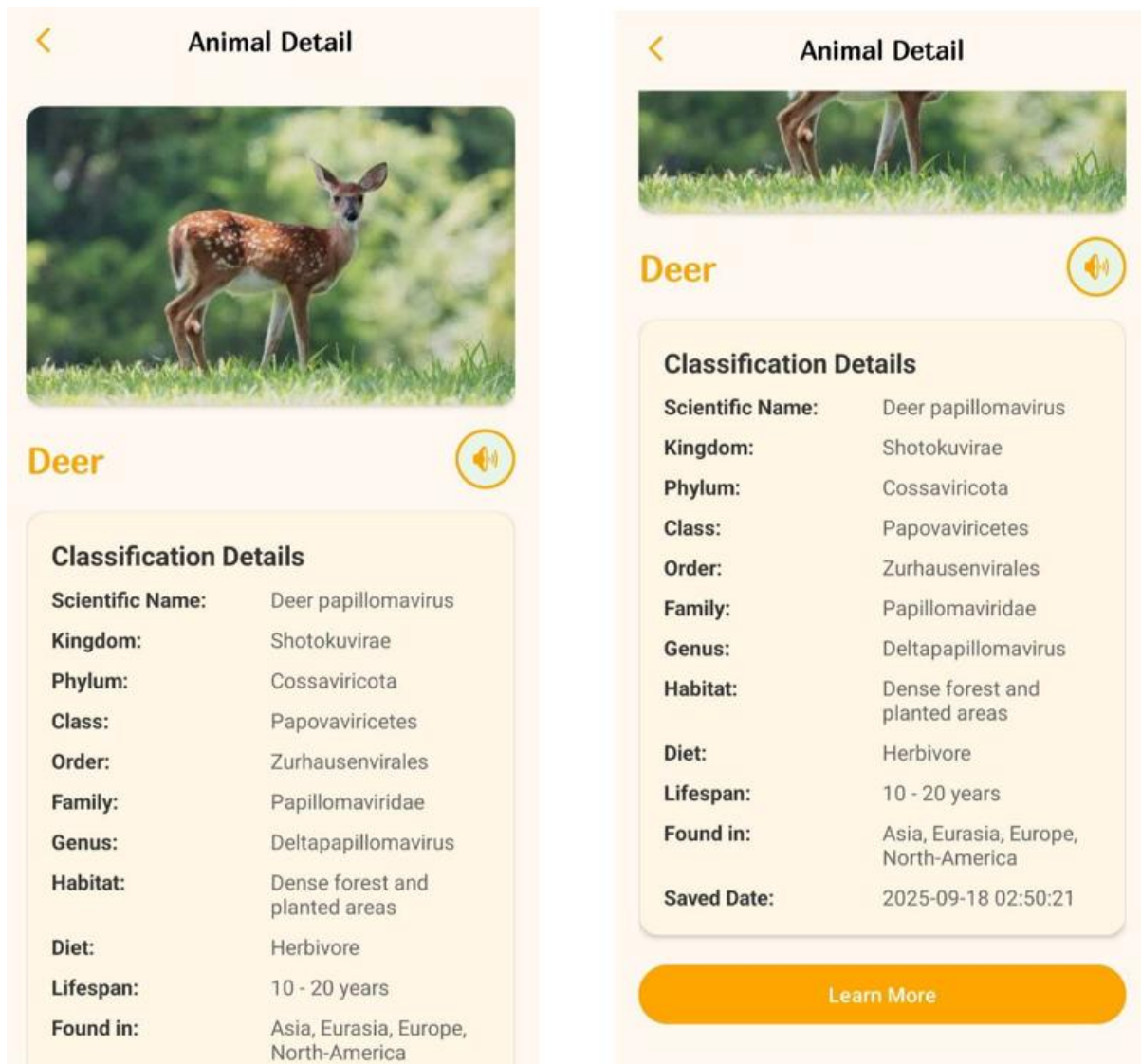


Figure 5.21c Animal Detail Page in Saved Animal Page

5.4.8 Admin Dashboard Page

Based on Figure 5.22, the admin dashboard page is an extra page different from the user's page; it stores information that only the admin can access, such as pending approval, admin messages, and pending descriptions. It provides quick access to an extra page for approval.

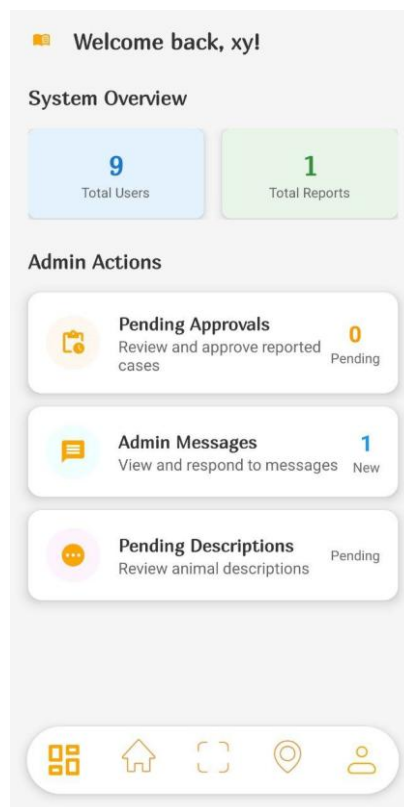


Figure 5.22 Admin Dashboard Page

5.4.9 ResQMe

ResQMe is the name of this mobile application, with a suitable logo image and name for animal rescue operations, as shown in Figure 5.23.



Figure 5.23 ResQMe Logo

5.5 Implementation Issues and Challenges

During the development of ResQMe, an animal rescue mobile application, various technical and non-technical issues were encountered. One of the challenges is **integrating Firebase and handling errors**. Initially, my lack of experience with Firebase prompted me to watch tutorials online and make some common mistakes. Then, Firebase does not always respond correctly with the ResQMe system in Android Studio, and the data inconsistency causes the system to fail. Inconsistency of real-time updates in notifications, report and description submissions, and volunteer applications is also a primary issue in completing the mobile application. For example, after the user submits a report, the notification should show with the content of “report submitted successfully” for the user and “received a report submission from the user” for the admin. However, the notification was only sent to users; the admin did not receive anything. To overcome the issue, the rules should be modified in Firestore.

Additionally, **OCR accuracy for Malaysian IC verification is low**, as it often detects the location as the name instead of the user’s full real name. Therefore, according to the format for Malaysian IC, the full name always appears on the location address. To resolve this issue, modify the OCR to detect the text over the location address and extract the actual names of users.

One of the challenges for developing ResQMe is **integrating Mapbox with Android Studio**. Conflicts with several errors always occur since the version incompatibility between Mapbox and Android Studio causes some features to be integrated incorrectly. It is difficult to start up the Gradle dependencies with the Mapbox access token. After carefully reviewing the Mapbox documentation and configuration for the access token, the problem can be solved.

It is always **difficult to create a clean and neat UI design** to enhance user experience. This requires color adjustment to make the content eye-catching. To solve this problem, feedback from other users is important to have an idea for improvement.

5.6 Concluding Remark

In conclusion, ResQMe, an intelligent animal rescue mobile application, has successfully overcome the issues and challenges with the operation of its core components. The ResQMe mobile application integrates several features such as animal recognition, location tracking, educational resources, form submission flow, AI-based built-in tools, real-time notifications, and OCR text extraction. The development process mentions the importance of balancing innovations such as OCR and AI-based models with real-world scenario practice.

Chapter 6

SYSTEM EVALUATION AND DISCUSSION

To ensure the completeness of ResQMe, the functional and non-functional tests should be conducted throughout the development cycle. Functional testing aims to verify that each of the modules can perform its expected output, such as the user authentication process, the flow for each page, button triggering, and navigation to other external applications. Non-functional testing involves performance of the system, such as response time and stability.

6.1 System Testing and Performance Metrics

This project applied black-box testing, which focuses on each feature's input-output behavior without internal code consideration. Firebase operations processes CRUD operations, session management, and real-time updates for Firebase authentication, Firestore, and message notifications.

6.1.1 User Login and Sign Up Module

The user login and sign up module was tested by submitting empty or invalid credentials, as shown in Figure 6.1a. The ResQMe system will detect input after the user submits the forms and display error messages to let the user fill in the blanks. In the login form, the user should fill in the email or password before accessing the system. In the sign up form, the user should fill in all information, such as IC photo, username, phone number, email, and password. This step is important to protect the reliability of Firebase Authentication securely. From Figure 6.1b, an example of testing a user leaving the IC photo section blank is given.

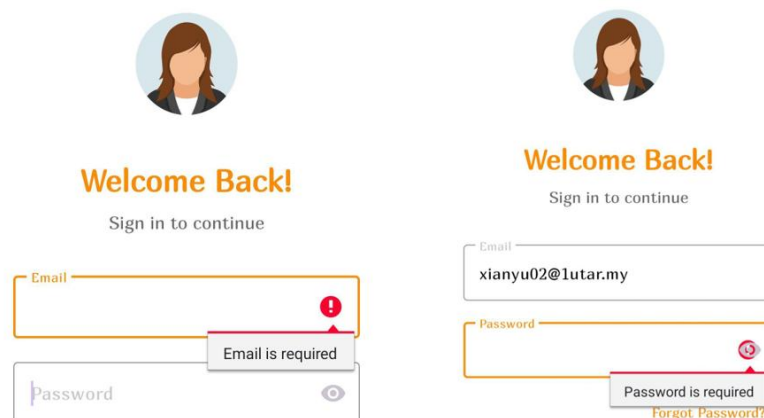


Figure 6.1a Error Messages for Blank Information in User Login

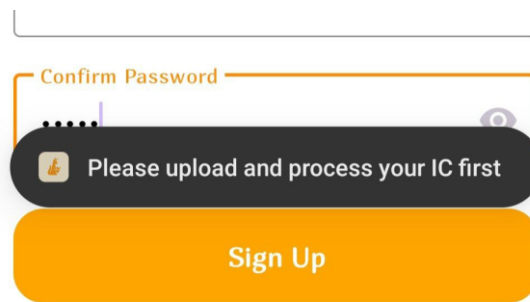


Figure 6.1b Error Messages for User Sign Up Page

In the user login page, error messages show if the user enters an invalid email or an incorrect password, as shown in Figure 6.1b. For example, a user types in the wrong format for email address without the symbol “@”, “gmail/1utar”, and “. my/.com” will be considered invalid input and rejected by the system.

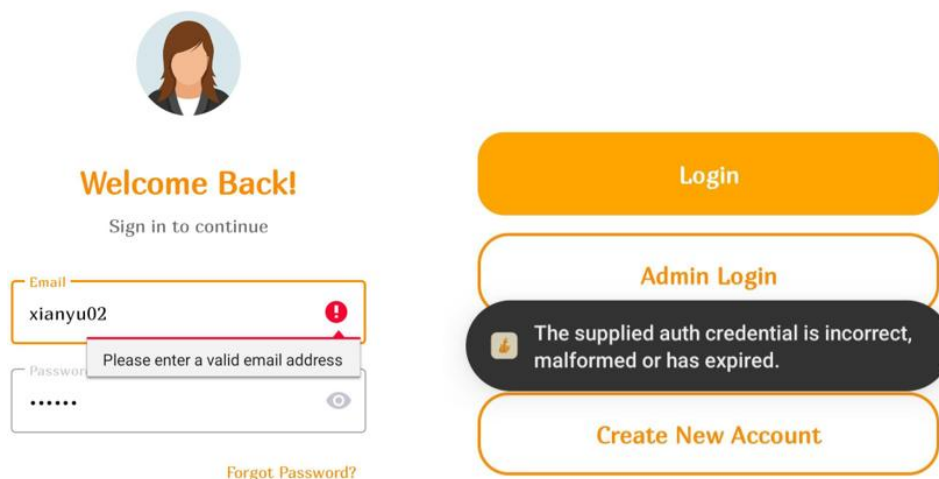


Figure 6.1c Invalid Email Address and Password

In the user sign-up page, some errors are tested when the user uploads an unclear or unrelated photo, an invalid email, or an inconsistent password. An example scenario is given as Figure 6.1d, a user uploads an unrelated photo. The purpose of this testing is to ensure that the user’s information is completed; sometimes this information can be used in other pages, such as submission auto-filled information, and volunteer applications with authenticated information sent back to the admin.

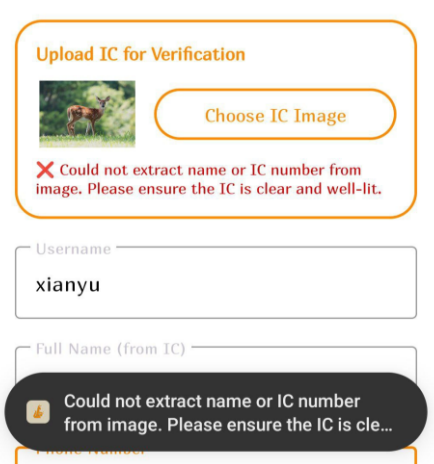


Figure 6.1d Error Message for IC Photo

6.1.2 Admin Login and Sign Up Module

The validation of an admin is stricter than that of a user, since admins have privileges to handle the approval of submissions and applications. Admin requires Firebase credentials and access code from Firestore during login and sign up.

The submission of wrong access codes, empty forms, and invalid email addresses was tested in both the login and sign up pages. In the admin sign up page, if the admin enters an inconsistent password, the error handling triggers, and the system shows an error message.

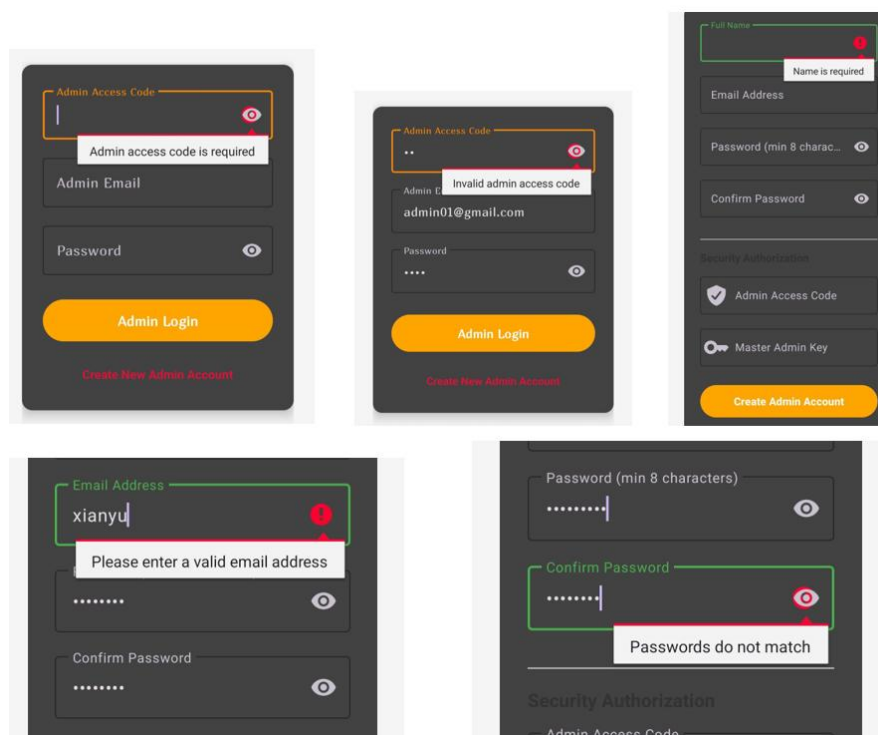


Figure 6.2 Testing for Admin Login and Sign Up Page

6.1.3 Report Submission Module

The report submission module ensures that the form validation, dropdown menu, and action buttons function properly. Figure 6.3a, Figure 6.3b, and Figure 6.3c show the testing process and its outputs. Error prompts triggered by submission of empty form, while the status of valid submitted forms stored in Firebase is “Pending”.

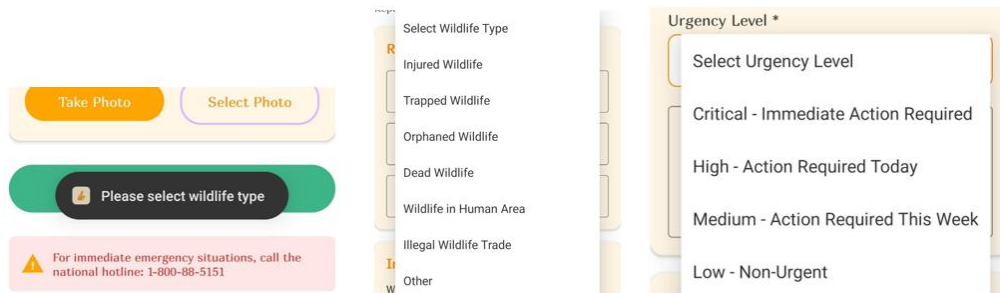


Figure 6.3a Blank Wildlife Type and Dropdowns Testing

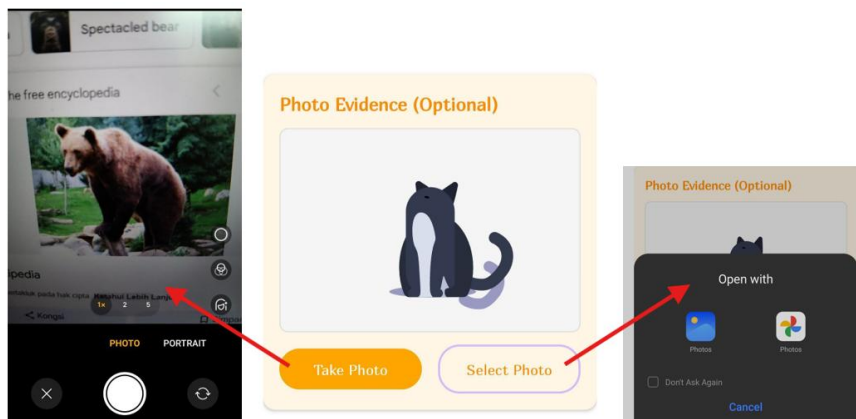


Figure 6.3b Take Photo and Select Photo Buttons Testing

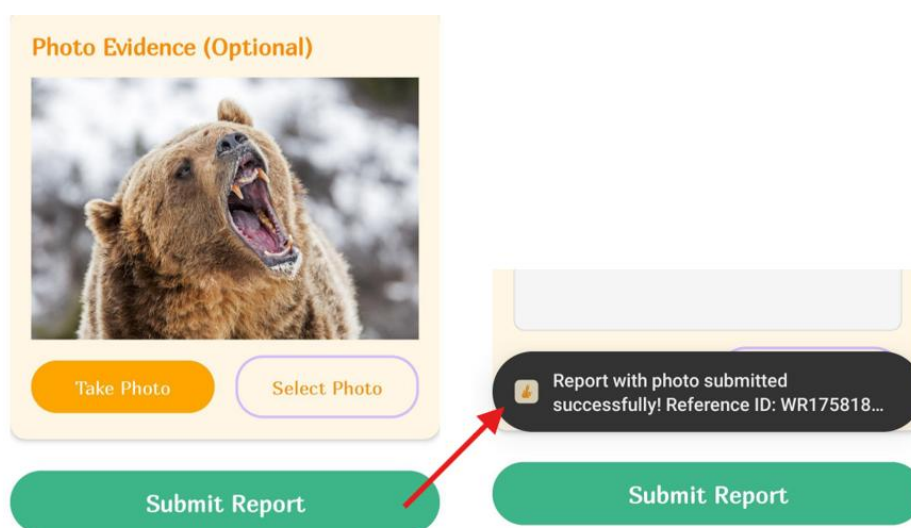


Figure 6.3c Submit Report Button Testing

6.1.4 Volunteer Application Module

Volunteer Application Module tested with several flows, which include button triggers, form application, and duplicated submission. Figure 6.4a and Figure 6.4b shows the testing results of buttons triggered, alert messages for application submission, and alert messages for duplicate submission.

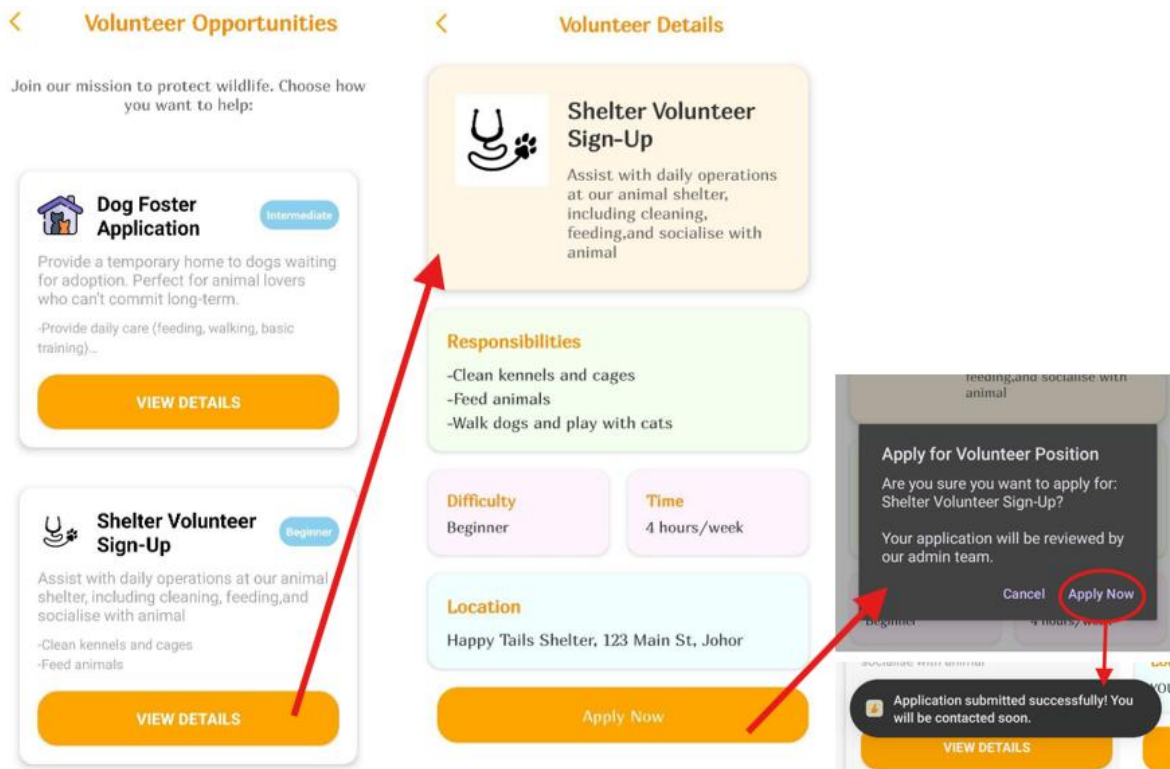


Figure 6.4a Volunteer Application Module Testing

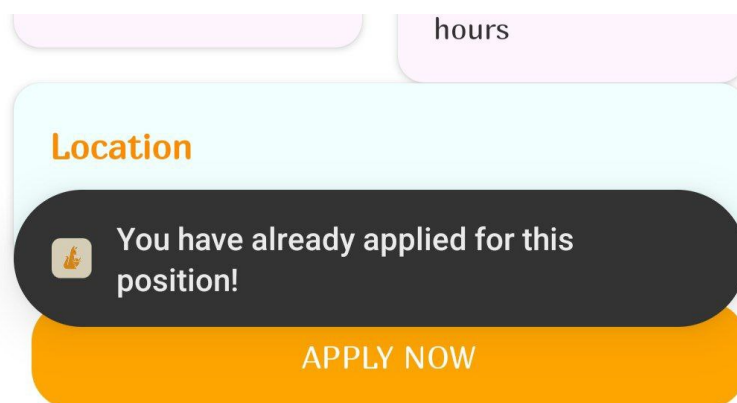


Figure 6.4b Duplicate Application Testing

6.1.5 Categories and Educational Resources Module

The categories section on the home page shows various types of animals. After the button of the categories box is triggered, the user will navigate to the animal type page that they selected, as shown in Figure 6.5a. The testing process was done after the button was clicked, and the search functions well on this page.

Then, if the user clicks on one of the animals, they will navigate to the animal details page. The testing process on this page involved Text-to-Speech (TTS) pronunciation, conversation with an AI-based Gemini chatbot, and button triggers. Figure 6.5b, Figure 6.5c, and Figure 6.5d tested search functionality, buttons trigger, Gemini AI conversation, and submission of description.

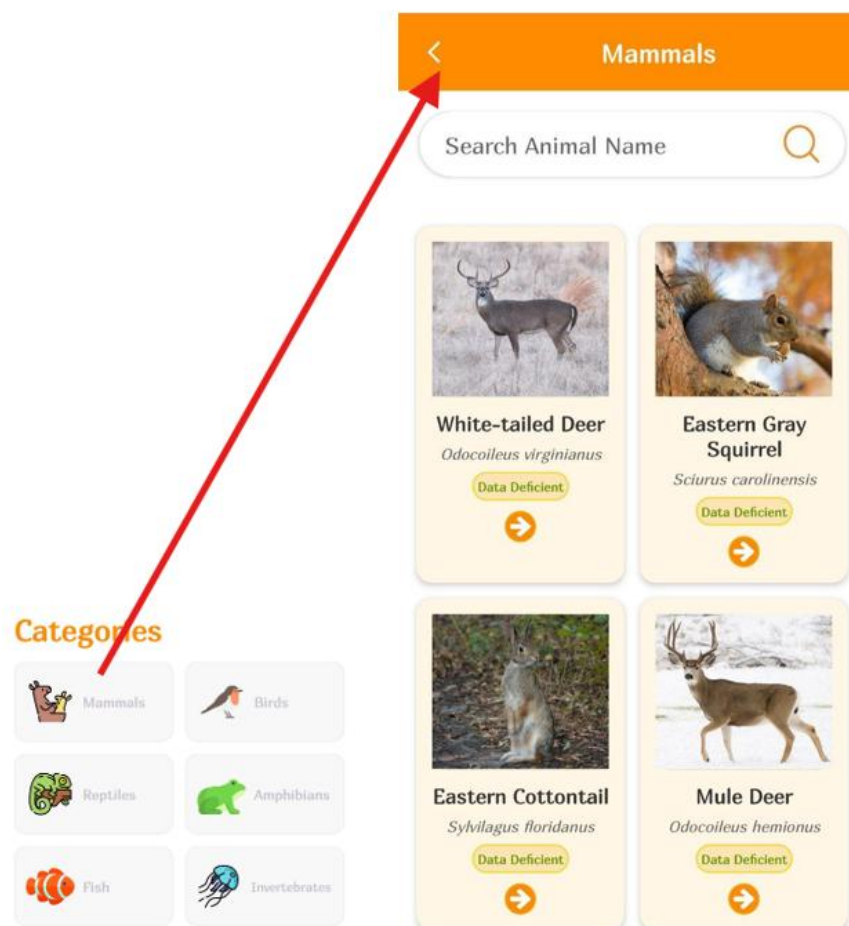


Figure 6.5a Categories Testing

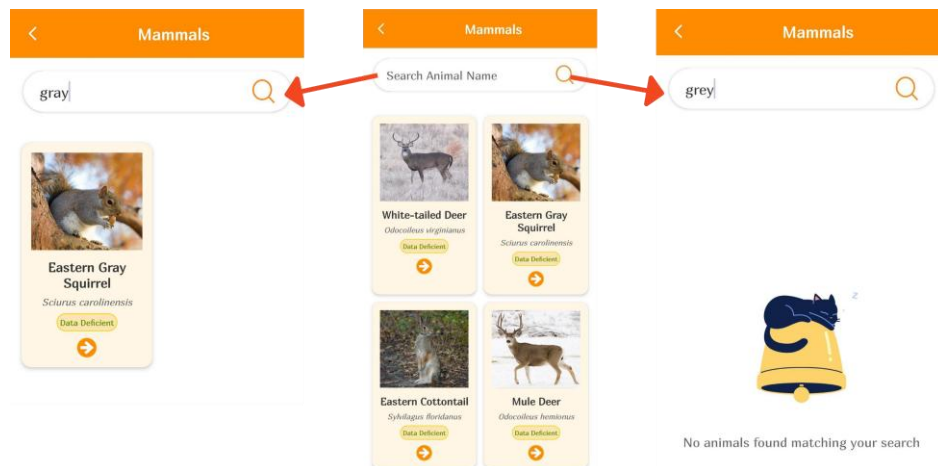


Figure 6.5b Search Functionality Educational Resources Testing

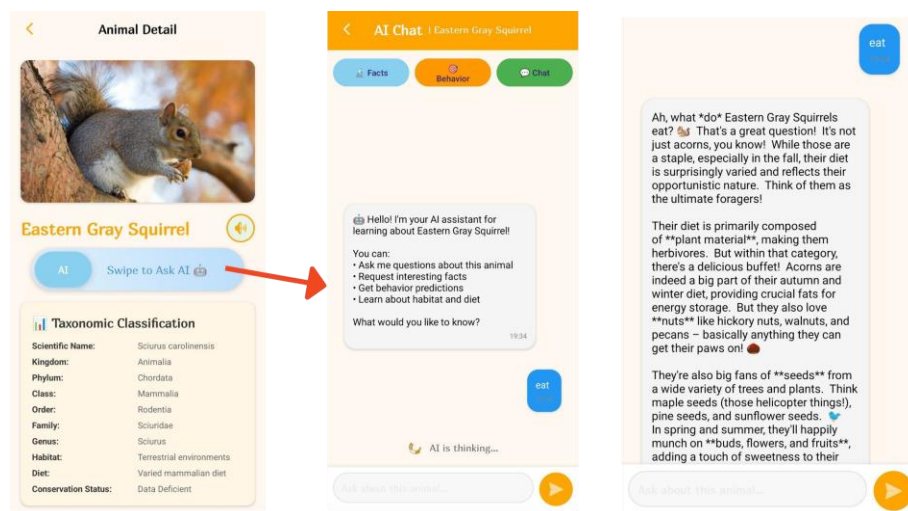


Figure 6.5c Gemini AI Testing

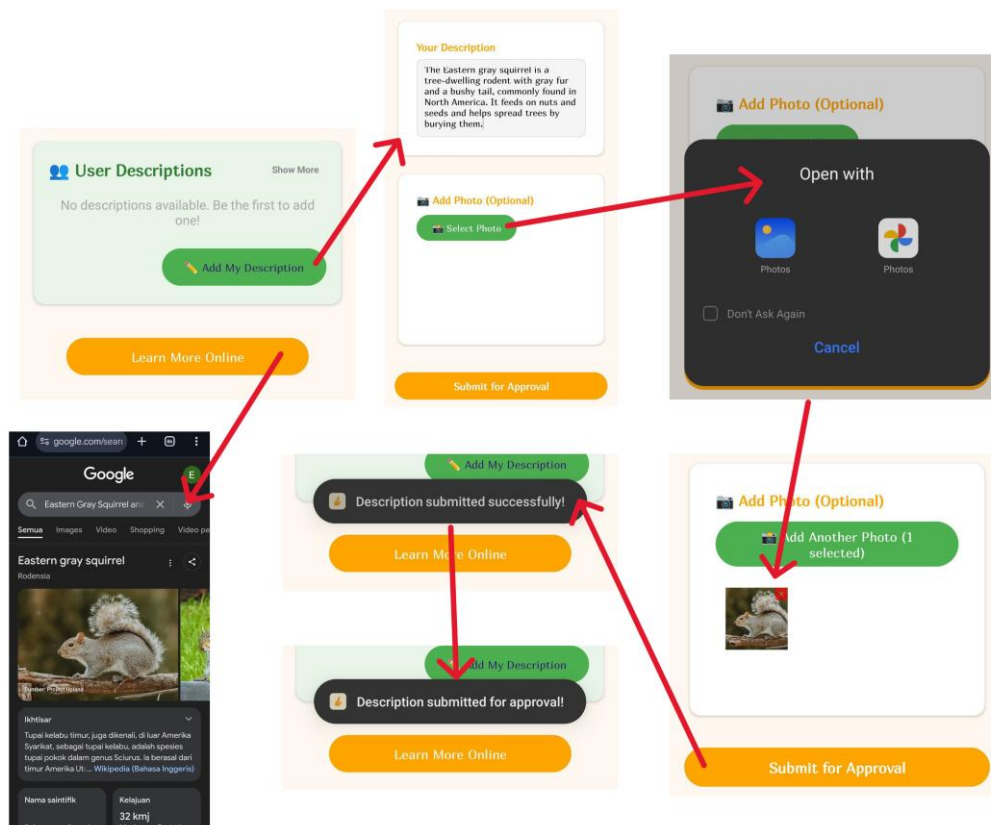


Figure 6.5d Add Description Testing

6.1.6 Admin Approval and Messages Module

The admin approval module was tested in approving or rejecting the submission of forms, volunteer applications, and descriptions. Approval process update in real-time in Firebase, with notifications for both admin and users' reminders by using NotificationHelper. The status also changed one action from admin done.

From Figure 6.6a to Figure 6.6e, the admin receives notifications and approves the submissions of reports, volunteer applications, and descriptions. After the action is done, the status will be changed. Figure 6.7a to Figure 6.7c shows the rejection process from the admin, like the approving process. Figure 6.8a and Figure 6.8b show users receive notifications and status updates in real-time once the admin takes action.

Admin approves:

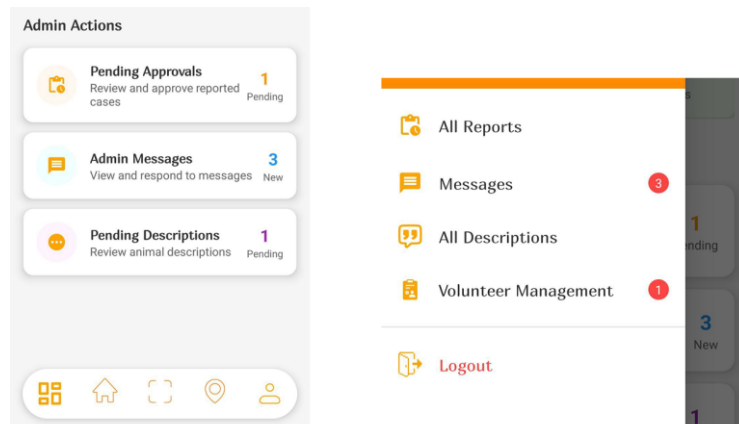


Figure 6.6a Notification from Admin Dashboard and Drawer Testing



Figure 6.6b Messages from Submissions

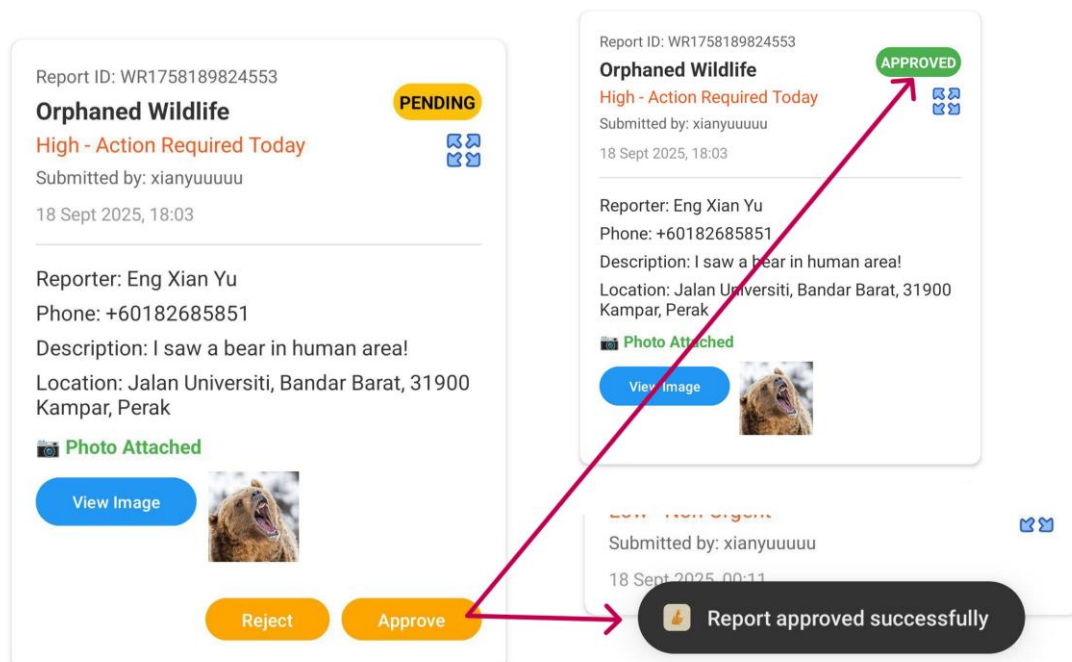


Figure 6.6c Admin Approve Report Testing

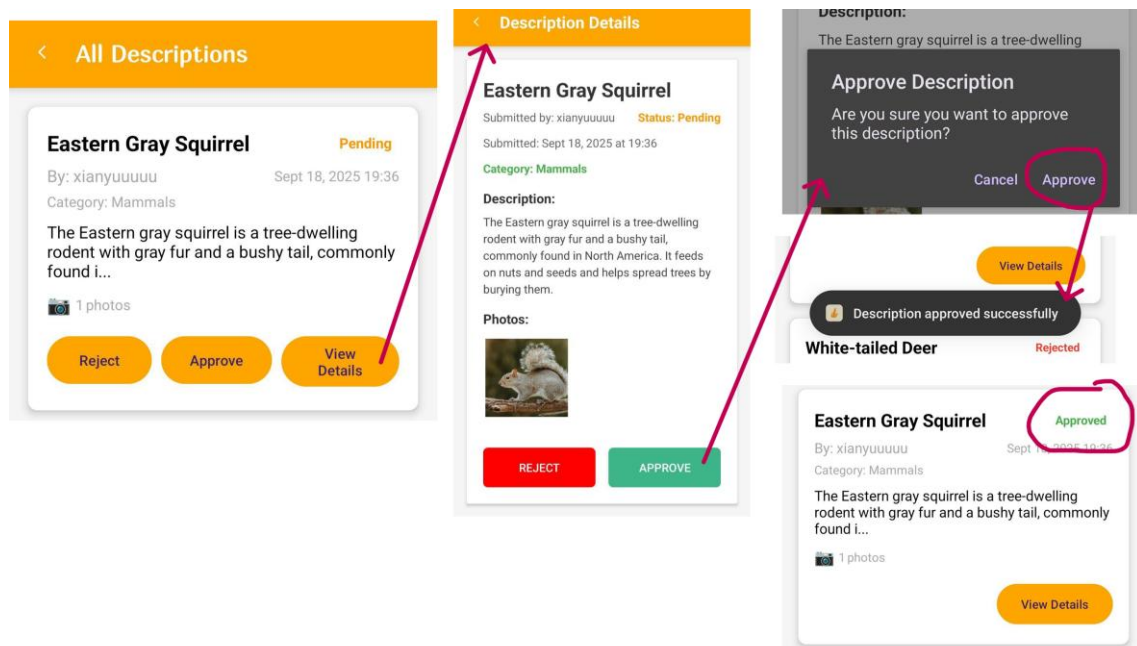


Figure 6.6d Admin Approve Description Testing

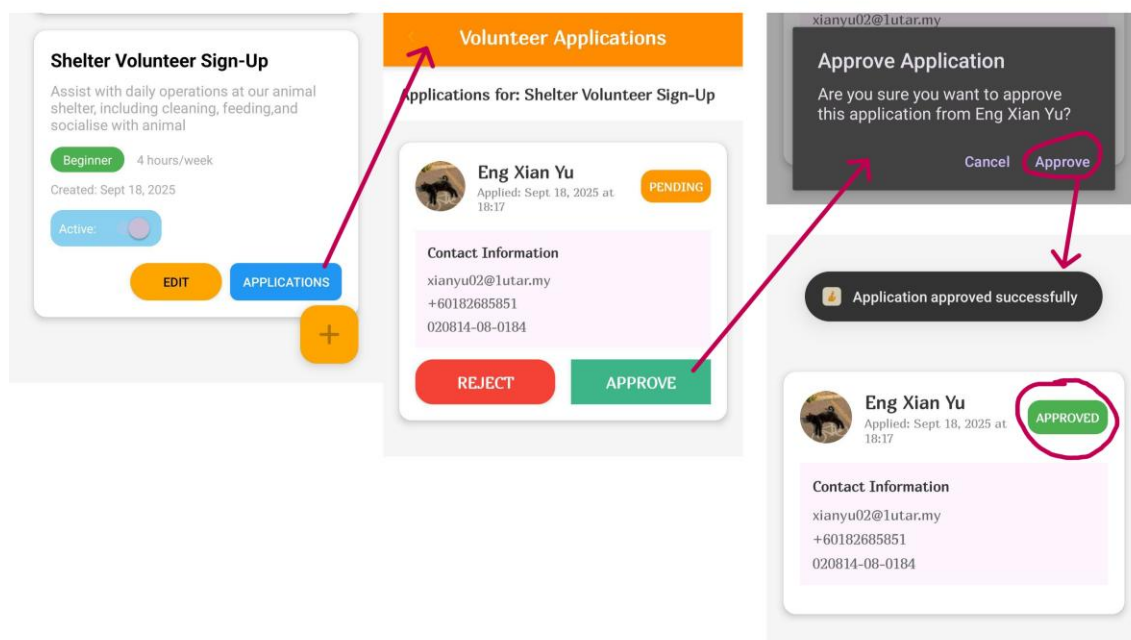


Figure 6.6e Admin Approve Volunteer Application Testing

CHAPTER 6

Admin Rejects:

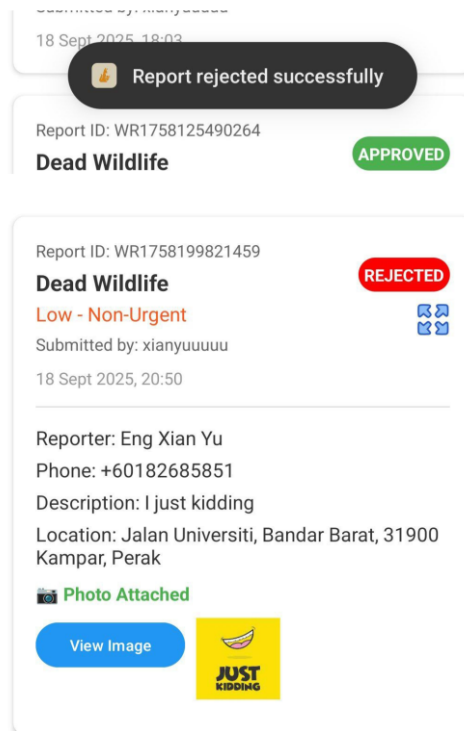


Figure 6.7a Admin Reject Report Testing

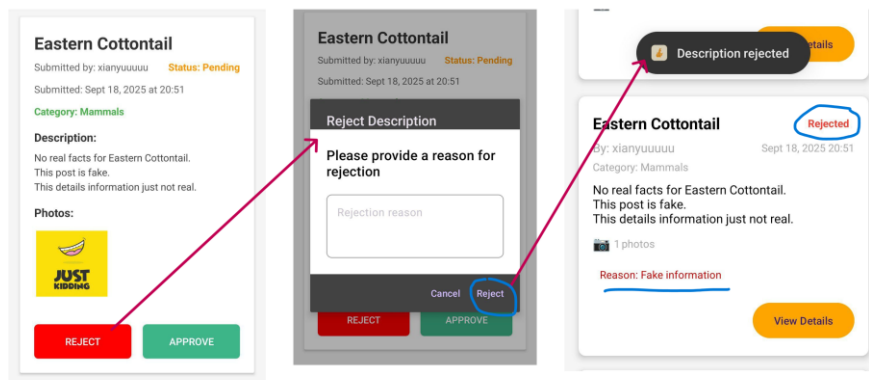


Figure 6.7b Admin Reject Description Testing

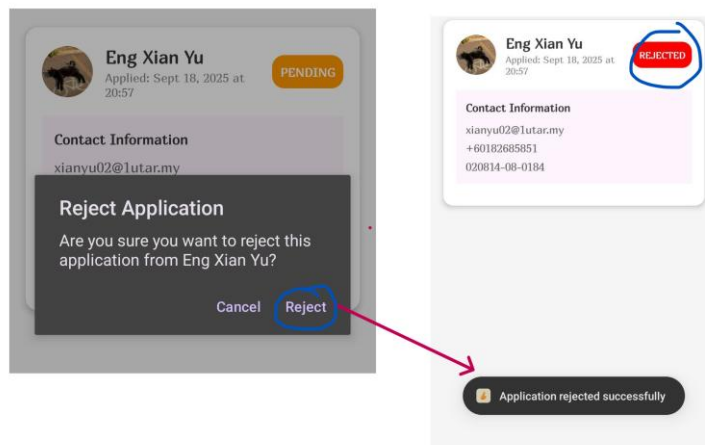


Figure 6.7c Admin Reject Volunteer Application Testing

User received notifications and status updates:

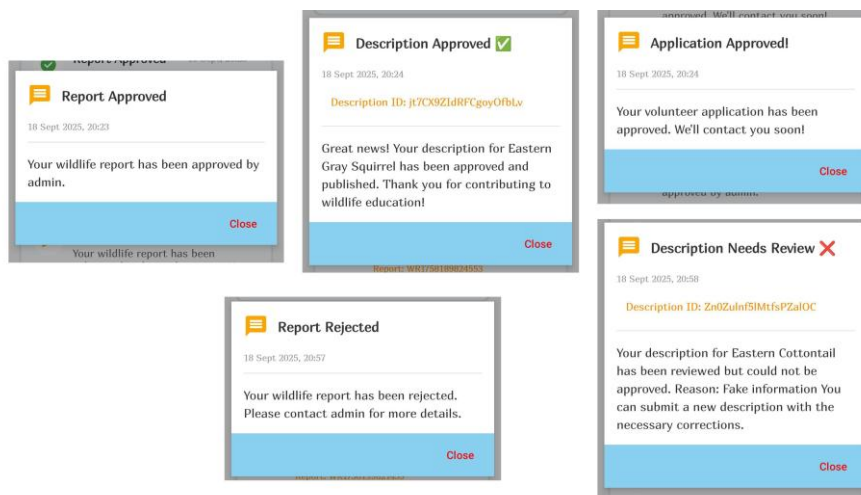


Figure 6.8a Notifications and Messages Testing

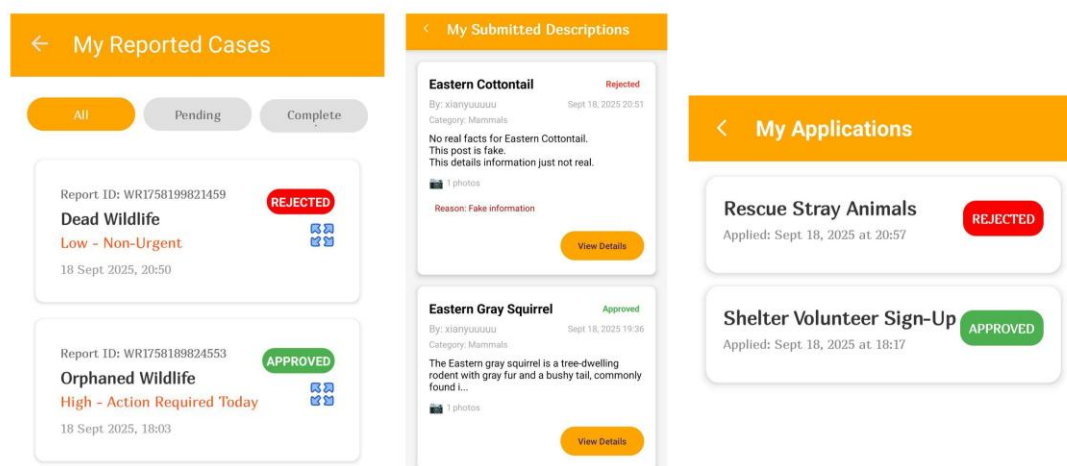


Figure 6.8b Submissions Status Update Testing

6.1.7 Profile Setting Module

Users can either change their username or profile photo in profile settings. The testing process involves button triggering and changes the picture. Figure 6.9a shows the testing process to upload the photo. From Figure 6.9b, once changed successfully, the profile image or username will be updated either on the home page or in the drawer.

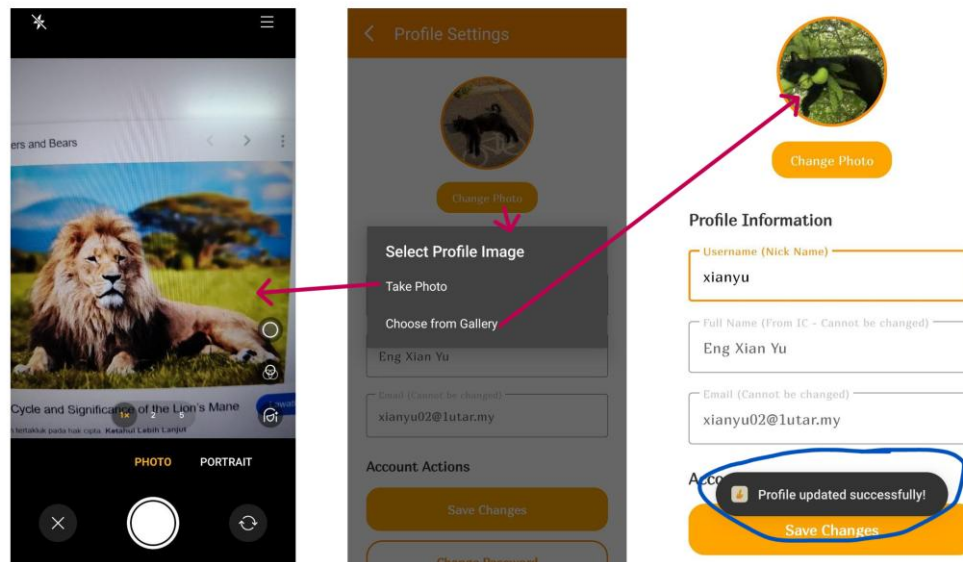


Figure 6.9a Profile Changing Testing

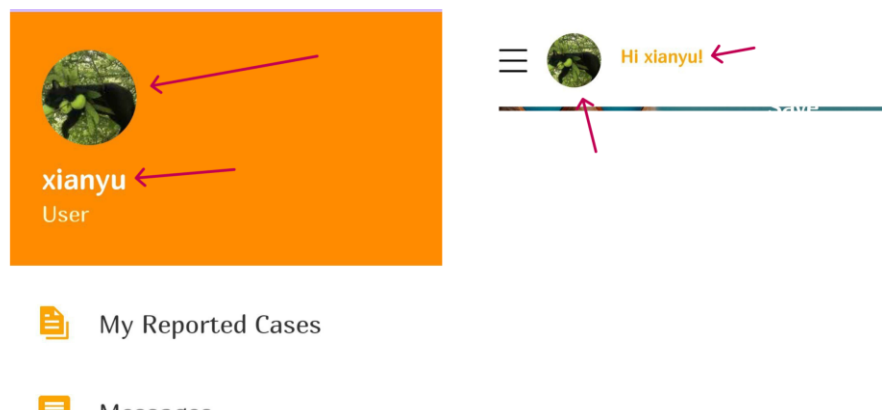


Figure 6.9b Profile Update Testing

6.1.8 News and Stories Module

This module's testing process includes button triggering, such as the "View More" button to access all news, the "Learn More" button to navigate inside the article, and the "Read Full Article" button to navigate between ResQMe and external sources as Figure 6.10 below.

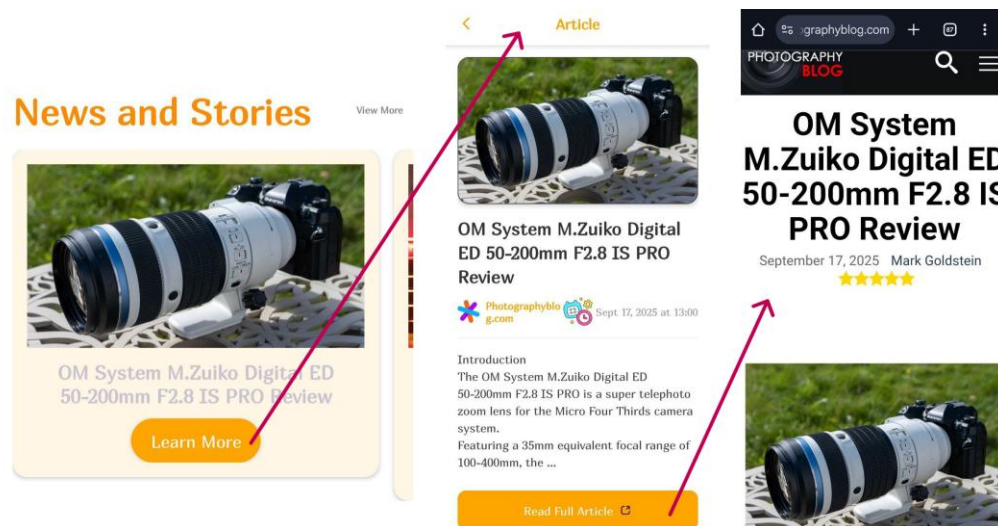


Figure 6.10 News and Stories Testing

6.1.9 Camera and Animal Description Module

This module's testing validates that either user can capture an image or select from the gallery. Button triggering in the animal description page and selecting the same image twice are tested. Based on Chapter 5, section 5.3.4 camera page, the testing showed such as photo uploads, button triggers, and animal recognition.

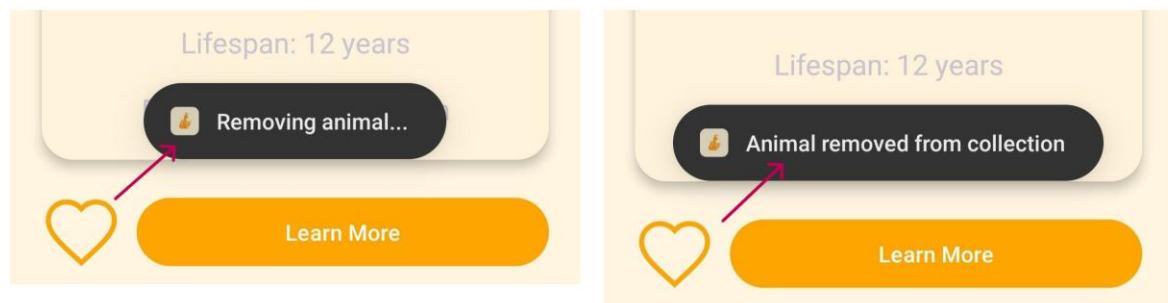


Figure 6.11 Remove Animal Testing

6.1.10 Location and Saved Animal Module

The location and saved animal modules tested their button triggering, which is shown in section 5.3.5 and section 5.3.6, respectively.

6.2 Testing Setup and Result

6.2.1 Authentication and Profile

Table 6.1 Authentication and Profile

Case ID	Use Case	Test Case	Test Steps	Expected Result	Actual Result	Status
TC1	User Login	Empty form	Leave email or password blank and submit.	Error messages display.	Error messages display.	Pass
TC2		Invalid Credentials	Enter the wrong email or password and submit	Access denied.	Access denied.	Pass
TC3	User Sign Up	Invalid IC photo	Upload an unclear IC and an unrelated picture.	OCR fails with an error prompt	OCR fails with an error prompt	Pass
TC4		Invalid email or phone	Enter an incorrect format email or phone number.	Error messages display.	Error messages display.	Pass
TC5		Password mismatch	Enter an inconsistent password field.	Error prompt shown.	Error prompt shown.	Pass
TC6	Admin Sign Up	Wrong access code or credentials	Submit invalid data information.	Access denied.	Access denied.	Pass
TC7		Wrong or empty information field	Submit an invalid email, a mismatched password, or a wrong code	Registration blocked.	Registration blocked.	Pass
TC8	Profile Settings	Change photo or username	Upload a new photo or username.	Data updated.	Data updated.	Pass

6.2.2 Reports and Volunteer Module

Table 6.2 Reports and Volunteer Module

Case ID	Use Case	Test Case	Test Steps	Expected Result	Actual Result	Status
TC1	Report Submission	Empty report	Submit the report without data.	Error messages display.	Error messages display.	Pass
TC2		Dropdown or button test	Select dropdowns and toggle buttons.	Dropdown expands; buttons function well.	Dropdown expands; buttons function well.	Pass
TC3		Valid report	Fill in the details and submit the report.	Report saved with “Pending” status.	Report saved with “Pending” status.	Pass
TC4	Volunteer Application	Apply Volunteer Application	Click “Apply Now”.	Application saved with “Pending” status.	Application saved with “Pending” status.	Pass
TC5		Duplicate application	Apply twice to the Volunteer Application.	Second attempt blocked	Second attempt blocked	Pass

6.2.3 Categories and Educational Resources Module

Table 6.3 Categories and Educational Resources Module

Case ID	Use Case	Test Case	Test Steps	Expected Result	Actual Result	Status
TC1	Categories	Click animal type	Click one of the animal categories.	Redirects to animal list.	Redirects to animal list.	Pass
TC2	Animal Details	Use features	Test TTS, AI chat, “Learn More” button, and “Show More” button.	Features trigger correctly	Features trigger correctly	Pass
TC3	User Description	Submit description	Submit an empty description	Error messages display.	Error messages display.	Pass
TC4			Submit a valid description.	Data saved to Firebase, waiting for approval.	Data saved to Firebase, waiting for approval.	Pass

6.2.4 Admin, Messages, and Notifications

Table 6.4 Admin, Messages, and Notifications

Case ID	Use Case	Test Case	Test Steps	Expected Result	Actual Result	Status
TC1	Admin Approval or Rejection	Admin approves submissions	Admin clicks the approve button.	Status updated and user notified via messages.	Status updated and user notified via messages.	Pass
TC2		Admin rejects submissions	Admin clicks the reject button.	Status updated and user notified via messages.	Status updated and user notified via messages.	Pass
TC3	Messages	Open messages	Both the admin and user open one of the messages in the list.	Message details show.	Message details show.	Pass
TC4		Close messages	Both the admin and the user close the message details	Message details closed.	Message details closed.	Pass

CHAPTER 6

6.2.5 News and Stories Module

Table 6.5 News and Stories Module

Case ID	Use Case	Test Case	Test Steps	Expected Result	Actual Result	Status
TC1	News and Stories	View all articles	Click the “View More” button.	All articles shown.	All articles shown.	Pass
TC2		View article’s details	Click the “Learn More” button.	Article’s details shown.	Article’s details shown.	Pass
TC3		View the article’s extra information	Click the “Read Full Article” button.	Direct to external source.	Direct to external source.	Pass

6.2.6 Camera Page Module

Table 6.6 Camera Page Module

Case ID	Use Case	Test Case	Test Steps	Expected Result	Actual Result	Status
TC1	Camera	Upload or capture a photo	Take a photo or upload from the gallery.	Navigate to the animal description page.	Navigate to the animal description page.	Pass
TC2	Animal Description	Animal Recognition	After uploading the photo, wait for the system detection.	An animal in the photo is recognized successfully.	An animal in the photo is recognized successfully.	Pass
TC3		Button toggle	Clicks the love icon or “Learn More” button.	An animal is saved in Firebase or navigates to	An animal is saved in Firebase or navigates	Pass

				an external website by keyword.	to an external website by keyword.	
--	--	--	--	---------------------------------	------------------------------------	--

6.2.7 Location Page Module

Table 6.7 Location Page Module

Case ID	Use Case	Test Case	Test Steps	Expected Result	Actual Result	Status
TC1	Location	Search for the nearest clinics	Applies real-time GPS tracking or manually enters an address.	Display clinics nearby.	Display clinics nearby.	Pass
TC2		Filter radius distance	Click the filter icon.	Display distances filtered by 10km, 25km, 50km, 100km, and 200km.	Display distances filtered by 10km, 25km, 50km, 100km, and 200km.	Pass
TC3	Map	Showing map	Clicks the “Learn More” button for the clinic on the location page.	Display a map with some information.	Display a map with some information.	Pass
TC4		Trigger buttons	Clicks the “Search” or “Navigate” button.	Direct to external website or navigational app.	Direct to external website or navigational app.	Pass

6.2.8 Saved Animal Page Module

Table 6.8 Saved Animal Page Module

Case ID	Use Case	Test Case	Test Steps	Expected Result	Actual Result	Status
TC1	Saved Animals	Manage list	Saves the animal in the animal description page.	Saved animal list display.	Saved animal list display.	Pass
TC2		Button triggering	Click the cross icon button to delete the saved animal.	Animals are deleted successfully.	Animals are deleted successfully.	Pass
TC3	Animal Detail	Present TTS (Text-to-Speech)	Click the sound icon beside the animal's name.	Animal name pronounced correctly.	Animal name pronounced correctly.	Pass
TC4		Button triggering	Click the "Learn More" button.	Navigate to the external website.	Navigate to the external website.	Pass

6.3 Project Challenges

The development of ResQMe encountered several challenges that affected the implementation process and timeline to complete the project. The **technical challenges**, like integration between different components, make it time-consuming. Real-time updates between different components add to their complexity, especially when role-based character admin and user changes occur. For example, at the beginning of the project, users submit forms that do not show the notification in the admin's messages module.

For data handling, the high demand for uploading images causes system lagging for a few seconds due to the limitation of Base24 storage. These challenges required an optimization process to prevent the ResQMe system slowdown and reduce the delay time.

The **practical challenge** in this project is time constraints. When training the custom animal recognition model, it takes several days to perform machine learning for handling dataset operations.

One of the non-technical challenges is the learning curve for integrating Firebase and AI components. Some tutorials are necessary to find on the website to have a better understanding of how to implement inside ResQMe. UX design is always a big challenge for developing a user-friendly interface provided to diverse users. Therefore, several design iterations need to handle system stability and simplify navigation.

6.4 Objectives Evaluation

The three objectives were set up at the beginning phase of the project to guide the development of ResQMe.

1. To Integrate CNN into the Animal Rescue System

This objective was successfully achieved in ResQMe, which integrated with a pre-trained MobileNetV2 model and a custom CNN model with a combination of datasets for animal recognition. ResQMe contains a camera fragment, which the CNN models implemented here by photo uploading. The user can select either to take a photo in camera view or select from the gallery. After selection, the system will process animal validation in a custom model, then turn to a pre-trained model if no dataset is found. The user will navigate to the animal description page after the animal recognition process, which shows animal information with the detected name.

2. To Provide Educational Resources Related to Animal Knowledge

The objective achieved in ResQMe, that the educational module allows users to browse the categories of animals, such as mammals, reptiles, fish, birds, amphibians, and invertebrates. Animals are listed based on their categories, and each animal's information is provided via several APIs, such as common and scientific name, kingdom, phylum, class, order, family, genus, habitat, and diet. Additional features such as text-to-speech pronunciation, AI-based conversation, and user-contributed descriptions enhance the innovation and usability of ResQMe.

3. To Implement Route Planning to Find the Shortest Route for Pet Clinics

ResQMe enables users to search for the nearest animal clinics by using real-time GPS tracking or inputting an address manually to achieve this objective. After distance calculation using the Haversine formula, the user can choose one of the clinics and navigate to the map page by using the Mapbox API. In the project testing phase, this navigational functionality can be successfully performed.

6.5 Concluding Remark

In summary, ResQMe reached these three objectives to increase its reliability, delivering a useful prototype that combines CNN animal recognition, educational resources, and route planning together with the help of an AI-based tools and APIs combination. The evaluation objectives phase demonstrated that ResQMe is a useful and innovative animal rescue platform. ResQMe provides some extra comprehensive features, including a wildlife rescue report, volunteer management, an AI chatbot, and real-time notifications.

Chapter 7

CONCLUSION AND RECOMMENDATION

7.1 Conclusion

ResQMe, an intelligent animal rescue platform application, aims to improve animal rescue operations and public awareness. The balancing of the ecosystem between humans and animals is important for environmental stability and biodiversity conservation. Therefore, ResQMe provides several novel contributions compared to existing applications, including CNN animal recognition models, educational resources, and real-time route tracking.

In testing and evaluation phase, ResQMe met the three objectives and showcased the functionality applicable, such as enabling submissions of forms, integration between admins and users, and providing AI-based conversation. This system leverages Firebase for user authentication, real-time interaction, database management, and notification to improve the user experience in a more secure way.

The CNN-based animal recognition models powered by machine learning and deep learning, which detect animal names through photo uploading. This valuable technology reduces the rescue times and accelerates the rescue operations. The extra educational resources across various animal categories gain users' awareness and knowledge. With the support of an integrated Gemini AI conversation, users can have a better understanding of that animal through real-time dialogue. Additionally, route planning with distance calculations shows the nearest animal clinics, which will reduce operation times by using real-time GPS tracking and powerful map review.

From a user's perspective, ResQMe provides a full rescue operation consideration, such as wildlife reporting, volunteer management, and route planning to clinics that are not included in other existing rescue applications. Login only needs to be taken when necessary actions, like submission forms, role-based access between admin and user, and saving animals into the database. This ensures that the rescue operation is not delayed by unnecessary identity

authentication. For example, users can quickly find the hotline number based on each state or chatbot conversation when they encounter an emergency case.

In conclusion, ResQMe demonstrated the combination of technologies in cloud databases and artificial intelligence that creates an impactful solution in animal rescue operations. This project achieves the three objectives following the system prototype, not only for necessary rescue operations but also for public awareness and knowledge. ResQMe has the potential to be developed into a larger-scale platform with future refinement.

7.2 Recommendation

For future enhancements, which can improve the usability and reliability of ResQMe, some steps can be taken, such as **enhancing offline capability** in Firebase, so users can use the location page even though they are in offline mode. This is because some volunteers or rescuers may be in an area with poor connectivity, which may delay the rescue operations if they continue waiting for the loading of the page.

Besides, **gamification and engagement** are two of the considerations for improvement. ResQMe is a mobile application platform designed for all age users; therefore, for the younger users, small games may be more appealing and enjoyable to them.

A community platform is always convenient for communicating and experience sharing with each other. ResQMe can make improvements with this feature. By extending ResQMe with this feature, users can share their knowledge and rescue stories with moral support for ongoing

REFERENCES

- [1] R. Chauhan, K. K. Ghanshala and R. Joshi, "Convolutional Neural Network (CNN) for Image Detection and Recognition," *IEEE Xplore*, pp. 278-282, 2018.
- [2] D. Ikasari, Widiastuti and R. Andika, "Determine the Shortest Path Problem Using Haversine Algorithm, A Case Study of SMA Zoning in Depok," *IEEE Xplore*, 2021.
- [3] N. Sharma, "What is MobileNetV2? Features, Architecture, Application and More," Analytics Vidhya, 26 November 2024. [Online]. Available: <https://www.analyticsvidhya.com/blog/2023/12/what-is-mobilenetv2/>.
- [4] "MissingPets," [Online]. Available: <https://missingpets-app.com/>. [Accessed 1 April 2025].
- [5] "Pawrpose," [Online]. Available: <https://pawrpose.org/>. [Accessed 1 April 2025].
- [6] P. K. R. O. M. B. a. P. S. T. Trnovszky, "Animal Recognition System Based on Convolutional Neural Network," *ResearchGate*, vol. 15, no. 3, pp. 517-525, 2017.
- [7] "Large Language Models powered by world-class Google AI," Google Cloud, [Online]. Available: <https://cloud.google.com/ai/llms?hl=en>. [Accessed June June 2025].
- [8] "Firebase," [Online]. Available: <https://firebase.google.com/>. [Accessed 3 June 2025].
- [9] "Android Jetpack," Android Studio, [Online]. Available: <https://developer.android.com/jetpack>. [Accessed 10 March 2025].
- [10] "ML Kit Analyzer," Android Studio, [Online]. Available: <https://developer.android.com/media/camera/camerax/mlkitanalyzer>. [Accessed 10 March 2025].
- [11] "Mapbox," Mapbox, [Online]. Available: <https://docs.mapbox.com/api/maps/>. [Accessed 15 June 2025].
- [12] R. Mittal and A. Garg, "Text extraction using OCR: A Systematic Review," *IEEE Xplore*, pp. 357-362, 2020.
- [13] "ML Kit," Google, [Online]. Available: <https://developers.google.com/ml-kit>. [Accessed 12 August 2025].
- [14] "Overpass API," [Online]. Available: https://wiki.openstreetmap.org/wiki/Overpass_API. [Accessed 3 July 2025].

REFERENCES

- [15] "Nominatim 5.1.0 Manual," GitHub, [Online]. Available: <https://nominatim.org/release-docs/latest/api/Reverse/>. [Accessed 4 July 2025].
- [16] "Maps SDK for Android," Mapbox, [Online]. Available: <https://docs.mapbox.com/android/maps/guides/>. [Accessed 3 July 2025].
- [17] "Waze," [Online]. Available: <https://www.waze.com/live-map>. [Accessed 4 July 2025].
- [18] "Google Map," Google, [Online]. Available: <https://www.google.com/maps>. [Accessed 4 July 2025].
- [19] "API Reference," iNaturalist, [Online]. Available: <https://www.inaturalist.org/pages/api+reference>. [Accessed 15 June 2025].
- [20] "GBIF API Reference," GBIF, [Online]. Available: <https://techdocs.gbif.org/en/openapi/>. [Accessed 15 July 2025].
- [21] "Open Tree of Life," [Online]. Available: <https://tree.opentreeoflife.org/opentree/argus/opentree15.1@ott93302>.
- [22] "Kaggle," [Online]. Available: <https://www.kaggle.com/>. [Accessed 4 April 2025].
- [23] "Android Studio," [Online]. Available: <https://developer.android.com/studio>. [Accessed 20 May 2025].
- [24] "Jupyter Notebook," [Online]. Available: <https://jupyter.org/>. [Accessed 1 May 2025].
- [25] "Java," [Online]. Available: <https://www.java.com/en/>. [Accessed 5 May 2025].
- [26] "Tensorflow," [Online]. Available: <https://www.tensorflow.org/>. [Accessed 1 May 2025].
- [27] "Keras," 20 April 2025. [Online]. Available: <https://keras.io/>.
- [28] "OpenCV," [Online]. Available: <https://opencv.org/>. [Accessed 20 March 2025].
- [29] "NumPy," [Online]. Available: <https://numpy.org/>. [Accessed 26 April 2025].
- [30] "Draw.io," [Online]. Available: <https://www.drawio.com/>. [Accessed 1 May 2025].
- [31] "Figma," [Online]. Available: <https://www.figma.com/>. [Accessed 1 May 2025].
- [32] "Canva," [Online]. Available: <https://www.canva.com/>. [Accessed 4 April 2025].
- [33] "API Ninjas," [Online]. Available: <https://api-ninjas.com/>. [Accessed 20 June 2025].
- [34] "TheDogApi," [Online]. Available: <https://thedogapi.com/>. [Accessed 20 June 2025].
- [35] "TheCatApi," [Online]. Available: <https://thecatapi.com/>. [Accessed 20 June 2025].

REFERENCES

- [36] "Encyclopedia of Life," Smithsonian, [Online]. Available: <https://naturalhistory.si.edu/research/eol>. [Accessed 20 June 2025].
- [37] "Animal API," API Ninjas, [Online]. Available: <https://api-ninjas.com/api/animals>. [Accessed 20 June 2025].
- [38] "Zoo Animal API," Rapid, [Online]. Available: <https://rapidapi.com/ahmadawais/api/zoo-animals-api>. [Accessed 20 June 2025].
- [39] "Nominatim," [Online]. Available: <https://nominatim.org/>. [Accessed 20 June 2025].
- [40] "New API," [Online]. Available: <https://newsapi.org/>.
- [41] "TheNewsAPI," [Online]. Available: <https://www.thenewsapi.com/>. [Accessed 20 June 2025].
- [42] "Gemini," Google, [Online]. Available: <https://gemini.google.com/>. [Accessed 3 July 2025].
- [43] "Hugging Face," [Online]. Available: <https://huggingface.co/docs/inference-providers/en/tasks/index>. [Accessed 25 July 2025].
- [44] "GPT AI," [Online]. Available: <https://openai.com/>. [Accessed 25 July 2025].
- [45] "<http://geeksforgeeks.org/nlp/nlp-algorithms-1/>," GeeksforGeeks, [Online]. Available: <https://www.geeksforgeeks.org/nlp/nlp-algorithms-1/>. [Accessed 25 July 2025].
- [46] "Geocoding API," Google, [Online]. Available: <https://developers.google.com/maps/documentation/geocoding/overview>. [Accessed 25 July 2025].
- [47] A. Trivedi, N. Pant, P. Shah, S. Sonik and S. Agrawal, "Speech to text and text to speech recognition systems-Areview," *IOSR Journal of Computer Engineering*, vol. 20, no. 2, pp. 36-43, 2018.

