AI-DRIVEN PET CARE APP FOR VIRTUAL ASSISTANCE AND SYMPTOM DIAGNOSIS

NIVIKA PRASAD A/P KASHI NATH

UNIVERSITI TUNKU ABDUL RAHMAN

AI-DRIVEN PET CARE APP FOR VIRTUAL ASSISTANCE AND SYMPTOM DIAGNOSIS

NIVIKA PRASAD A/P KASHI NATH

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Software Engineering (Honours)

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

September 2025

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Name : Nivika Prasad A/P Kashi Nath

ID No. : 2104910

Date : 17/9/2025

COPYRIGHT STATEMENT

This final year project report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Software Engineering (Honours) at Universiti Tunku Abdul Rahman (UTAR). This final year project report represents the work of the author, except where due acknowledgement 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

I would like to express my deepest gratitude to Dr. Lee Ming Jie, my dedicated research supervisor, for his invaluable guidance, continuous support, and unwavering patience throughout the development of this project. His expertise, insightful feedback, and encouragement were instrumental in shaping the direction and success of my work.

I also extend my sincere appreciation to the faculty and staff of the Lee Kong Chian Faculty of Engineering and Science (LKC FES), Universiti Tunku Abdul Rahman, for providing an enriching academic environment and access to essential resources that greatly supported my learning and research journey. The knowledge and skills I have gained during my studies have been fundamental to the completion of this project.

To my dear friends and the pet-loving users who participated in testing PawHub, thank you for your honest feedback, moral support, and companionship throughout this journey. Your insights helped shape the app into a user-centered solution, and your encouragement kept me motivated to deliver my best.

Lastly, to my beloved family, thank you for your endless love, understanding, and unwavering belief in me. Your constant encouragement gave me strength during challenging times and motivated me to push forward no matter the obstacles.

This project is not only a step toward smarter pet care but also a personal achievement made possible by the kindness and support of everyone mentioned above.

ABSTRACT

Pet ownership brings joy but also significant responsibilities in health management and care. Existing applications often offer fragmented features, lack reliable information, and fail to assist owners in assessing urgent medical concerns. To address these challenges, this project presents PawHub, a mobile application that integrates artificial intelligence with comprehensive pet care tools to deliver a unified, user-friendly experience. The system was developed using the Agile methodology, leveraging React Native for cross-platform frontend development, Node.js with Express for the backend, and Supabase as the centralized database and authentication provider. Key functionalities include an AI-powered chatbot utilizing multiple models via OpenRouter AI, an AI symptom diagnosis tool that generates severity-based assessments, digital health record tracking, automated email reminders through Resend, and a curated educational content module powered by web scraping from trusted sources such as the American Kennel Club (AKC). The application underwent rigorous testing and the results demonstrated high functionality and usability. PawHub successfully bridges the gap between pet owners and virtual veterinary support by providing timely, context-aware guidance, structured health tracking, and expert-backed knowledge. In conclusion, the app proves that AI-enhanced mobile solutions can significantly improve pet care decision-making and owner confidence.

Keywords: Artificial Intelligence; Pet Health Management; Symptom Diagnosis; Web Scraping; Pet Care; Mobile Application

Subject Area: QA76.75-76.765 Computer software

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Over the years, pet ownership has grown significantly, with more individuals and families welcoming pets into their homes. Pets are more than just animals, they are companions that bring joy, comfort, and emotional support to their owners. However, caring for a pet comes with responsibilities, including providing proper nutrition, regular health check-ups, and ensuring their overall well-being. For first-time pet owners, this can be overwhelming, especially when they are unsure about recognizing early signs of illness, maintaining vaccination records, or understanding their pet's specific needs.

Today, life moves quickly, and many people who have pets find it hard to take care manage their pet's health the way they should. Searching for information online can sometimes lead to confusing or unreliable advice, making it difficult to make the right decisions for a pet's well-being. Additionally, scheduling vet visits for minor concerns can be both time-consuming and costly, leaving many owners wishing for a simpler way to get answers about their pet's health.

With advancements in Artificial Intelligence (AI), technology now plays a crucial role in transforming pet care. AI-powered solutions can assist pet owners by providing instant, reliable, and data-driven insights into pet health and wellness. PawHub, an AI-based pet care mobile application, is designed to bridge this gap by offering an AI chatbot for general pet care inquiries and an AI-powered symptom diagnosis tool. The chatbot can answer questions related to pet care, including nutrition, training, and common health concerns, while the symptom diagnosis feature helps pet owners assess whether their pet needs immediate veterinary attention.

Beyond AI assistance, this application will also serve as a comprehensive pet management tool, allowing users to track their pet's health records, vaccination schedules, and important medical history in one place. Additionally, the app provides educational resources through web scraping

tailored for first-time pet owners, guiding them through the essential aspects of pet care.

By leveraging AI, cloud-based data storage, and an easy-to-use interface, this application aims to simplify pet ownership and ensure that pet owners have quick and accurate access to information whenever they need it. PawHub will provide a clever, dependable, and practical solution for pet care in the digital age, regardless of whether a person is a beginner pet owner looking for advice or an experienced pet owner seeking a more effective approach to maintain health records.

1.2 Importance of the Study

Pet ownership is a lifelong commitment that requires continuous care, yet many owners struggle to keep up due to busy lifestyles. Issues like early illness detection, diet changes, and behavioural concerns often go unnoticed, leading to preventable health problems. The lack of accessible, reliable pet care information further complicates decision-making.

In the digital age, pet owners rely on online sources, but these often contain conflicting or misleading advice. Managing health records is also challenging, especially with physical booklets or scattered digital notes. Existing pet care apps typically focus on either health tracking or AI chat support but rarely provide a comprehensive solution that integrates multiple features seamlessly.

This study explores the potential of an AI-powered pet care app that will assist owners by offering symptom diagnosis, real-time AI chatbot support, and efficient health record management. By analysing reported symptoms, the AI will help detect early health concerns, reducing delays in treatment and unnecessary vet visits. A 24/7 AI chatbot with multiple AI models will provide instant, reliable guidance on nutrition, training, and common health issues, minimizing dependence on unverified online sources. Additionally, a digital health management system will streamline pet record-keeping, ensuring organized and stress-free tracking of medical history, vaccinations, and appointments.

Designed for all pet owners, including first-time users, the app will prioritize an intuitive and user-friendly interface. This ensures that key

features such as symptom diagnosis, health tracking, and AI support are easily accessible.

By integrating AI, database management, and an intuitive design, this application has the potential to transform pet care, making it more efficient and accessible. This study highlights the need for AI-driven pet care solutions that empower owners with instant, accurate, and reliable insights, ensuring better well-being for pets in the future.

1.3 Problem Statement

Pet ownership is a fulfilling experience, but it comes with significant responsibilities, particularly in maintaining a pet's health and overall well-being. Ensuring proper medical care, nutrition, and timely veterinary checkups can be challenging, especially when reliable information and structured health management tools are not readily available. Despite the existence of multiple pet care applications, most focus on isolated functionalities such as basic health tracking or AI-powered chatbots. However, there is currently no integrated platform that seamlessly combines AI-driven symptom diagnosis, structured health record management, and reliable pet care guidance. This fragmentation forces pet owners to rely on multiple sources, which is inefficient, time-consuming, and may lead to misinformation or improper pet care.

1.3.1 Difficulty in Accessing Reliable and Expert-Backed Pet Health Information

Pet owners often struggle to find trustworthy sources for pet health information. Many turn to online platforms, which can contain conflicting or misleading advice not backed by veterinary science. This reliance on unverified sources increases the risk of misdiagnosis and improper treatment, potentially worsening a pet's condition. A study exploring pet owner's online search experiences highlights that while the internet is a common resource, the quality and reliability of information vary significantly, affecting interactions with veterinarians (Lai et al., 2021).

The spread of misinformation has been observed in both human and animal healthcare, leading to poor decision-making. For instance, during the

COVID-19 pandemic, a surge in misinformation paralleled trends in human healthcare, diminishing trust in evidence-based treatments. A notable example involved a pet owner who faced opposition in online communities for choosing conventional chemotherapy over holistic treatments for her dog's cancer, showcasing the broader issue of misinformation influencing pet care decisions (Senter, 2024). To address this gap, an AI-powered system providing reliable, expert-backed health guidance is essential for ensuring informed pet care decisions.

1.3.2 Inefficient Health Record Management for Pets

The absence of a centralized system for tracking and managing pet health records poses significant challenges. Many pet owners rely on scattered notes or physical booklets to track vaccinations, vet appointments, and medical history. This disorganized approach can result in missed vaccinations, overlooked medical symptoms, and inconsistencies in treatment plans. Proper record-keeping is crucial in veterinary practice to ensure high-quality care and legal compliance. Inadequate record-keeping can compromise a veterinarian's defence during trial or board review (PLIT, 2021).

Additionally, the transition from traditional paper records to AI-powered Electronic Medical Records (EMRs) has introduced challenges such as data privacy concerns and the need for robust data management systems. Even though EMRs provide advantages like more accessibility and fewer mistakes, sensitive data must be protected during adoption by stringent data security procedures. The evolution of veterinary records underscores the need to address these challenges to fully leverage the advantages of EMRs (Vetrec.io, 2024). A standardized, secure, and accessible health record management system would not only enhance pet healthcare but also safeguard veterinary professionals against legal risks associated with inadequate documentation.

1.3.3 Difficulty in Assessing When to Seek Veterinary Care

Determining whether a pet's condition requires immediate veterinary attention or can be managed at home is a common challenge. Pet owners often find themselves uncertain, leading to either unnecessary vet visits for minor issues or dangerous delays in seeking treatment for serious conditions. Insufficient access to veterinary treatment affects human health in addition to being a serious threat to animal health. (Niemiec et al., 2024).

Since pets cannot verbally communicate their discomfort or symptoms, owners must rely on observable signs, which can be difficult to interpret without expert guidance. The lack of accessible, veterinary-backed resources exacerbates this issue, leaving pet owners dependent on personal judgment or unreliable online sources. For example, during the H5N1 bird flu outbreak, misinformation regarding transmission risks to domestic animals like cats led to confusion and varied responses among pet owners, highlighting the need for clear, accurate guidance (Patterdale, 2025). Without reliable tools to assess their pets' health, owners may either expose them to unnecessary medical interventions or fail to address critical health issues promptly. Implementing an AI-driven symptom checker could bridge this gap, helping pet owners make informed decisions about seeking veterinary care

1.4 Aim and Objectives

1.4.1 Aim

The project's goal is to develop a smartphone application for pet care that uses artificial intelligence (AI) to serve as a virtual assistant for pet owners, offering instant AI chatbot support, symptom-based health assessments, and a digital health record management system. By providing a clever, data-driven, and approachable solution that enables pet owners to make knowledgeable decisions regarding the well-being and health of their pets, the aim is to make pet care easier. By integrating Artificial Intelligence (AI) and cloud-based data management, the app will ensure that pet owners have reliable, instant, and personalized support for their pet care needs, reducing misinformation, unnecessary vet visits, and disorganized health records.

1.4.2 Objectives

i) To develop an AI-powered chatbot with multiple AI models that provides instant responses to pet care inquiries, ensuring reliable and accessible guidance.

ii) To implement an AI-driven symptom diagnosis tool to help pet owners assess health concerns and determine if immediate veterinary attention is needed.

iii) To create a structured digital health record management system to streamline tracking of medical history, vaccinations, and other essential pet's data.

iv) To offer educational resources through web scraping that provides valuable insights and expert knowledge on pet care.

By achieving these objectives, this project will not only streamline pet care management but also enhance the overall pet ownership experience. Pet owners will have access to a reliable, AI-driven virtual assistant that simplifies decision-making, reduces reliance on misinformation, and ensures that their pets receive timely and appropriate care.

1.5 Proposed Solution

The project's intended solution is a mobile application for pet care powered by artificial intelligence that helps pet owners manage the health and wellbeing of their pets by acting as a virtual assistant. This application will integrate artificial intelligence, cloud-based database management, and user-friendly mobile technology to provide an all-in-one platform for pet care. The primary goal of PawHub is to offer pet owners a reliable, efficient, and accessible tool to monitor their pet's health, receive AI-generated care recommendations, and maintain essential records, reducing the stress and uncertainty of pet management.

The key features of the application include an AI-powered chatbot with options from multiple AI model, a symptom diagnosis tool, and a pet health records management system. The AI chatbot, powered by third party API's from multiple AI models, will provide real-time responses to general pet care inquiries, offering users guidance on nutrition, training, and common pet

health concerns. The symptom diagnosis feature will also utilize AI models to analyse user-provided symptoms and categorise the health issues based on severity, enabling pet owners to take timely action when necessary. Additionally, the pet health records management system will allow users to store and track their pet's information, vaccination history, vet visits, and other medical records, ensuring that critical health information is always accessible.

To ensure secure access and data protection, the application will implement Supabase authentication for user management. The database operations will also be handled by Supabase, which will store user and pet-related data efficiently while allowing seamless integration with the application's features. Moreover, the app will include educational resources, such as pet care articles and guides, to enhance pet owner's knowledge and provide preventive care insights, this will be implemented through web-scraping. The PawHub application will be developed using React Native for cross-platform compatibility and node.js for backend, ensuring that it can reach a wider audience of pet owners using both Android and iOS devices.

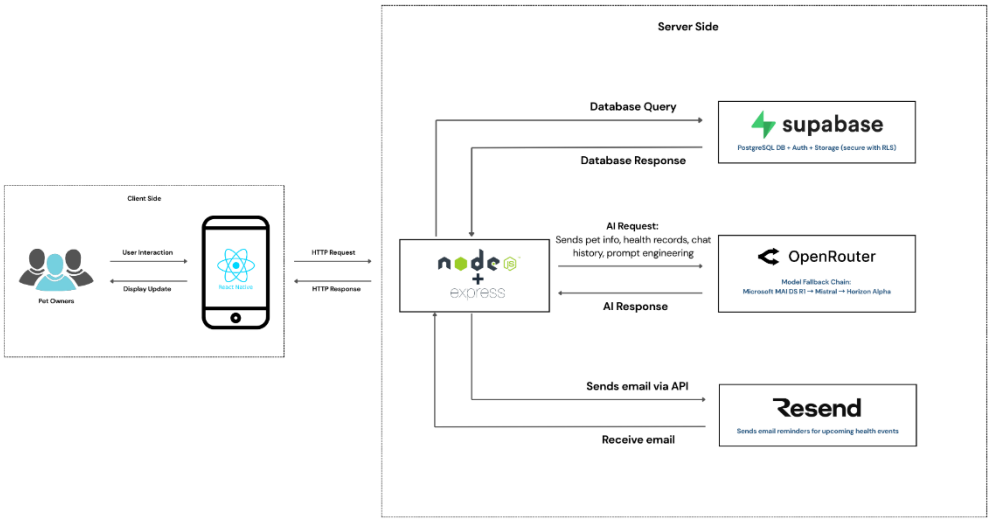


Figure 1.1: System Architecture

PawHub was built on a secure, full-stack architecture that integrates React Native (frontend), Node.js with Express (backend), and Supabase (PostgreSQL database and authentication). The frontend communicated with a dedicated backend server, which acts as a secure gateway for all API requests.

This server handles authentication, data validation, and routing while enforcing rate limiting and JWT-based access control to protect user data.

The backend connects to Supabase to manage user profiles, pet details, health records, and AI interaction history, all secured with Row Level Security (RLS) policies to ensure users only access their own data. For AI-powered features, the backend forwarded enriched prompts to OpenRouter.ai, enabling intelligent chatbot responses and symptom diagnosis with model fallback logic for reliability. Additionally, an automated email reminder system uses Resend to notify users of upcoming health events, triggered by a background checker that runs every 60 seconds.

All sensitive credentials including Supabase, OpenRouter, and Resend API keys are securely stored in environment variables using .env files, ensuring they are never exposed in the frontend or version control. This layered, well-integrated architecture supports scalability, security, and real-time functionality, making PawHub a robust and intelligent solution for modern pet care management.

1.6 Proposed Approach

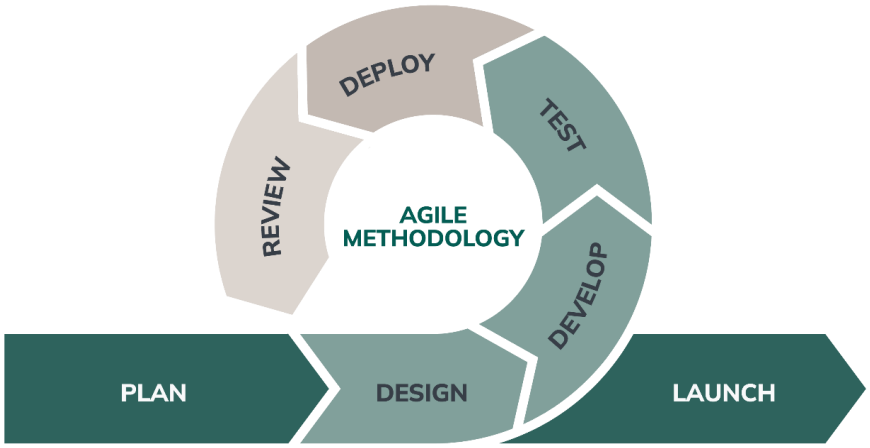


Figure 1.2: Agile Methodology (Agile software development: everything you need to know, 2024)

Throughout the project lifespan, PawHub will follow an Agile software development methodology combined with an incremental approach to ensure continuous improvement, adaptability, and high-quality outcomes. This approach supports the gradual development and refinement of features such as

the AI chatbot, symptom diagnosis tool, and pet health records system by dividing the project into manageable sprints or development cycles. At the end of each sprint, a functional component of the application will be tested, and evaluated, allowing for supervisor feedback and necessary adjustments to be integrated into future iterations.

Agile's flexibility is especially beneficial for the mobile application, PawHub that relies on modern technologies such as AI and cloud-based services, where requirements may evolve based on testing outcomes or user needs. The project will begin with an initial planning and requirement analysis phase, followed by design, development, testing, and deployment, with each phase being revisited and refined as needed. By adopting this methodology, the development process remains dynamic and user-focused, ensuring that the final product is not only functional and reliable but also aligned with user expectations and evolving technological trends.

1.6.1 Requirement Analysis

The first phase involves requirement analysis, where research will be conducted to identify the challenges pet owners face when managing their pet's health. Surveys and studies on existing pet care applications will be used to gather insights, allowing for the identification of essential features that differentiate PawHub from other solutions. This phase is crucial for defining the scope of the project and aligning the application's functionalities with user needs.

1.6.2 Design and Prototyping

Following this, the design and prototyping phase will focus on creating UI/UX wireframes to ensure an intuitive and user-friendly experience. The system architecture will be structured to define how the application components, including the AI chatbot, database, and user interface, will interact. This stage will also outline data models for pet health records and chatbot responses.

1.6.3 Design and Prototyping Development

The development phase will involve coding and integrating the core features of the application. This includes implementing Supabase authentication for

secure user management, configuring Supabase for database operations, and integrating the AI API models for AI-powered chatbot functionality. Integrating Resend API configuration for pet health record reminders. The symptom diagnosis module will be developed using AI models capable of analysing input symptoms and providing relevant analysis and recommendation. Throughout this phase, React Native and Node.js will be used to develop the application, ensuring efficiency in both functionality and design.

1.6.4 Testing and Optimization

Once the core features are implemented, the testing and optimization phase will be conducted to evaluate the app's performance, usability, and accuracy. Unit testing and system testing will be performed to detect bugs and ensure smooth interactions between different components. The AI chatbot and symptom diagnosis system will also undergo validation testing to measure response accuracy and reliability. User feedback will be gathered through initial test runs to refine the application further, addressing any usability concerns or feature gaps.

By following an agile and iterative development approach, PawHub will be refined continuously to enhance its efficiency, accuracy, and user experience. The structured methodology ensures that the project progresses systematically while allowing flexibility for improvements. This approach not only facilitates efficient project execution but also ensures that PawHub evolves to meet the dynamic needs of pet owners effectively.

1.7 Scope and Limitation of the Study

1.7.1 Scope

The research study is about the development and implementation of an AI-powered application for pet care that aims to give pet owners virtual support in overseeing the health and general welfare of their animals. The proposed application will be developed using React Native, ensuring compatibility for cross platform devices. For backend services, Node.js and Supabase will be utilized to handle user authentication, database management, and cloud storage, offering a secure and scalable solution for storing pet health records and user

data. The AI-powered functionalities, including the chatbot and symptom diagnosis tool, will be integrated using an AI models to enhance user experience by providing intelligent and data-driven assistance tailored to pet care needs.

1.7.2 Development and Research Scope

The proposed application will incorporate AI-based virtual assistance and pet health management functionalities. The scope of research includes the evaluation of AI-powered chatbots for pet care, symptom diagnosis accuracy, and the effectiveness of cloud-based storage for managing pet health records. The development scope will involve implementing core features such as user authentication, pet profile management, symptom diagnosis, AI-driven chatbot interactions, and reminder notifications.

1.7.3 Target Users

The primary target users of the PawHub application are pet owners who are looking for accessible, AI-powered virtual assistance to support them in managing the health and well-being of their pets. These users may include first-time pet owners who need guidance on general care, feeding, training, and early detection of symptoms, as well as experienced owners who want to streamline their pet management routines using digital tools. In addition to the Pet Owner, the Admin is a secondary user responsible for managing feedback submitted by users and curating educational content. All admin tasks are performed via the Supabase web dashboard and are not part of the PawHub mobile application.

The application is also beneficial for veterinarians, pet caregivers, and pet care enthusiasts. Veterinarians may find value in features such as organized health records and symptom logs provided by the app, which can improve diagnosis accuracy and treatment plans during in-person consultations. Pet caregivers, such as pet sitters or boarding staff, can also use the app to better understand the pet's needs, care routines, and medical history while the owner is away. Meanwhile, pet care enthusiasts and community members who actively engage in learning about pet health and behaviour can benefit from the educational resources and AI chatbot for general pet-related inquiries.

1.7.4 Application Features

The proposed application will include several core features to simplify and enhance pet care management:

i) AI-powered chatbot

The AI Chatbot in the PawHub app acts as a virtual assistant, helping pet owners by answering questions related to pet nutrition, training, behaviour, and common health concerns. Powered by natural language processing (NLP) APIs, the chatbot provides structured and relevant responses based on user queries, pet informations and health records, offering quick and accessible support without the need to search multiple sources. To improve the accuracy and adaptability of responses, the chatbot integrates three different AI models, allowing users to switch between them based on their preference for response style or depth. This multi-model approach ensures more flexible and tailored assistance across a variety of pet care topics and a fallback logic is implemented to avoid system downtime. While the chatbot does not replace professional veterinary care, it helps bridge the gap by offering valuable insights and general guidance to pet owners at any time.

ii) Symptom diagnosis tool

The Symptom Diagnosis tool in the PawHub app is designed to assist pet owners in assessing the urgency of their pet's health condition based on user-input symptoms. By utilizing an AI-powered model and a AI fallback logic to avoid system downtime, the tool analyzes the described symptoms and returns a health status categorized by a three-colour indicator system, green for minor issues, yellow for moderate concern, and red for serious conditions that may require immediate veterinary attention. This simple visual system helps users quickly understand the potential severity of their pet's symptoms. A clear disclaimer is displayed at the top, reminding users that the tool is for informational purposes only and does not serve as a replacement for professional veterinary diagnosis or treatment. The

Symptom Diagnosis tool aims to empower pet owners to make more informed decisions while emphasizing the importance of seeking professional veterinary care when needed.

iii) Pet profile management

The Pet Profile Management system allows users to create and manage individual profiles for each of their pets. These profiles will include vital details such as the pet's name, breed, age, gender, weight, and medical history. The feature is designed to give pet owners a personalized dashboard where they can quickly view and update pet-specific information. This makes it easier to keep track of important dates such as vet appointments, vaccination due dates, and dietary preferences. For users with multiple pets, the feature offers a convenient way to switch between profiles, ensuring each pet's needs are addressed efficiently.

iv) Pet health record management

This feature enables users to digitally store, view, and update their pet's medical records in an organized and secure format. It includes essential information such as vaccination history, scheduled check-ups with email reminders, prior illnesses, diagnoses, prescribed medications and allergy notes. By centralizing this data within the app, pet owners can easily track their pet's health status and share comprehensive reports with veterinarians during appointments. This function reduces the risk of missing important medical details and helps ensure continuity in healthcare, particularly for pets with chronic conditions or special medical needs.

v) Educational resources

The Educational Resources section acts as a learning hub, offering curated content such as expert-written articles, how-to guides, and infographics on pet nutrition, grooming, exercise, training, and common illnesses. The content is organized into clear categories for easy navigation with a search function. Web scraping is used to

automate the gathering of educational materials from trusted pet care websites. By using libraries like Axios and Cheerio to fetch and parse web data, PawHub intelligently collects and updates its resources without manual input. This automation ensures that users, especially first-time pet owners, have continuous access to reliable and categorized pet care information within the app.

vi) User profile management

This feature allows users to create secure and personalized accounts by registering with their name, email address, password and phone number. Once logged in, users can manage their profile settings and update information. The system ensures secure authentication and session handling to protect user data, while also offering a personalized interface that enhances the overall user experience. User profile management also facilitates the storage of app activity and preferences, making it easier for users to resume tasks and interact with the application seamlessly.

vii) User feedback system

The User Feedback System provides a structured platform within the app where users can submit feedback, report bugs, and suggest improvements. This feature is crucial for ongoing development and refinement of the application, as it gives direct insights into user experiences and expectations. The feedback system may include rating options, comment sections, and a quick survey form to collect both qualitative and quantitative data. By engaging users in the development cycle, this feature supports continuous improvement and ensures that the app evolves based on actual user needs and challenges.

1.7.5 Limitations

While this application aims to provide an advanced and intelligent platform for pet care management, several limitations must be acknowledged:

i) Accuracy Constraints of AI-Powered Features

The chatbot and symptom diagnosis tool will rely on pre-trained AI models API that generate responses based on existing datasets. While efforts will be made to improve precision using prompts, the AI may not always provide entirely accurate diagnoses or recommendations, especially in rare or complex medical cases. The system will be trained on general pet care patterns, meaning that unique scenarios or breed-specific conditions may not be well-addressed. Additional validation from professional veterinarians will be necessary for cases requiring expert medical judgment.

ii) Not a Replacement for Veterinary Care

The AI-powered tools will provide guidance and preliminary assessments but cannot substitute the expertise of a licensed veterinarian. The application will serve as a supplementary tool to help pet owners understand potential health concerns and manage routine pet care, but it will not provide official medical diagnoses or treatment plans. Users will be strongly encouraged to consult a veterinarian for serious or urgent medical conditions rather than relying solely on AI-generated suggestions.

iii) Internet Dependency

Since many of the AI-driven features and cloud-based services will rely on API calls for data processing and retrieval, a stable internet connection will be required for optimal performance. Users in areas with poor or limited connectivity may experience delays in chatbot responses, incomplete symptom analysis, or restricted access to stored health records. While certain offline functionalities may be implemented in the future, real-time AI interactions and cloud-stored data will still require an active internet connection.

iv) Potential User Over-Reliance on AI

The convenience of AI-powered recommendations may lead some pet owners to rely too heavily on automated guidance instead of seeking human expertise. While the chatbot and symptom analysis tool will aim to provide valuable insights, they should be used as advisory tools rather than definitive sources of medical advice. To mitigate this risk, the application will emphasize the importance of professional veterinary consultation and promote responsible usage among its users.

Despite these limitations, the proposed AI-driven pet care application is designed to be a valuable and user-friendly tool that enhances pet care management. By integrating AI capabilities with essential health management features, the app will provide a structured and informed approach to pet care. While technology cannot replace the expertise of veterinarians, this application aspires to bridge the gap between traditional pet care and modern AI-driven assistance, making pet ownership more convenient, informed, and accessible.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The improvement of artificial intelligence (AI) and mobile technology has significantly transformed the pet care industry, enabling pet owners to monitor their pet's health more efficiently. AI-powered solutions such as chatbots, symptom diagnosis tools, and veterinary decision-support systems have emerged to assist pet owners in making informed healthcare decisions. Through immediate access to health-related insights, early diagnosis, and real-time pet health monitoring, these advances aid in bridging the gap between pet owners and veterinary specialists.

However, despite the availability of various AI-driven pet care applications, many solutions face limitations, such as a lack of comprehensive health monitoring, reliance on internet connectivity, and limited AI accuracy in symptom diagnosis. Some applications focus on image-based disease detection, while others provide general pet care advice without offering personalized health insights. This literature review explores existing AI-based pet care applications, their benefits, and their limitations to highlight the need for a more integrated and accessible solution.

The PawHub application aims to address these gaps by offering an AI-powered chatbot and API-based symptom diagnosis system that provides real-time, reliable health insights. Unlike other pet care apps, PawHub integrates multiple AI-driven features, symptom analysis, and offline functionality, ensuring a user-friendly and accessible pet healthcare assistant.

2.2 Literature Review

This section will examine key research and existing solutions relevant to PawHub's AI-driven pet care features.

2.2.1 Artificial Intelligence (AI) Chatbots in Pet Healthcare

The use of AI-powered chatbots in pet healthcare has gained significant attraction, offering pet owners immediate access to health-related information

and preliminary diagnoses. Jokar et al. (2024) emphasize that AI chatbots, such as ChatGPT, provide 24/7 availability and convenience, allowing pet owners to make informed decisions without requiring immediate veterinary consultations. Since these virtual assistants employ Natural Language Processing (NLP) to understand pet health issues and deliver suitable solutions, they are helpful tools for managing pet care.

The power of AI chatbots to offer immediate access to medical information is one of its main advantages. Chatbots assist pet owners by analysing symptoms, offering first-aid suggestions in urgent situations, and eliminating consultation fees for minor health concerns. Additionally, these AI-driven assistants serve as educational resources, providing insights into pet nutrition, behaviour, and preventive care strategies.

Despite these advantages, AI-driven pet healthcare systems also present several challenges. AI chatbots lack the ability to perform physical examinations, increasing the risk of misdiagnosis. Furthermore, pet owners may over-rely on AI recommendations, potentially delaying professional veterinary care, which could worsen medical conditions. AI-generated responses may also include inaccurate or generalized advice, particularly for complex medical cases requiring professional expertise.

To mitigate these risks, Jokar et al. (2024) propose integrating AI chatbots as complementary tools alongside veterinarians, ensuring that chatbots handle preliminary diagnosis and routine inquiries while complex cases remain under veterinary supervision. They also recommend the establishment of regulations and guidelines to prevent AI chatbots from acting as independent substitutes for professional veterinary care.

For my PawHub application, AI chatbots will play a crucial role in enhancing pet healthcare accessibility by offering instant, AI-driven insights while ensuring that pet owners receive verified and professional veterinary support when needed. By implementing an AI-powered chatbot that integrates accurate symptom analysis, first-aid recommendations, and direct veterinarian referrals, PawHub aims to address the limitations of existing AI-based pet care solutions, ensuring a more reliable and comprehensive virtual assistant for pet owners.

2.2.2 Artificial Intelligence (AI) in Veterinary Medicine

The use of artificial intelligence (AI) in animal health care has significantly improved diagnostic accuracy, treatment planning, and medical record management. AI-based radiographic analysis is one of the key advancements, allowing automated detection of abnormalities such as pulmonary nodules, cardiomegaly, and skeletal disorders with high precision (Appleby & Basran, 2022). AI-powered imaging tools assist veterinarians in identifying diseases more efficiently, reducing the time required for diagnosis while improving overall patient care.

Beyond imaging, AI applications in veterinary medicine are expanding into predictive analytics and decision support systems, which can help veterinarians analyse historical health data and anticipate potential medical conditions. AI-driven systems are also enhancing automated medical record management, enabling structured data extraction from unstructured clinical notes to streamline patient histories and improve continuity of care.

However, AI adoption in veterinary medicine presents challenges, particularly in data quality and model reliability. AI models require large datasets of accurately labeled veterinary records to ensure generalizability and precision. Variability in diagnostic interpretations and inconsistencies in data collection across different veterinary clinics may lead to potential biases in AI-driven diagnoses. Therefore, veterinary professionals must be actively involved in training AI models, ensuring that these technologies align with clinical best practices and ethical considerations (Appleby & Basran, 2022).

For PawHub, the integration of AI-driven health monitoring and symptom analysis can significantly enhance pet care by providing real-time insights to pet owners. AI-driven diagnostic technologies can help discover diseases early by providing initial health evaluations prior to a professional veterinary consultation. By leveraging AI in veterinary medicine, PawHub aims to bridge the gap between pet owners and veterinarians, ensuring timely and informed healthcare decisions for pets.

2.2.3 Machine Learning for Pet Health Monitoring

The use of artificial intelligence (AI) and machine learning (ML) in veterinary medicine has greatly improved the accuracy of diagnosing and tracking the health of pets. Specifically, advanced machine learning techniques, including CNNs, have shown remarkable success in detecting skin conditions in pets and identifying other potential health problems. Studies have shown that CNN-based models can achieve 92% accuracy in classifying common pet conditions such as dermatitis, eczema, and fungal infections (Mehra, 2025). These findings suggest that AI-driven image classification is a valuable tool for early disease detection and intervention.

Traditional veterinary diagnosis relies heavily on human expertise, which can lead to inconsistencies in assessment and delayed detection of health issues. Through the use of huge amounts of annotated photos, the development of AI-powered dermatology tools seeks to standardize diagnoses, increase diagnostic reliability, and lessen the workload for veterinarians. Mehra (2025) emphasizes that deep learning models, when trained on extensive veterinary databases, can effectively differentiate between various skin conditions, allowing for faster and more accurate diagnoses with minimal human intervention.

However, despite these advancements, AI-driven pet health monitoring systems face multiple challenges. The accuracy of CNN-based models depends on the quality of training images, which may vary based on factors such as lighting conditions, image resolution, and severity of the skin condition. Furthermore, AI models may struggle with new or rare pet illnesses, leading to misclassifications that require human verification. Another major concern is user adoption, as many pet owners still prefer direct veterinary consultations, making AI-based diagnostics a supplementary tool rather than a standalone solution. Additionally, cloud-based AI models often require consistent internet connectivity, limiting their usability in offline environments or remote areas where veterinary services are already scarce.

While PawHub does not rely on image-based CNN models, the concept of leveraging machine learning API for accurate symptom analysis aligns with the app’s goal of offering reliable, real-time health insights. By focusing on API-based symptom diagnosis rather than image classification,

PawHub aims to provide a more accessible solution that does not depend on high-quality images. This ensures that pet owners can benefit from AI-assisted diagnostics even in low-resource environments, making PawHub a more practical tool for everyday pet healthcare management.

2.2.4 Comparison of Literature Review Research Papers with my PawHub Application

Table 2.1: Comparison of Literature Review Research Papers

Study	Key Features	Limitations	Relevance to PawHub
AI Chatbots in Pet Healthcare	-24/7 virtual assistance using NLP -Provides general health info -Cost-effective	-Cannot conduct physical exams -Risk of misdiagnosis -May delay professional help	-Integrates chatbot for instant support -Offers symptom-based suggestions -Encourages vet consultation
AI in Veterinary Medicine	-AI in radiography (e.g., X-ray analysis) -Medical record automation -Predictive analytics	-Requires large, clean datasets -Clinical bias possible -Not accessible to pet owners directly	-Bridges gap with vets through AI insights -Provides real-time monitoring data to users
Machine Learning for Pet Health Monitoring	-CNNs detect skin conditions -High diagnosis accuracy -Deep learning image classification	-Accuracy depends on image quality -May miss rare conditions -Internet dependency in cloud models	-Focuses on symptom input over image data -Reduces internet dependency -Offers generalized suggestions

A comparative analysis of AI-based pet healthcare solutions reveals both strengths and weaknesses across various approaches. AI chatbots offer 24/7 virtual assistance for health advice and symptom assessments through Natural Language Processing (NLP), but their inability to conduct physical exams can lead to misdiagnoses and over-reliance on AI-generated recommendations, delaying veterinary care. PawHub addresses this by integrating an AI chatbot that provides real-time insights while guiding users to professional veterinary services when needed.

In veterinary medicine, AI technologies like radiographic image analysis and predictive analytics improve diagnostic accuracy, yet their effectiveness relies on high-quality data and remains inaccessible to pet owners. PawHub bridges this gap by offering AI-driven text input symptom analysis, enabling pet owners to gain preliminary health insights before seeking professional consultation. While machine learning models such as convolutional neural networks (CNNs) are effective in detecting skin diseases and health conditions through image-based classification, they are limited by factors like image quality, internet dependency, and challenges in diagnosing rare conditions. In contrast, PawHub does not rely on image-based analysis but uses an AI-powered API for symptom-based diagnosis, providing broader usability, including offline functionality.

2.3 Analysis of Existing Pet Care Applications

The increasing use of mobile technology in pet care has led to the development of numerous applications that help pet owners in track the health of their pets. The applications range from fundamental pet management tools to advanced AI-driven health monitoring systems. The integration of artificial intelligence has significantly enhanced the ability of these applications to detect symptoms, provide virtual veterinary consultations, and offer personalized care recommendations. However, despite the availability of several pet care applications, many existing solutions have notable limitations.

Some applications lack comprehensive AI-driven support, while others do not provide an all-in-one solution for pet health monitoring, symptom diagnosis, and veterinary assistance. Additionally, the reliance on internet connectivity in most AI-powered pet care apps limits their

accessibility in offline environments, which can be a significant drawback for pet owners in areas with poor connectivity.

This literature review examines four widely used pet care applications analysing their features, strengths, and limitations. By assessing these existing solutions, this review aims to identify gaps in the market and establish the need for a more integrated, AI-driven pet care application. The proposed PawHub application seeks to address these gaps by offering a holistic and intelligent pet care assistant that combines AI-driven diagnostics, real-time health tracking, multi-user accessibility, and offline functionality.

2.3.1 TTcare Application

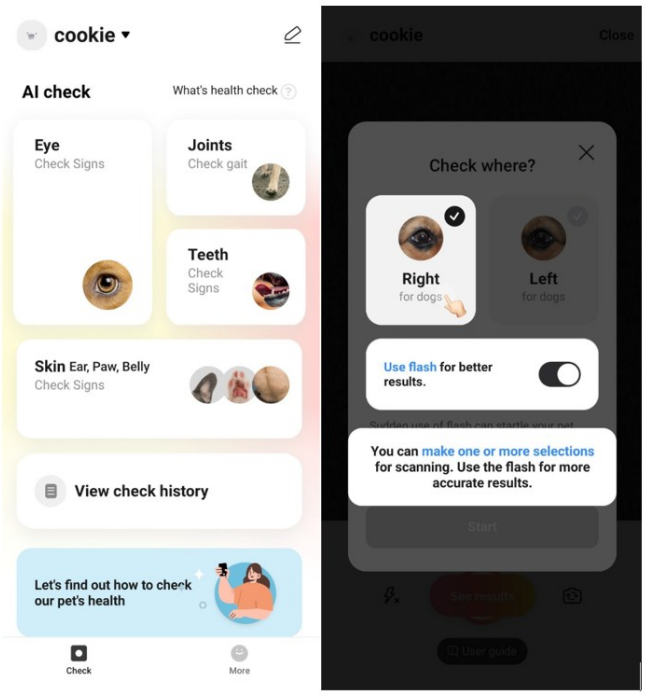


Figure 2.1: Main Features of TTcare Application

TTcare, developed by AI FOR PET, is an AI-powered pet health monitoring application that focuses on early disease detection through image recognition technology. By allowing pet owners to upload photos of their pets, the application uses artificial intelligence to analyse visual indicators of potential health issues, particularly in the eyes, skin, joints, teeth and ears. As shown in Figure 2.1, One of its advantages is, it uses of AI in detecting abnormalities, enabling early intervention before symptoms worsen and once a user selects

and option, it provides user with clear guidelines to get accurate results. (PET, 2023).

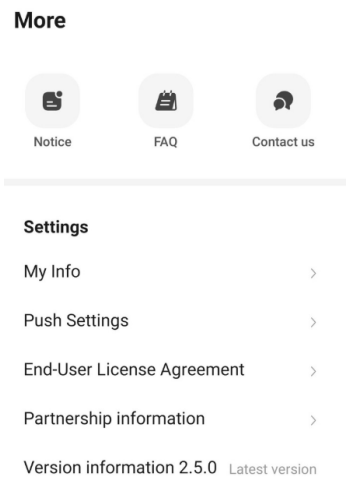


Figure 2.2: Basic Features in TTcare Application

Other than that, it also provides user with some basic functions like user profile management, FAQ and contact us page that will help the user in identifying how the application works as shown in Figure 2.2.

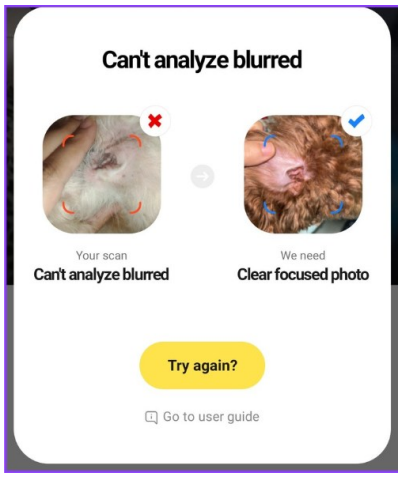


Figure 2.3: TTcare Application Analysis Error

Despite these strengths, TTcare has major limitations. Its diagnostic approach relies solely on image-based analysis, meaning it cannot assess internal health conditions or detect symptoms that do not present visible signs. Blur images or slightly discoloured images are also one of the setbacks as

shown in Figure 2.3, the image that provided to the system was captured using flash and is not blur but the system couldn't analyse the image for diseases detection. Furthermore, the application solely relies on image recognition technology feature and no assistance, leaving users with limited guidance beyond scan results. Another significant drawback is its dependency on internet connectivity for AI-based analysis, making it inaccessible in offline environments.

2.3.2 PetVet AI Application

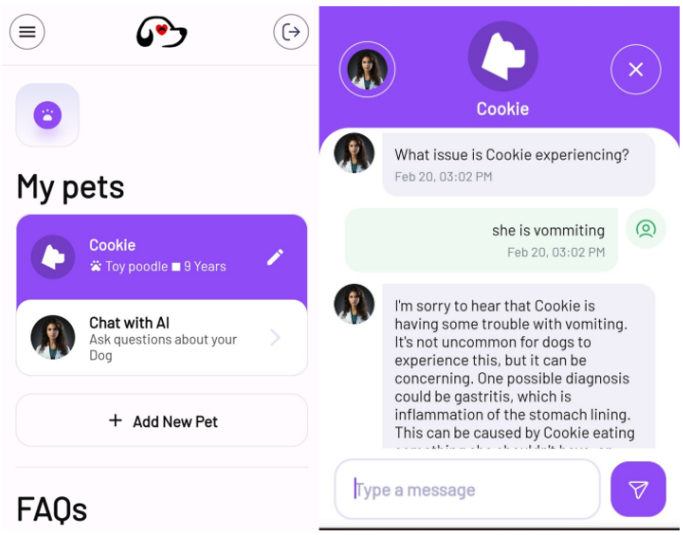


Figure 2.4: PetVet AI Main AI Chatbot Feature

PetVet AI is a pet health management application that offers AI-driven virtual veterinary assistance. Unlike TTcare, which focuses on image-based detection, PetVet AI allows users to have real time conversations for personalised pet care enquiries. As shown in Figure 2.4, the application includes an AI chatbot that delivers real-time responses to pet health-related inquiries, assisting pet owners in making well-informed decisions about the welfare of their animals. This application is trained with vast veterinary medical records and expert insights to assist users on their pet care enquiries (LLC, 2023).

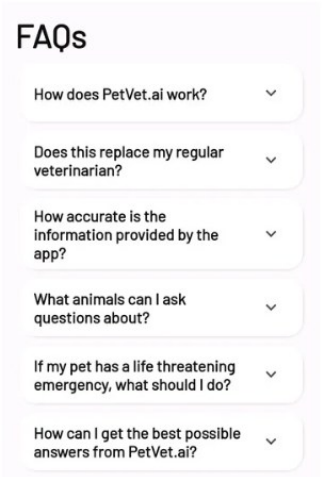


Figure 2.5: PetVet AI simple FAQ

Additionally, the app provides a simple FAQ as shown ins Figure 2.5 that gives the users with a basic knowledge about the application and how to use it. While PetVet AI Pet offers real-time AI support, its recommendations remain advisory rather than diagnostic, meaning they should not replace professional veterinary consultations. Another limitation is its reliance on internet connectivity, restricting accessibility in areas with poor network coverage. The application also lacks advanced health management tool, making it difficult for pet owners to track long-term changes in their pet's health.

2.3.3 PetVitality Application

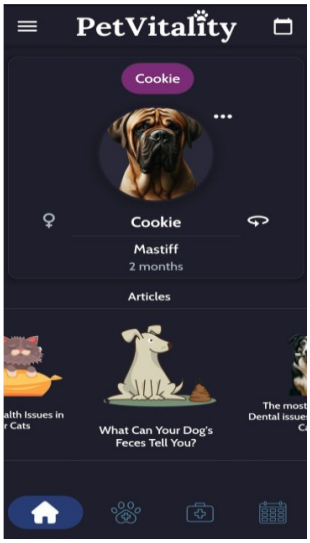


Figure 2.6: PetVitality Home Page

PetVitality is a wellness-focused pet care application that integrates health monitoring with activity tracking. It provides pet owners with tools to track their pet's weight, temperature, and vital signs, helping them detect potential health concerns early. The application also features activity tracking capabilities that users can input manually, allowing owners to monitor movement and behaviour patterns. The app also provides appointment reminders for veterinary visits and medication schedules, ensuring that pet owners stay on top of their pet's healthcare routines. Other than that, it also provides users with articles for general pet care training guides, personal gallery and document storage. (Lyssa AS, 2024).

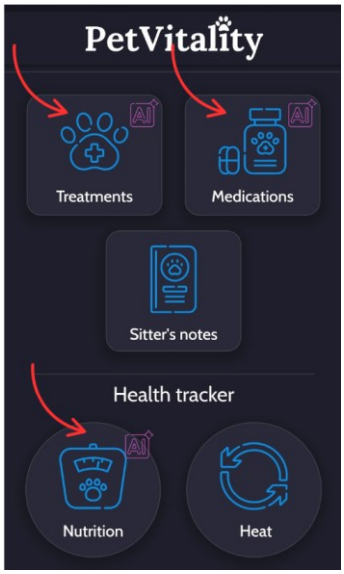


Figure 2.7: PetVitality AI tools features

Firstly, this app integrates several AI tools, including an ingredient scanner that analyses pet food labels for harmful or beneficial components, and a breed-and-age-specific content generator for personalized care tips. Another AI feature offers automated medication insights and emergency mini guides. Other than that, it also provides an AI-assisted pattern recognition tool that analyses the data logged by users to detect potential early signs of health issues. For example, if consistent changes are noted in weight, temperature, or feces tracking, the app may alert the user to seek veterinary advice.

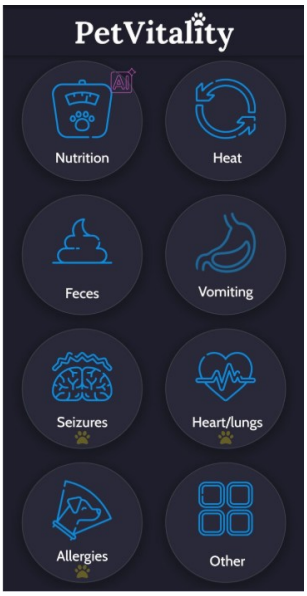


Figure 2.8: PetVitality Specialised health trackers feature

PetVitality also includes specialized health trackers such as a visual poop score tracker for digestive health, a heat cycle predictor for reproductive planning, and a manual heart and lung respiratory rate monitor for pets with heart/lungs conditions, vomiting tracker, manual seizure tracker and allergies tracker. These features allow users to closely observe specific aspects of their pet's physical health. The seizure tool is designed for pets with epilepsy, this specialized tool tracks the regularity and frequency of seizures and uses predictive analytics to estimate when the next episode might occur. This helps owners be more prepared and proactive in managing their pet's condition.

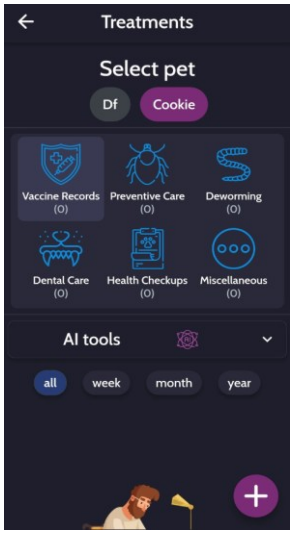


Figure 2.9: PetVitality Detailed Health Tracking Feature

PetVitality also includes a detailed health tracking feature as shown in Figure 2.9 that allows users to log and monitor vital health information such as medical conditions, vaccine records, dental care etc. This component acts as a centralized health log, helping users detect unusual trends or changes in their pet's condition over time.

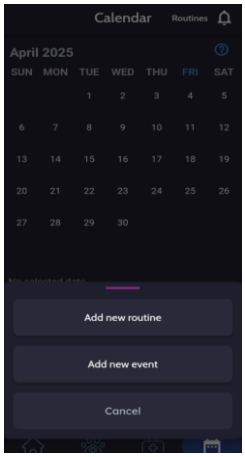


Figure 2.10: PetVitality Routine Scheduler and Reminders Feature

As shown in Figure 2.10, the routine scheduler and reminders is another essential feature that enables users to schedule and receive reminders for essential pet care tasks such as grooming, training, medication administration, and routine vet check-ups. Users can set recurring alerts and view their pet's care calendar.

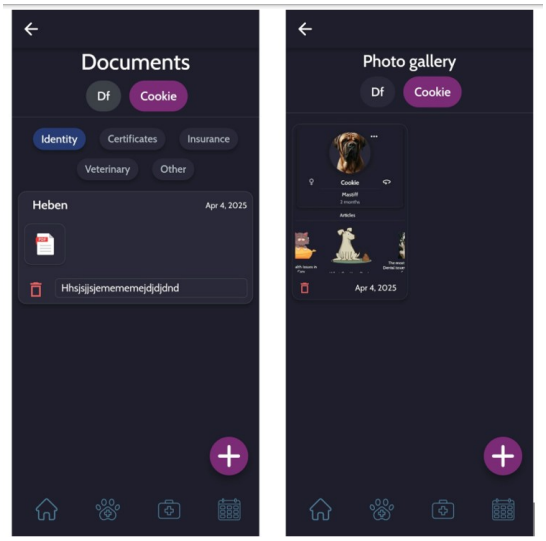


Figure 2.11: PetVitality Document Storage and Personal Gallery Feature

Besides, the document storage feature allows pet owners to scan and upload important medical records, vaccination cards, and prescription documents directly into the app. These documents are securely saved and can be organized in folders for easy access and sharing. Other than that, to create a more personal experience, the app also allows users to store photos of their pets in a personal gallery.

Despite its comprehensive approach to pet wellness, PetVitality has limitations. Although the app tracks activity levels, it does not provide predictive health alerts based on behavioural changes. Another drawback is the absence of an AI chatbot, which limits the app's ability to provide instant pet care guidance.

2.3.4 11Pets Application

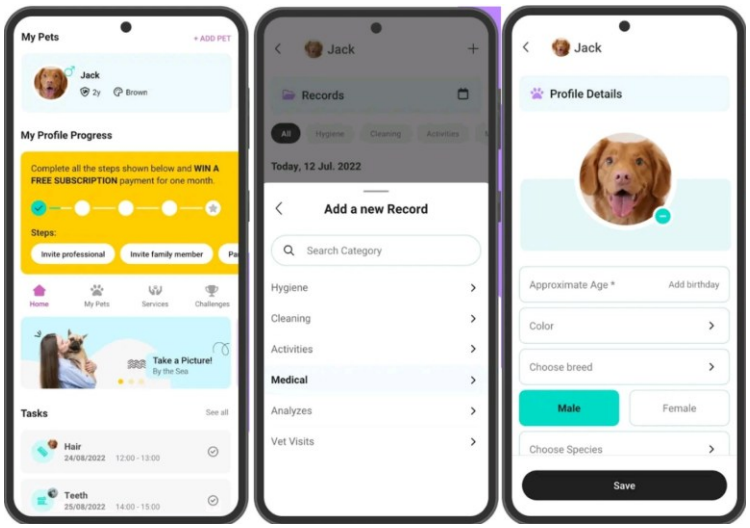


Figure 2.12: 11Pets General Functions

11Pets is a pet care management application designed for both individual pet owners and professional pet caretakers. It serves as a centralized platform for storing medical records, tracking grooming schedules, and managing pet nutrition. The application allows users to maintain a digital record of vaccinations, medications, allergies, and lab results, ensuring that all pet health data is easily accessible.

Additionally, it provides features for managing feeding schedules and tracking grooming routines, making it a valuable tool for maintaining a

structured pet care plan. One of the unique aspects of 11Pets is its support for pet shelters and rescue organizations, allowing them to manage adoption profiles and track the medical history of rescued animals (Ltd, 2015).

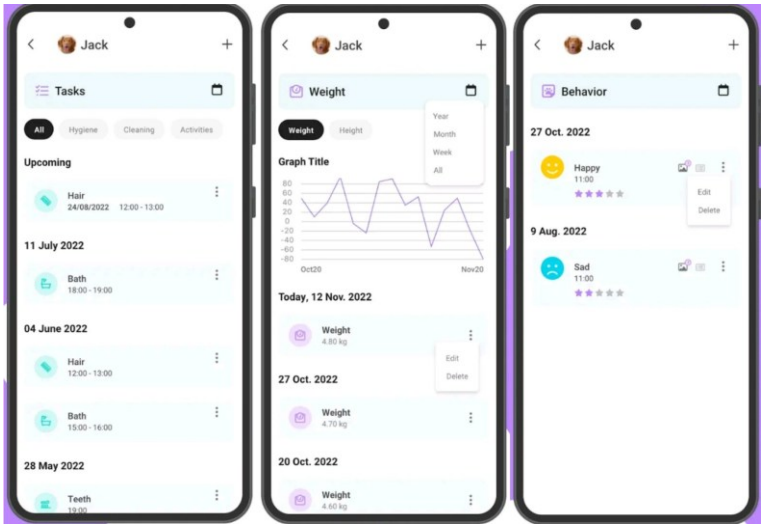


Figure 2.13: 11Pets Health Tracking Features

One of the standouts features of 11Pets is its versatility, it offers two distinct versions, Pet Care for personal use and Business for professional pet care providers, including shelters and boarding services. These functionalities make it an ideal tool for managing both single-pet households and large-scale animal welfare operations. Moreover, the platform enables easy sharing of medical records with veterinarians and allows offline data access, ensuring users can manage pet care even in low-connectivity environments.

It also includes features to manage grooming routines and hygiene care. Pet owners can schedule and track activities such as baths, brushing, nail trimming, and other grooming tasks, helping maintain their pet's cleanliness and well-being. It also allows them to track their pets weight and behaviour and can view their analytics in a graph format.

Although 11Pets is a robust pet management tool, it lacks AI-driven features. Without AI-powered insights, pet owners must rely on manual record-keeping rather than automated health assessments. While 11Pets offers offline access, making it more convenient in low-connectivity environments, its functionality is primarily limited to record management rather than proactive pet health monitoring.

2.3.5 Comparison of Existing Pet Care Applications with PawHub App Features

Table 2.2: Comparison between existing pet care applications and PawHub App Features

Features/App	TTCare	PetVet AI	PetVitality	11Pets	PawHub
AI Chatbot	No	Yes	No	No	Yes
AI-Symptom Diagnosis	Yes (image)	No	No	No	Yes
Pet Profile Management	Yes	No	Yes	Yes	Yes
Pet Health Record Management	No	No	Yes	Yes	Yes
User Profile Management	No	No	Yes	Yes	Yes
Educational Resources	Yes (FAQ)	Yes (FAQ)	Yes	No	Yes
Feedback	Yes	No	Yes	No	Yes

The comparative analysis of existing pet care applications TTcare, PetVet AI Pet, PetVitality, and 11Pets, reveals significant gaps that PawHub aims to address. While TTcare excels in AI-powered visual diagnostics, it is limited to image-based analysis and requires consistent internet connectivity. PetVet AI Pet introduces a helpful AI chatbot yet lacks core diagnostic tools or health record management. PetVitality offers comprehensive wellness tracking and personalized care features, but it does not include AI-based diagnostics or chatbot interaction for real-time assistance. Meanwhile, 11Pets focuses on extensive pet health and grooming record-keeping with offline access but does not implement AI capabilities.

In contrast, PawHub combines the strengths of all these apps, integrating AI-driven symptom diagnosis through text input, AI chatbot support, pet health records management, user management, pet care

educational resources, and feedback system. This holistic approach positions PawHub as a more intelligent and unified solution, offering pet owners a versatile and accessible digital assistant for effective pet care.

2.4 Software Development Methodologies

To guide PawHub's development, three classic Software Development Life Cycle (SDLC) approaches were reviewed, Waterfall, Agile, and Rapid Application Development (RAD). Each offers different trade-offs between predictability, flexibility, and speed, which are critical when integrating evolving AI features into a mobile pet-care app.

2.4.1 Waterfall Methodology

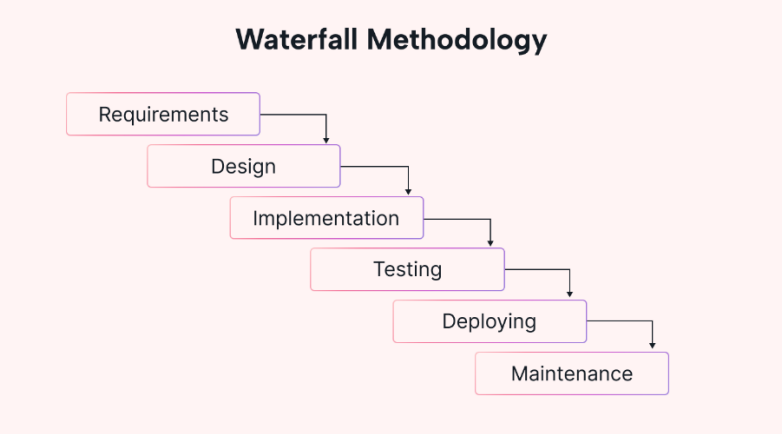


Figure 2.14: Waterfall Methodology (Motion, 2023)

Figure 2.15 illustrates the Waterfall model, a classic, step-by-step approach to software development that divides the project into separate, sequential stages. Despite the linear structure, there is usually some overlap or interconnection between the different phases as the project progresses. According to this method, every phase must be thoroughly completed and formally approved before advancing to the next one, thereby resulting in a detailed, sequential process that relies heavily on rigorous planning. Waterfall methodology has been widely adopted in environments where project requirements are stable and clearly defined from the outset. Its structured nature enables project teams to produce detailed documentation and set rigorous milestones, which can

serve as a roadmap for managing time, budgets, and resources effectively. According to Atlassian (2024), the Waterfall model unfolds as a series of well-defined steps, each of which is very important in following the software development process.

i) Requirements Phase

This phase is the foundation of the entire project. It involves gathering all the necessary conditions, features, and specifications required for the system. Stakeholders and end-users are consulted to obtain a comprehensive understanding of what the software must achieve. Every requirement is documented rigorously, usually in a Software Requirement Specification (SRS) document, which serves as the baseline for the entire project. This phase is crucial because any oversight or ambiguity here could propagate errors throughout later stages.

ii) Design Phase

Once the requirements are clearly documented, the Design phase begins. Here, the focus shifts to defining the technical architecture and system design based on the gathered requirements. This phase involves outlining the software architecture, designing the user interface, creating database schemas, and determining the necessary system components. Detailed design documents and Figures are produced to provide a clear blueprint that developers will follow during implementation. A well-executed design process contributes to the system's scalability, maintainability, and adherence to the established criteria.

iii) Implementation Phase

The actual coding happens during the Implementation phase. Developers use the selected programming languages and development tools to convert the design specifications into source code. This phase is typically carried out in multiple stages or modules, corresponding to the various components described in the design

documents. The focus during this phase is on writing efficient, error-free code that faithfully represents the design. While this stage brings the project closer to a working application, it heavily relies on the accuracy of the previous phases.

iv) Verification (Testing) Phase

Following implementation, the Verification phase is devoted to quality assurance. To make sure it satisfies the SRS standards, the system is put through a thorough testing process. This includes tests like acceptance testing, system testing, integration testing, and unit testing which are used to identify and address defects. Verification ensures that the software operates as intended and that issues from the earlier stages are addressed. However, deferring testing until after the bulk of the development is complete can lead to late discovery of bugs, which might be expensive and time-consuming to correct.

v) Maintenance Phase

The maintenance phase starts once the program has been deployed and put to use. To maintain the software's effectiveness over time, this phase contains regular updates, bug repairs, performance increases, and improvements. Maintenance can also include adapting the software to new environments or changing user requirements. Although it is not part of the initial development, effective maintenance is critical for ensuring the long-term reliability and relevance of the application.

2.4.1.1 Advantages of Waterfall Methodology

The clear, linear structure of the Waterfall model is one of its main advantages, it makes the development process quite predictable. With each phase explicitly defined, project managers and stakeholders can easily track progress against predetermined milestones, ensuring that the project remains on schedule and within budget. This high level of documentation not only facilitates communication among team members but also provides a comprehensive reference for future maintenance, helping to reduce long-term risks. Moreover,

because each phase has a distinct deliverable, it is relatively straightforward to identify where issues occur, at least from a management standpoint. The Waterfall model's methodical, sequential flow is most useful for projects when the goal is to provide a well-defined product with few features and requirements are not anticipated to alter.

2.4.1.2 Disadvantages of Waterfall Methodology

Despite its advantages, the rigid structure of the Waterfall model has significant drawbacks, especially in the context of modern, user-centric applications. One of the core disadvantages is its inflexibility, once a phase is complete, it becomes extremely challenging to revisit or modify previous work. This inherent rigidity means that any discovered mistakes or changed requirements later in the process can result in extensive rework and delays. The model also defers testing until after the implementation phase, which increases the risk of late defect detection, a scenario that can escalate costs and extend delivery timelines. Such a late discovery of problems limits the ability to respond quickly to shifting market demands or to integrate emergent user feedback, both of which are critical for the success of applications with complex, evolving functionalities.

2.4.2 Agile Methodology

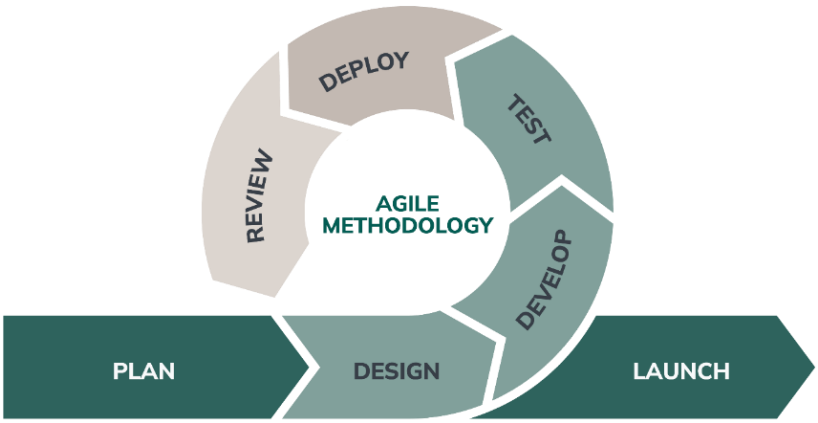


Figure 2.15: Agile Methodology (Agile software development: everything you need to know, 2024)

Agile software development is an incremental, iterative process that prioritizes adaptability, teamwork, and ongoing development. Agile divides the project into manageable, brief cycles called sprints, as opposed to the Waterfall model's linear progression. A subset of features are planned, designed, implemented, tested, and reviewed throughout each sprint, which typically lasts one to four weeks. By encouraging continuous interaction with stakeholder feedback, this iterative process makes sure that requirements modifications or new information may be promptly included. Originally popularized by the Agile Manifesto (Drumond, 2024). Agile has become a leading approach for projects where requirements evolve, and rapid adaptation is crucial.

i) Sprint Planning and Requirements Gathering

Every Agile sprint starts with a planning phase in which the developer determines the most important tasks and user stories based on stakeholder input and existing priorities. This phase involves defining the sprint backlog, where each user story is refined into actionable tasks with clear acceptance criteria. Although Agile promotes a flexible approach, meticulous planning at the beginning of each sprint ensures that the vision for the product remains aligned with user expectations. This ongoing requirement refinement is critical for an evolving application like PawHub, where the integration of AI features and dynamic pet health functionalities demands continuous user input.

ii) Design and Implementation

During the sprint, the design and implementation process occurs concurrently in an iterative cycle. In this phase, the chosen features are designed in detail and then translated into code using suitable programming languages and development tools. Agile encourages frequent communication, often through daily stand-up meetings or personal reviews to quickly identify issues and adapt designs as necessary. This phase benefits from modern techniques such as pair programming, code reviews, and frequent prototyping, which

collectively foster rapid development while maintaining high quality. For PawHub, this approach allows for the quick incorporation of user feedback into features like AI chat support and symptom diagnosis, ensuring that the application evolves to meet real-world user needs.

iii) Testing and Review

At the end of each sprint, Agile emphasizes rigorous testing and review of the newly developed features. Unit, integration, and user acceptability tests are regularly conducted as part of continuous integration, a fundamental approach that aims to identify flaws early. Post-sprint reviews or demos enable the developer to showcase working functionality and gather valuable feedback from stakeholders or early users. This constant testing cycle is crucial for PawHub, as it ensures that complex functionalities especially AI components are validated continuously, reducing the risk of late-stage issues and contributing to overall product robustness.

iv) Sprint Retrospective (Continuous Improvement)

Following the testing and review phase, the sprint concludes with a retrospective session where the developer reflects on the process. This self-assessment involves analysing what went well, identifying challenges encountered, and determining areas for improvement in subsequent sprints. By continuously refining development practices, Agile ensures that the process becomes progressively more efficient and aligned with project goals. For PawHub, this step is vital in ensuring that the evolving features, particularly those involving AI, are continually enhanced based on both technical insights and user feedback, promoting long-term success and high-quality performance.

2.4.2.1 Advantages of Agile Methodology

Agile's flexible and iterative structure makes it possible to quickly adjust to changing requirements, which is crucial for projects like PawHub that have changing requirements. Its continuous testing and integration practices facilitate early defect detection and prompt resolution, reducing risk and

enhancing product quality. In addition, Agile's emphasis on regular feedback ensures that user requirements remain central to the development process, leading to higher user satisfaction and product relevance. Although Agile typically results in less formal documentation, its focus on dynamic improvement and collaboration outweighs this drawback in projects where innovation and responsiveness are critical.

2.4.2.2 Disadvantages of Agile Methodology

Despite its many strengths, Agile can be resource-intensive since it demands constant collaboration and frequent adjustments. If modifications to this model are not properly managed, scope creep could occur which will affect the budgets and schedules. Agile's iterative process may also lead to inconsistent documentation quality when compared to more conventional methods like Waterfall, which could cause problems for maintenance or handovers in the future. For a solo developer or a small project, maintaining discipline and consistent progress across multiple short cycles can be challenging without formal team structures. However, when managed effectively, these challenges are mitigated by Agile's substantial benefits.

2.4.3 Rapid Application Development (RAD) Methodology

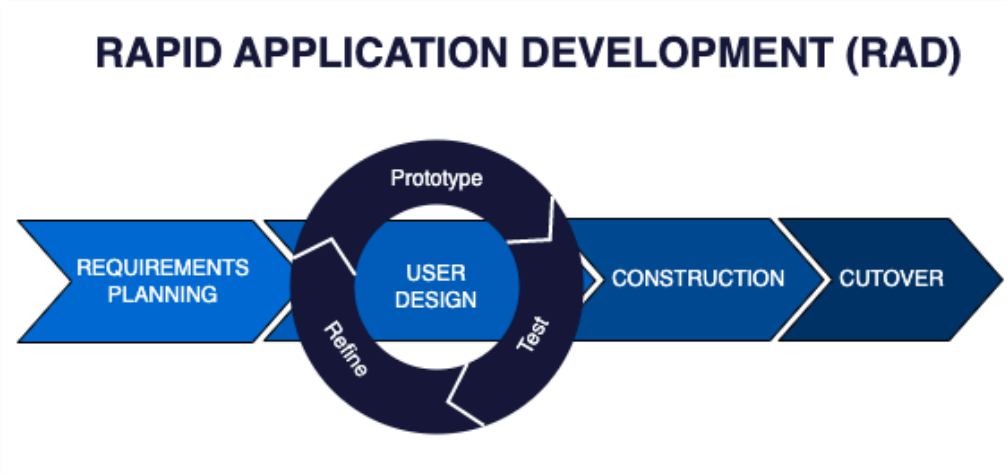


Figure 2.16: RAD Methodology (Rapid Application Development (RAD), no date)

The iterative process of Rapid Application Development (RAD) places a strong emphasis on quick user feedback and rapid prototyping. Rather than relying on comprehensive upfront planning, RAD emphasizes the creation of a functional prototype early on, which is then improved through several iterations. RAD facilitates parallel development in phases that allow developers to quickly modify and adapt the application based on immediate user responses. This methodology is particularly popular in scenarios where speed to market is critical, and requirements are expected to evolve rapidly (Kissflow, 2022).

i) **Requirements Planning**

In RAD, the initial planning phase is focused on outlining high-level requirements rather than detailed specifications. Stakeholders provide input on the desired features, which are then translated into user stories. This phase is less exhaustive than in traditional models, allowing developers to begin prototyping sooner.

ii) **User Design Phase**

During the user design phase, developers build preliminary models such as wireframes, prototypes, and mock-ups. These prototypes are shown to stakeholders and potential users for feedback. This phase is crucial for ensuring that the design meets user needs, although the rapid cycle may result in less formal documentation. For PawHub, iterative prototyping is valuable for testing AI functionalities like chatbots and symptom diagnosis interfaces, enabling early identification of usability issues.

iii) **Rapid Construction**

The rapid construction phase is where developers build the actual system components quickly. With a focus on speed, the emphasis is on delivering working software as soon as possible. Despite the benefit of early delivery, the focus on rapid development can sometimes compromise thorough testing, leading to quality issues if not managed carefully. For PawHub, rapid construction can validate

core functionalities quickly; however, the complexity of integrating secure health management and AI algorithms requires careful balance to avoid sacrificing reliability.

iv) **Cutover (Implementation) Phase**

The final phase in RAD involves the actual deployment of the application. It includes final testing, user training, and product documentation. Although RAD aims for a swift transition to production, a lack of detailed documentation and rushed testing may lead to unforeseen maintenance challenges post-deployment. For PawHub, while RAD allows for rapid prototyping and early market entry, the potential quality risks and scalability issues are significant concerns given the app's need for long-term stability and high accuracy in AI outputs.

2.4.3.1 Advantages of RAD Methodology

RAD offers notably fast development cycles, facilitating rapid prototyping and early user feedback. This accelerates time-to-market and enables developers to quickly iterate and improve the product. However, the methodology may sometimes result in reduced documentation and insufficient testing, which can compromise long-term maintainability and product quality. The need for a highly skilled and cohesive developer to manage rapid iterations is another drawback that may increase risk for complex applications like PawHub.

2.4.3.2 Disadvantages of RAD Methodology

The rapid pace and iterative cycles in RAD can lead to challenges such as scope creep and quality inconsistencies. With a strong focus on speed, some vital aspects especially extensive testing and thorough documentation may be overlooked. This is particularly risky for applications requiring robust AI integration and sensitive data management, as even minor oversights can impact overall reliability and user trust.

2.4.4 Comparison of Software Development Methodologies

Table 2.3: Comparison of Software Development Methodologies

Methodology	Advantages	Disadvantages
Waterfall	-Clear structure -Strong documentation -Predictable timeline	-Inflexible to changes -Late bug detection -Hard to adapt user feedback
RAD	-Fast prototyping -Quick user feedback -User-focused updates	-Weak documentation/testing -Risk of scope creep -Needs skilled management
Agile	-Flexible and adaptive -Early bug detection -User-driven development	-Resource-demanding -Scope creep if unmanaged -Lighter documentation

Agile methodology is particularly well-suited for developing PawHub because it enables continuous, iterative refinement of its complex, AI-driven features. Unlike Waterfall, Agile facilitates rapid integration of user feedback, allowing the app to evolve in real-time as user requirements and AI algorithms improve. Although RAD offers fast prototyping, its potential drawbacks in documentation and testing can compromise reliability, issues unacceptable for an application intended to deliver accurate, secure, and timely pet health insights. Agile's emphasis on regular sprint planning, frequent testing, and retrospectives ensures that each component, such as the AI chatbot and symptom diagnosis system, is rigorously validated and fine-tuned to meet high standards of quality, making Agile the most appropriate methodology for PawHub.

2.5 AI APIs for AI features

To create PawHub's intelligent features such as the chatbot and symptom diagnosis, three leading AI APIs were evaluated which is OpenAI, DeepSeek, and OpenRouter. Each offers distinct capabilities, pricing models, and levels of community support. The following subsections summarize their features, limitations, and fit for PawHub.

2.5.1 OpenAI API

Figure 2.17: Open AI API (Postman, 2025)

OpenAI has been a leader in general-purpose AI, with its GPT-4 model setting new standards for tasks like natural language generation, understanding, and reasoning (OpenAI, 2023). The OpenAI API provides access to powerful models such as GPT-3.5 and GPT-4, enabling developers to build sophisticated chatbots, summarization tools, classification engines, and more. For PawHub, these capabilities translate into advanced pet care chatbots, symptom evaluators, and FAQ assistants that can handle free-form queries with human-like fluency.

The primary strength of OpenAI's API is its unparalleled natural language understanding and generation quality, which allows the chatbot to carry context, ask clarifying questions, and generate highly nuanced outputs. Additionally, OpenAI provides structured endpoints for both completion and chat modes, supports embeddings for semantic search, and allows fine-tuning for domain-specific tasks. These features align well with the medical and caregiving nature of PawHub, where accurate, clear communication is critical.

However, OpenAI's major limitation is its cost and rate limits. While a free trial is available, it offers limited usage, and sustained usage incurs a pay-as-you-go pricing model that may quickly escalate in high-traffic applications. This is a significant consideration for apps like PawHub that expect real-time and frequent interactions. Furthermore, strict terms of service and compliance requirements especially around health data, may add legal or ethical complexity.

Still, OpenAI remains a top-tier choice for projects requiring cutting-edge performance, and for specific components of PawHub (e.g., onboarding assistant, complex symptom analysis), its reliability and accuracy can justify its cost when used strategically in hybrid setups.

2.5.2 DeepSeek API

Figure 2.18: DeepSeek API (TechNode Feed, 2025)

DeepSeek is an emerging player in the AI development landscape, known for its deep contextual inference and semantic search capabilities. It is engineered to excel at interpreting complex, ambiguous, or domain-specific language, which makes it a promising candidate for applications such as symptom-based diagnostic assistants or personalized care suggestions in PawHub. The core value proposition of DeepSeek lies in its ability to understand intent beyond surface-level language, allowing it to deliver tailored responses that are more relevant to the user's query (Guo *et al.*, 2024).

Technically, DeepSeek APIs support inference using their own optimized large language models, such as DeepSeek-V2. However, DeepSeek is still maturing. As of 2024, its API offerings are limited in scope and its developer resources, community activity, and framework integrations are less mature compared to OpenRouter and OpenAI. The documentation is minimal, and support forums are sparse, potentially increasing integration time for teams requiring precise control and debugging support. (Your First API Call | DeepSeek API Docs, 2025)

Additionally, DeepSeek's free tier is relatively constrained, offering limited API tokens per month and fewer customizations unless subscribed to a premium plan. In the context of PawHub, while DeepSeek may offer superior context retention and tailored inference capabilities, these benefits come at the cost of lower transparency, less documentation, and potential vendor lock-in due to the proprietary nature of its models.Nonetheless, DeepSeek holds high strategic potential as a supplementary AI tool in PawHub, especially for providing nuanced responses during complex symptom queries or multilingual expansion in the future. However, it may not yet be reliable enough as a standalone solution until the ecosystem becomes more robust.

2.5.3 OpenRouter AI API

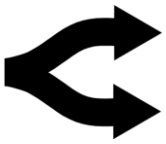


Figure 2.19: OpenRouter AI API (OpenRouter Logo PNG Vector (SVG) Free Download, 2025)

OpenRouter is a robust, developer-friendly API designed to bridge the gap between conversational AI systems and customizable, scalable chatbot solutions. Leveraging the power of transformer-based models, OpenRouter focuses on providing accessible and high-quality conversational AI capabilities that can be easily integrated into various applications, including customer support, personal assistants, and diagnostic tools.

OpenRouter's primary strength lies in its open-source approach, offering significant flexibility in how developers can implement its models. Its ability to work well with different languages and frameworks makes the platform ideal for those seeking a customizable and open solution. The API offers multiple pre-trained models suited for a wide variety of tasks such as question answering, summarization, and natural language processing.Additionally, OpenRouter's open-source nature enables seamless collaboration among developers and greater control over the chatbot's performance.

However, one of the limitations of lies in its relatively high infrastructure and resource demands. OpenRouter requires careful optimization in high-traffic environments, as excessive computational needs could lead to slow response times or increased operational costs. Additionally, while it's documentation are growing, it still does not match the maturity of other major platforms like OpenAI in terms of developer resources and community engagement (Principles - OpenRouter's Core Values, 2025).

Despite these challenges, OpenRouter remains a strong choice for applications like PawHub, especially when control over the AI model are prioritized. It can be a highly effective part of PawHub's AI-powered ecosystem. With the right infrastructure in place, OpenRouter's flexibility,

scalability, and open-source nature can significantly enhance PawHub's ability to provide dynamic and accurate pet care support.

2.5.4 Comparison of AI APIs for AI features

Table 2.4: Comparison of AI APIs for AI features

Criteria	OpenAI	DeepSeek	OpenRouter
Performance	Excellent NLP with GPT-4/GPT-3.5	Strong contextual understanding	Strong conversational AI across models
Cost	Pay-as-you-go; expensive at scale	Limited free tier	Open-source and cost-effective
Customization	Low (limited fine-tuning access)	Moderate	High (model flexibility, prompt control)
Ease of Integration	Well-documented SDKs & endpoints	Less documentation and slower onboarding	Simple REST API; growing dev support
Open-source Access	No	No	Yes
Community Support	Mature ecosystem	Emerging	Growing open-source developer base

All three APIs, OpenRouter, OpenAI, and DeepSeek provide unique strengths for powering the AI features in PawHub, an AI-driven pet care application. OpenRouter emerges as the most balanced and developer-friendly option, offering a cost-effective, open-source platform with high flexibility and customization. Its support for multiple models and prompt control makes it ideal for modular chatbot functionalities and pet-specific symptom analysis. OpenAI delivers superior natural language performance with models like GPT-4, making it suitable for handling highly complex queries. However, its

high cost and limited customization options make it not ideal. Meanwhile, DeepSeek shows promise in understanding context and intent, offering value for nuanced symptom interpretation or future multilingual support. However, its limited documentation, constrained free tier, and smaller community make it less suited compared to OpenRouter.

2.6 Backend Development Tools

The selection of an appropriate backend development tool is a critical component in ensuring that the PawHub application operates efficiently, securely, and at scale. Given the nature of the application which includes managing sensitive pet health records, real-time user interactions, and secure authentication choosing a backend solution that offers reliability, flexibility, and robust database features is paramount. This section reviews two popular Backend-as-a-Service (BaaS) platforms Supabase and Firebase, both of which are widely adopted for modern application development. These platforms provide integrated services for authentication, data storage, file management, and real-time functionality.

2.6.1 Supabase

Figure 2.20: Supabase Backend Tool (asierr.dev, 2024)

Supabase is an open-source backend platform that provides developers with a powerful suite of tools built around PostgreSQL, a proven and mature relational database. Branded as an “open-source Firebase alternative,” Supabase offers out-of-the-box support for authentication, real-time subscriptions, auto-generated APIs, and secure file storage, all using familiar SQL syntax. One of Supabase’s most significant strengths is its relational data model, which is essential for PawHub, where pet profiles, user accounts, appointment records, vaccination history, and symptom logs need to be properly linked and queried with high precision.

Supabase's authentication system supports email and password login, OAuth (Google, GitHub, etc.), and third-party integrations while maintaining row-level security (RLS), a PostgreSQL-native feature that allows developers to write policies that enforce fine-grained access control. This is especially useful for PawHub, where each user should only access their own pet's data and health records. Another benefit is the real-time capability, which is made possible via PostgreSQL's replication features. This allows changes in the database to be broadcast immediately to clients, ensuring live updates, such as tracking a pet's symptoms or syncing appointment reminders, appear instantaneously in the app.

The storage system in Supabase allows for secure and scalable file uploads, including photos of pets, scanned documents, and medical records. With the free tier offering up to 500MB of database storage and 50,000 monthly active users, Supabase is more than capable of supporting the early development and testing phases of PawHub without incurring high costs. Moreover, since Supabase is open source, it can be self-hosted if greater control or cost optimization is needed in the future. (Supabase, no date)

In addition, Supabase is developer-friendly with RESTful and GraphQL API generation, excellent documentation, and community-driven support. It integrates smoothly with modern front-end frameworks like React Native, making it a natural backend pairing for PawHub's tech stack. However, as a relatively newer player compared to Firebase, Supabase may have less mature tooling in some advanced use cases (e.g., serverless functions, analytics), though these gaps are rapidly narrowing as the platform evolves.

2.6.2 Firebase

Figure 2.21: Firebase Backend Tool (Setting up Firebase / Google Analytics, 2025)

Firebase is a backend platform developed by Google, offering a highly integrated and scalable BaaS solution. It includes a suite of services such as Cloud Firestore (NoSQL database), Realtime Database, Firebase Authentication, Firebase Storage, and Firebase Cloud Functions. Mobile developers have widely embraced Firebase for its user-friendly interface, dependable performance, and seamless integration with the Google Cloud ecosystem.

Firebase's Cloud Firestore is a document-based NoSQL database that stores data in collections and documents. This structure is particularly effective for apps with loosely structured data or where relationships between entities are minimal. However, in complex applications like PawHub, which require multiple relationships between pets, users, and medical records, the lack of structured querying compared to SQL may introduce challenges in data modeling and querying efficiency.

Firebase's authentication module is robust and user-friendly, supporting email/password login, federated identity providers (e.g., Google, Facebook), and custom authentication systems. Firebase also excels in real-time capabilities, with Firestore and Realtime Database enabling live syncing across devices, a strong feature for interactive apps like PawHub (Build Documentation | Firebase Documentation, no date).

One of Firebase's major selling points is its infrastructure scalability. Backed by Google Cloud, Firebase can effortlessly handle spikes in user activity and data loads, making it an excellent choice for apps expecting high traffic or viral growth. Additionally, Firebase includes analytics, A/B testing, and performance monitoring tools, which help developers optimize user engagement and app stability.

However, Firebase is not open-source, and its NoSQL nature may lead to difficulties in maintaining complex queries, particularly when working with hierarchical or relational data. Furthermore, as your app scales, Firebase's cost structure can become quite expensive especially when it comes to real-time database reads, writes, and storage usage. For PawHub, where each user may frequently interact with medical records and symptom logs, the cost model can become unpredictable without tight control over API usage.

2.6.3 Comparison of Backend Development Tools

Table 2.5: Comparison of Backend Development Tools

Feature	Supabase	Firebase
Database Type	PostgreSQL (SQL-based)	Firestore / Realtime Database (NoSQL)
Open Source	Yes	No
Hosting Options	Self-hosting or managed	Google-managed only
Security	Row-Level Security (RLS)	Role-based Rules
Real-time Support	Yes	Yes
Authentication	Built-in Auth (Email, OAuth, etc.)	Built-in Auth (Email, Socials, etc.)
Pricing	Generous free tier, transparent pricing	Free tier, can become costly at scale
Ideal For	Structured data, high control, SQL queries	Fast MVPs, scalable cloud-native apps

Supabase is the most suitable backend development tool for PawHub due to its relational database capabilities, transparent pricing, and open-source flexibility. Its integration with React Native, support for role-based access control, and real-time features make it ideal for managing pet health records, secure user access, and responsive symptom tracking. Unlike Firebase, which offers high scalability but can lead to unpredictable costs and complexity with NoSQL data modeling, Supabase provides a better balance of structure, control, and affordability particularly important for the long-term sustainability and data integrity needs of an AI-powered pet care app like PawHub.

2.7 Frontend Development Tools

Choosing the right frontend framework is a pivotal decision in mobile application development, especially for feature-rich apps like PawHub, which aims to offer real-time AI chatbot assistance, pet health record management,

and offline accessibility. The frontend must be responsive, performant, visually appealing, and compatible with backend APIs (such as Supabase). Two of the most widely used cross-platform development frameworks today are React Native and Flutter. Both frameworks let you maintain one codebase for Android and iOS, which accelerates development and cuts down on costs. However, they differ significantly in terms of architecture, flexibility, performance, learning curve, and ecosystem maturity. This section provides a comprehensive analysis of React Native and Flutter.

2.7.1 React Native

Figure 2.22: React Native (Okoone, 2025)

React Native is Meta's open-source framework that allows to write in application in JavaScript and React and have it run on both Android and iOS which can build once instead of coding separately for each platform. It bridges JavaScript and native mobile components through a mechanism called the "bridge", allowing code reuse while still delivering near-native performance. For an app like PawHub, which requires tight integration with local storage, and backend services, React Native provides robust support for both functional and UI layers.

A major benefit of React Native is its large developer ecosystem and the wide range of third-party libraries available, which significantly enhance the development process. This allows rapid development of features such as push notifications, file uploads, authentication flows, and navigation, all of which are crucial for a user-centric app like PawHub. Moreover, since React Native supports the same design principles as React.js, developers familiar with web development can quickly transition into mobile development, reducing onboarding time and improving productivity (Introduction · React Native, no date).

Another notable advantage is React Native makes development faster and smoother with its reload capability, showing code edits instantly without rebuilding everything. For PawHub, this enables faster prototyping and iteration on features like chatbot interfaces, health tracker screens, and form input for symptom analysis.

React Native also integrates well with Supabase, using fetch-based API calls and async data storage. Its support for SQLite and other local databases allows for implementing offline functionality, a critical feature for users in low-connectivity regions.

Despite its strengths, React Native does have limitations. The reliance on third-party modules can sometimes result in unstable dependencies or delayed support for new operating system updates. Additionally, in scenarios involving heavy UI activity like intricate animations, React Native may show slight delays due to its use of a JavaScript to access native features. However, these performance issues can be addressed through optimizations using native modules or libraries like Reanimated.

2.7.2 Flutter

Figure 2.23: Flutter (What is Flutter? Guide for Flutter App Development | Relia Software, no date)

Flutter, made by Google, makes it easier to build apps for different platforms using just one codebase. Unlike React Native, Flutter uses the Dart programming language and compiles directly to machine code. This architecture results in faster startup times and better runtime performance for animation-heavy or highly interactive apps.

One of Flutter's standout features is its "widget-centric" architecture. Every element on the screen is a widget, giving developers granular control

over the UI and allowing them to create highly customized designs. This is advantageous for apps with demanding UI requirements or complex interfaces.

Additionally, Flutter provides its own rendering engine, Skia, which eliminates dependence on the native platform's UI components ensuring consistent design and performance across devices Flutter is particularly strong in animation handling, making it ideal for apps with splashy transitions or micro-interactions. For a simple, functional app like PawHub, however, this level of visual finesse is not a primary requirement (Flutter documentation, no date).

The main drawbacks of Flutter lie in its relatively smaller community, larger app size, and learning curve associated with Dart, which is not as widely used as JavaScript. Integrating Flutter with platforms like Supabase may require additional configuration or community plugins, which might not be as mature or well-documented as React Native equivalents.

2.7.3 Comparison of Frontend Development Tools

Table 2.6: Comparison of Frontend Development Tools

Feature	**React Native**	**Flutter**
Programming Language	JavaScript	Dart
UI Performance	Native-like (bridged)	High performance (custom renderer)
Developer Ecosystem	Mature, large community	Growing, but smaller community
Library/Plugin Support	Extensive (NPM, GitHub)	Moderate
Learning Curve	Easier for web developers	Steeper due to Dart
App Size	Moderate	Larger by default
Ideal Use Case	API-heavy apps, MVPs, integration-rich	Custom UI apps, animation-focused apps

Supabase is the most suitable backend development tool for PawHub due to its relational database capabilities, transparent pricing, and open-source flexibility. Its integration with React Native, real-time features make it ideal for managing pet health records, secure user access, and responsive symptom tracking. Unlike Firebase, which offers high scalability but can lead to unpredictable costs and complexity with NoSQL data modeling, Supabase provides a better balance of structure, control, and affordability, particularly important for the long-term sustainability and data integrity needs of an AI-powered pet care app like PawHub.

2.8 Summary

The literature review underscores the transformative impact of AI and mobile technology on the pet care industry, enabling tools like chatbots, symptom diagnosis systems, and veterinary decision-support platforms. Despite advancements, existing solutions face limitations such as fragmented health monitoring, reliance on internet connectivity, and insufficient accuracy in AI-driven diagnostics. Applications like TTcare (image-based analysis) and PetVet AI (chatbot support) address niche needs but lack holistic integration, while PetVitality and 11Pets focus on record-keeping without robust AI integration.

PawHub aims to bridge these gaps by offering a comprehensive, AI-powered solution combining real-time symptom analysis, multi-user accessibility, and offline functionality. The app integrates an AI chatbot for instant guidance, API-based symptom diagnosis, and health record management, ensuring reliability and accessibility even in low-connectivity environments.

Agile methodology was selected for development due to its flexibility, iterative refinement, and alignment with evolving AI features. OpenRouter emerged as the optimal AI API, balancing cost-effectiveness, customization, and conversational accuracy, while Supabase was chosen for its relational database capabilities, security, and scalability. React Native's mature ecosystem and seamless backend integration further support PawHub's cross-platform performance.

By synthesizing AI-driven diagnostics, user-centric design, and offline accessibility, PawHub positions itself as a pioneering solution for proactive, informed pet healthcare, addressing critical gaps in the market and enhancing the connection between pet owners and veterinary professionals.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

The follow section shows the approach and schedule that will direct the development of PawHub, an AI-powered mobile app designed to help pet owners manage their pet's health. The project will follow the Agile methodology, allowing for flexible, iterative development through continuous feedback and adaptive sprint planning. PawHub aims to address common challenges in pet care, including early symptom detection, digital health recordkeeping, and access to veterinary advice. Key features will include an AI chatbot, a symptom checker, multi-pet profiles, and educational content. The app will be built using React Native for the frontend and Supabase for the backend, with AI features powered by the OpenRouter AI API. The table below provides a summary of the project's resource allocation:

Table 3.1: Resources Allocation

Resources	**Allocation**
Personnel	Independently manage all aspects of the project, including requirement gathering, design, development, and integration. Additional testers will be involved during the testing phase to validate functionality and identify potential defects.
Time	The project will be carried out over six months. The first three months will be devoted to requirement analysis, literature review, and UI prototyping, while the remaining three months will focus on development, testing, and refinement.
Costs	The project will incur no financial cost. All tools and platforms used including React Native, Node.js, Supabase, the OpenRouter API (free tier), and development software will be open-source or freely available.
Materials	Software to be used will include Figma, Visual Studio Code, Android Studio Emulator, GitHub, Node.js and React Native

	libraries. AI capabilities will be integrated using the OpenRouter API. Development and testing will be performed using a personal laptop and smartphone.

3.2 Collecting Requirements

This section will focus on the process of identifying and collecting essential requirements for the development of PawHub, an AI-powered pet healthcare assistant mobile application. By adopting the mindset of a pet owner and end user, this stage aim to uncover the necessary features, functionalities, and pain points that the app must address to deliver a seamless and intelligent user experience.

To ensure the requirements are grounded in actual user needs, a survey-based approach will be adopted. The survey will target active pet owners, particularly those responsible for tracking pet health, handling emergencies, and ensuring regular veterinary checkups. Responses will be collected, covering areas such as current pet care habits, difficulties in health management, expectations of AI features, and mobile app usage preferences.

In addition, four existing pet care mobile applications, TTcare, PetVet AI Pet, PetVitality, and 11Pets has be reviewed to assess their core functionalities and identify common shortcomings. Features such as symptom checkers, health tracking, appointment reminders, AI chatbots, and document storage are evaluated. While each of these applications may offer partial solutions, none provides a comprehensive system that integrates AI-driven diagnosis, health record management, personalized care tips, and offline accessibility within a single platform.

Several common limitations and usability challenges are observed. The absence of integrated AI chatbots and fragmented health record management require users to change from one app to another to manage different functions of pet care. Some applications also lack interactive features such as reminders, multi-pet profile support, or emergency guidance. Moreover, offline functionality, crucial for pet owners in remote or rural areas were missing from most solutions. Based on the app analysis, the following requirements has been identified as critical for the success of PawHub:

i) AI-powered chatbot to answer pet care queries and provide general advice.
ii) Symptom checker for early diagnosis based on user input (text-based, not image-based).
iii) Secure digital pet health records including vaccination logs and medical history.
iv) Offline access to essential health data and records.
v) Personalised alerts and reminders for pet health records.
vi) Support for multi-pet profiles with individualized health logs.
vii) FAQs for first time pet owners.
viii) Clean, user-friendly interface suitable for a broad range of users.
ix) Built-in feedback system to support ongoing feature refinement and updates.
x) Real-time alerts for symptom concerns and vaccination reminders.

However, it is important to acknowledge the gap between ideal functionality and realistic development capabilities. Advanced features such as real-time vet chat, image-based diagnostics, or AI analytics dashboards may require substantial computing resources and infrastructure support. Given the limitations in time, manpower, and budget, a phased development strategy will be adopted.

The initial version of PawHub will focus on delivering the most essential and feasible features, the AI chatbot, text-based symptom checker, digital health records, offline data accessibility, and a responsive, user-friendly interface. These core modules will address the most pressing needs of users while staying within the project's technical and temporal constraints. Additional enhancements such as emergency vet locators, advanced APIs, or community support channels may be explored in future iterations once the foundation is successfully built and validated. In the next section, the collected requirements will be analyzed in detail to determine their feasibility within the available resources, development tools, and timeline constraints.

3.3 Analysis of Requirements

This section will evaluate the feasibility of the key functional requirements identified in Section 3.2, considering the actual constraints of the project. These constraints will include limited manpower (as a solo developer), a tight three-month development timeframe, no allocated budget, and the reliance solely on open-source tools and available hardware resources. The purpose of this analysis will be to realistically assess which features can be implemented effectively during the current development phase of PawHub.

i) AI Chatbot Integration

The AI chatbot will be considered a core feature of PawHub, intended to assist users in accessing general pet care information. Upon analysis, this requirement will be achievable using free-tier access from open-source APIs such as OpenRouter. The chatbot will be trained to provide common responses to pet health inquiries. However, integrating premium APIs like OpenAI for more advanced interactions will be considered unrealistic due to API usage fees that cannot be sustained under a zero-cost constraint. Therefore, the basic chatbot will be implemented using OpenRouter with the potential to upgrade in future iterations.

ii) Symptom Checker API Integration

Implementing a text-based symptom checker using a third-party API will be feasible within the project's limitations. Free or trial-tier APIs, such as self-built logic models hosted via Supabase functions, will allow for symptom input and provide general guidance. As this feature will not require complex image recognition or machine learning training from scratch, it will be both cost-effective and achievable within the project timeline.

iii) Pet Health Record Management

A core feature of PawHub will be the ability to store vaccination, illness, and treatment records digitally. With Node.js and Supabase calls as the backend, an open-source Firebase alternative, this functionality will be efficiently implemented using structured database tables. Supabase's

authentication and PostgreSQL database services will be available under a generous free tier, making this feature highly feasible within current resource constraints.

iv) Offline Functionality

Providing limited offline access to pet health records and educational content will be partially achievable. Both features rely on Supabase for cloud storage, which requires an active internet connection for real-time updates and retrieval. However, to enhance usability during low or no connectivity, offline caching will be implemented using React Native's local storage (e.g., AsyncStorage). This approach allows recently accessed health records and educational articles (scraped and stored in Supabase) to be temporarily saved on the user's device. While this enables basic read-only access when offline, any updates or AI-related functions such as chatbot queries or symptom analysis will still require a stable internet connection due to their dependency on external APIs and cloud databases.

v) Personalized Alerts and Reminders

Features like vaccination reminders, feeding schedules, and wellness notifications will initially be considered useful, but implementing a full-fledged background scheduling system with push notifications will require more time and backend logic than available. As a compromise, users will be able to manually input and view upcoming vaccination reminders, but auto-generated alerts and schedule notifications may be excluded from the first version.

vi) Multi-Pet Profile Support

Allowing users to register multiple pets with individual profiles will be a scalable feature that aligns well with Supabase's relational database structure. The development of this functionality will be feasible and will be included in the initial build, although advanced segmentation features (like charts or analytics) will be deferred to future updates.

vii) Educational Content and Tips

Including a curated list of pet care tips and articles will be considered realistic. This feature will be achieved by dynamically retrieving web-scraped content and storing it in Supabase storage, eliminating the need for manual hardcoding or local JSON files. The content, categorized into topics like nutrition, training, and general pet care, will be regularly updated using external scraping scripts and then accessed by the app via Supabase queries. As this approach does not rely on complex real-time APIs and offers flexibility for offline caching, it will remain well within the project's scope and achievable using the existing tech stack.

viii) Clean, User-Friendly UI

Given the project scope and the use of React Native with design tools, implementing a visually appealing and easy-to-navigate UI will be highly feasible. Existing UI libraries such as React Native Paper and ShadCN will offer pre-built components that reduce development time. This will align well with both the solo development structure and the time constraints.

ix) Feedback System

A basic feedback collection mechanism (e.g., a form submitted to Supabase) will be achievable. However, advanced analytics or rating systems that require detailed backend processing will be reserved for future adjusments.

x) Use of Paid APIs and Advanced AI Models

Advanced services such as OpenAI's GPT-4 API, real-time veterinarian consultations, or proprietary ML models will be deemed out of scope for the current phase. These features will often require payment structures and infrastructure beyond the project's capabilities and may be explored once the app reaches a more mature state or receives funding.

In conclusion, the requirements identified for PawHub will be carefully analyzed against the available resources. Most essential features

including an AI chatbot (via OpenRouter), health record storage (via Supabase), symptom checker API integration, multi-pet support, and offline access to static data will be found realistically implementable. However, more resource-intensive features such as automated schedule reminders, image-based diagnosis, and full offline AI capabilities will be excluded or scaled down for future iterations. This phased approach will allow the app to remain functional, user-friendly, and impactful within the available budget and time constraints.

3.4 Software Development Methodology Used

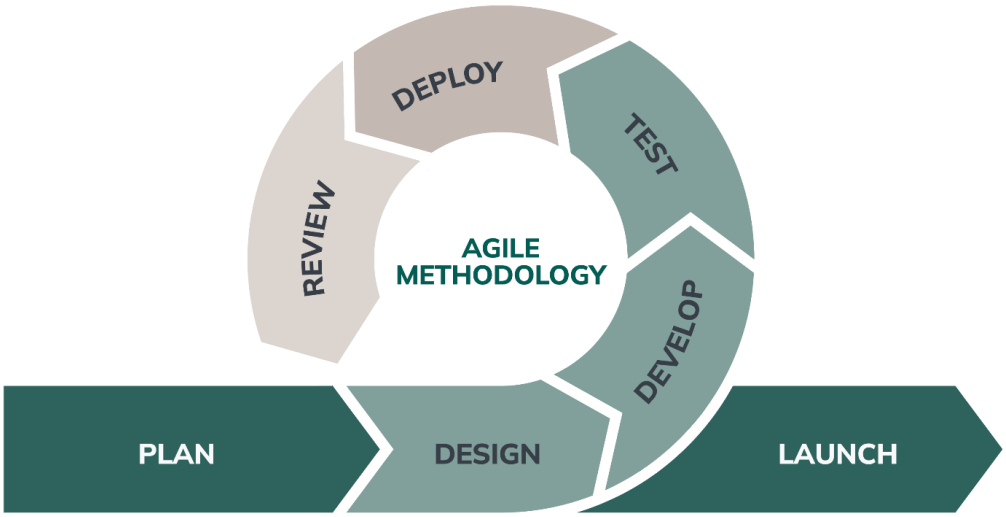


Figure 3.1: Agile Methodology (Agile software development: everything you need to know, 2024)

The development of PawHub will be guided by the Agile software development methodology, which is particularly well-suited for projects with evolving requirements and a focus on continuous improvement and delivery. Agile is a great option for creating a mobile application that incorporates cutting-edge AI-powered features like an AI chatbot, a symptom diagnosis tool, and pet health record management modules since it places an emphasis on teamwork, adaptability, and responsiveness. By adopting Agile, the project will be able to accommodate changes in user feedback and technical direction while maintaining a consistent development rhythm that leads to the creation of a high-quality, user-centric product.

According to Kaleel and Harishankar (2013), agile methodology supports mobile software engineering by addressing challenges unique to Android development. Agile development will be structured into incremental iterations called sprints, each typically lasting between two to four weeks. Every sprint will consist of essential phases such as planning, design, development, testing, and review. These repeated cycles will allow for gradual refinement of each feature, enabling early detection of issues and rapid adaptation to shifting user expectations and technical considerations. The following subsections will explain how Agile will be implemented throughout the various phases of PawHub's development.

3.4.1 Requirement Analysis and Sprint Planning

The first step in the Agile process will involve thorough requirement analysis to ensure that the application features align with user expectations. For PawHub, this phase will begin with preliminary research into common challenges faced by pet owners, including health management difficulties, a lack of reliable pet health information, and delays in recognizing symptoms that require veterinary attention.

Surveys and comparative analysis of existing pet care applications will be conducted to better understand what users want and what gaps currently exist. Insights from this phase will inform the definition of key features such as the AI chatbot, the symptom diagnosis tool, and a digital pet health record module. Based on these findings, Sprint Planning will be conducted to define clear goals, allocate tasks, and establish a development roadmap for the upcoming sprint cycles.

3.4.2 Design and Prototyping

The design and prototyping stage will begin after the requirements analysis. In order to visually organize the application's layout and interaction flow, user interface (UI) and user experience (UX) designs will be created at this phase. According to Käpyaho and Kauppinen (2015), using prototypes in agile development facilitates better understanding of requirements and fosters iterative refinement. The focus will be on delivering an intuitive and accessible user experience, particularly for pet owners who may not be tech-savvy.

Low-fidelity wireframes will be created using tools like Figma to visualize each screen's layout, content arrangement, and navigation logic. Concurrently, the system architecture will be planned to outline the communication flow between components, specifically between the front-end which will be built using React Native and the back-end services such as Node.Js, Supabase for database and authentication, OpenRouter API for chatbot and symptom-related processing and Resend for email reminders.

As part of Agile's iterative nature, this design process will involve multiple review cycles. Feedback from supervisors and early users will be collected at the end of each sprint to refine the designs and align them closely with usability goals. Any identified issues will be incorporated into subsequent design improvements.

3.4.3 Development and Integration

Once the prototypes and architecture have been reviewed and validated, the development and integration phase will begin. This phase will involve the actual coding and implementation of PawHub's core modules, structured across four main sprints.

3.4.3.1 Sprint 1

Sprint 1 will focus on user interface (UI) development and navigation setup. During this period, the core UI screens such as Login, Register, Home, Pet Dashboard, and Profile will be implemented using React Native. A navigation flow will be created between the screens to support smooth transitions using appropriate navigation components. The UI layout will be tested and reviewed by the supervisor, after which necessary adjustments will be made to improve responsiveness, spacing, and visual hierarchy.

3.4.3.2 Sprint 2

Sprint 2 will concentrate on backend implementation and user authentication. Supabase will be used to set up authentication logic, enabling secure registration, login, and user sessions. A database schema will be established for both user and pet profiles. CRUD (Create, Read, Update, Delete) operations will be developed for managing user data and pet data, including

health records and vaccination details. User flow tests will be carried out to ensure seamless interaction, and any authentication or session-related bugs will be resolved during this sprint.

3.4.3.3 Sprint 3

Sprint 3 will focus on the implementation and enhancement of PawHub's AI-driven features, aiming to significantly elevate the app's intelligence, user support capability, and educational value.

i) AI Chatbot Integration Module (Multi-Model via OpenRouter)

In this phase, the AI chatbot interface will be connected to multiple specialized large language models (LLMs) using the OpenRouter API. Users will be given the ability to select from three optimized AI models, each with unique conversational strengths, depending on the nature of their enquiry (e.g., general care, emergency advice, behavioral issues). A model selector dropdown will be implemented within the chat interface, dynamically routing user queries to the chosen AI in real-time.

Prompt engineering techniques will be applied to tailor the input prompts for each model, ensuring that the AI maintains a strict context around pet-specific conversations. Special prompt tokens, contextual conditioning, and instruction fine-tuning will be used to make the chatbot reject irrelevant questions (e.g., human-related queries) and stay focused on pet care advice. Error handling will also be implemented to gracefully manage API failures, rate limits, and fallback model selection to ensure uninterrupted user experience.

ii) Symptom Checker Module (AI Risk Assessment Tool)

The Symptom Checker will allow users to input a structured set of symptom data through interactive forms. The form will support input type such as text fields for user to input their pet symptoms. Upon submission, the collected symptom profile will be compiled into a dynamically generated prompt and sent to a selected OpenRouter LLM

that specializes in clinical reasoning or healthcare inference. The AI model will return a risk-assessed recommendation, which will then be interpreted into a color-coded risk indicator:

- Green: Mild, Monitor at home
- Yellow: Moderate, Suggest vet consultation if symptoms persist
- Red: Severe, Immediate veterinary attention required

iii) Educational Resource Module (Dynamic Web-Scraped Content)

The Educational Resources screen will be dynamically retrieved content using web scraping server. Scheduled scraping scripts (run externally) will fetch pet care articles, expert advice, and how-to guides from verified sources (e.g., ASPCA, PetMD, VCA Hospitals). The articles will be categorized intelligently into various core segments.

A searchable UI will be implemented to allow users to filter and find articles based on keywords or categories. Each article card will display a title, snippet, publication date, and a quick link to full details inside the app's reader view. Metadata tagging (e.g., dog health, cat grooming, puppy training) will be assigned automatically, further enhancing search relevance.

3.4.3.4 Sprint 4

Sprint 4 will serve as the final sprint of the development phase. This sprint will involve the implementation of additional features such as offline support (using local storage, where feasible), a feedback collection module, and final user experience testing with real pet owners. Integration testing will be carried out across all modules to ensure consistent data flow between the front end, database, and AI services. Performance improvements and final bug fixes will also be prioritized, culminating in the preparation of the final working prototype for demo and evaluation purposes.

The entire development phase will follow an incremental and flexible approach. Each sprint will result in a usable build of the application, allowing

for continuous testing and refinement. This process will help ensure that PawHub is delivered with reliable functionality, user-friendly design, and meaningful AI-driven features for pet healthcare support.

3.4.4 Testing and Optimization

Once the core features of PawHub have been successfully developed and integrated, the project will proceed into the testing and optimization phase. This phase will be essential in identifying and resolving bugs, validating the app's functionality, and ensuring that each component meets the defined user requirements. Testing will be conducted iteratively throughout the development cycle, aligning with the Agile methodology. This will include unit testing, integration testing, and system testing, applied incrementally as each feature is completed.

The AI-related modules, specifically the chatbot and the symptom diagnosis tool will undergo detailed validation to assess the accuracy of their responses. Simulated unit test cases will be designed to reflect real-world pet health issues, allowing the system's logic and prompt outputs to be evaluated for reliability and relevance.

In addition to formal testing techniques, informal usability testing sessions will be conducted with actual pet owners. Their feedback will provide valuable insights into the app's design, navigation, and overall user experience. Any issues discovered during these sessions will be documented and addressed, leading to a series of refinements. The goal of this phase will be to ensure PawHub is not only technically stable but also intuitive and user-centric.

3.4.5 Review and Continuous Improvement

As part of Agile methodology, PawHub's development will embrace a culture of continuous review and enhancement. After the completion of each sprint, a sprint review meeting will be conducted to assess the progress made, evaluate whether sprint objectives have been met, and gather feedback from the supervisor. This feedback will be used to guide the priorities and improvements in the subsequent sprint.

Upon nearing the project's completion, a final review will be conducted. This will serve as the last quality checkpoint to confirm that all

functionalities are implemented correctly, the system performs smoothly, and the app provides a cohesive and seamless experience. Final adjustments and polishing will be performed based on the cumulative feedback, ensuring PawHub meets its full potential before final deployment and presentation.

3.5 Development Tools Used

In this section, all the development tools that will be used throughout this project to achieve the application's success will be discussed.

3.5.1 React Native

Figure 3.2: React Native (Okoone, 2025)

React Native will be selected as the primary front-end development framework for PawHub due to its cross-platform capabilities and efficient component-based architecture. React Native, created by Facebook, is a time-and money-saving option for developing mobile apps since it enables developers to write JavaScript code that compiles to native components for both iOS and Android. This will be especially beneficial for a solo developer working under time and resource constraints, as it will remove the need to build separate native apps for different platforms (Introduction · React Native, no date).

Another advantage of React Native will be its support for a rich ecosystem of libraries and modules, which will allow for rapid integration of UI components, state management systems, and third-party APIs. Libraries like React Navigation and Native Base will be utilized to streamline navigation between screens and enhance user interface consistency. React Native will also provide hot-reloading capabilities, which will make it easier to test UI changes in real time without restarting the app, thereby improving productivity during sprints.

React Native will prove to be a scalable and adaptable choice for PawHub, especially as it will need to integrate with various back-end services like Supabase and AI APIs like OpenRouter. Its flexibility will allow for smooth communication with external APIs using tools like Axios and Fetch, enabling seamless real-time interactions for features like chatbot messaging and pet profile updates. The app's ability to function smoothly across devices will further validate the selection of React Native as the core framework.

3.5.2 Visual Studio Code

Figure 3.3: Visual Studio Code (Hill, 2024)

For building PawHub, Visual Studio Code (VS Code) will be the main tool used to write, debug, and organize the app's code. It's a lightweight and flexible editor made by Microsoft, chosen because it's fast, easy to customize, and comes with smart features like code suggestions and helpful extensions. Its customizable interface and built-in terminal will provide an efficient and personalized development environment tailored to the project's workflow.

The wide variety of extensions available in VS Code will significantly enhance productivity. Tools such as Prettier for code formatting, ESLint for syntax checking, and React Native Tools for debugging will make the development process smoother and more efficient. Integrated Git support will allow real-time version control and commit management directly within the IDE, reducing context switching and streamlining the development process. The built-in terminal will make it easy to execute Supabase CLI commands, run build scripts, and interact with Node.js-based packages (Visual Studio Code, 2023).

Throughout the development of PawHub, VS Code's intuitive UI, performance, and extension-rich ecosystem will enable a faster development pace, especially during tight sprint cycles. Real-time linting and syntax highlighting features will help reduce logical and syntactical errors early in the process. Its seamless integration with React Native and support for collaborative features (such as GitHub Copilot) will make it an indispensable tool in delivering a high-quality application.

3.5.3 Android Studio Emulator

Figure 3.4: Android Studio Emulator (Najjar, 2023)

The main testing environment for PawHub's Android version will be the Android Studio Emulator. It enables programmers to simulate different Android devices and OS versions. This will be particularly useful for testing responsiveness, functionality, and UI behaviour across different screen sizes and resolutions (Run apps on the Android Emulator, no date).

Using the emulator will help identify device-specific issues early in the development process without requiring access to a wide range of physical devices. This will be critical for ensuring compatibility and consistency, especially for features like pet profile creation, symptom checker interactions, and chatbot UI layouts. Various use-case scenarios such as poor network conditions, navigation between screens, and user input validation will be tested to ensure a seamless experience under real-world constraints.

While React Native will support hot-reloading on physical devices, the Android Studio Emulator will provide a safe and efficient sandbox for debugging major UI and system-level behaviours. It will also allow simulation of hardware features like camera, location services, and battery performance, which will be essential for validating features like health reminders. The emulator will play a crucial role in streamlining the QA and testing process.

3.5.4 GitHub

Figure 3.5: GitHub (GitHub Logo Download - SVG - All Vector Logo, 2016)

GitHub will be used as the version control and code collaboration platform for PawHub. Although the project will be developed individually, GitHub will ensure that all changes are systematically tracked, enabling effective source code management and a robust backup strategy. By committing updates regularly, the developer will be able to roll back to previous versions when necessary, reducing the risk of irreversible errors or data loss (GitHub, 2024).

GitHub's project board and issue-tracking features will be leveraged to plan tasks, organize sprint goals, and track bugs or enhancement requests. This will make it easier to manage workload, prioritize features, and maintain a structured development pipeline aligned with the Agile methodology. Milestones and branches will also be used to isolate features, such as separating the chatbot development branch from the pet profile UI updates.

Additionally, GitHub will make it possible to host the project in a secure, cloud-based environment. This will ensure continuous access to the codebase across different development machines. The repository will also serve as a medium for showcasing the development history, which can be shared with supervisors, testers, or future collaborators. Overall, GitHub will support both the technical and organizational aspects of PawHub's development lifecycle.

3.5.5 Node.Js

Figure 3.6: Node.Js Backend Tool (Node.js Development Services Company | Hire Node.js Developers, 2016)

Node.js will be used as the backend runtime environment for PawHub to enable secure, server-side logic and centralized API management between the React Native frontend and external services. Node.js, enables developers to use JavaScript in front-end and back-end applications, promoting code consistency, faster development cycles, and easier debugging, especially beneficial for a solo developer managing a full-stack application.

The Express.js framework, running on top of Node.js, will be utilized to define RESTful API endpoints that handle authentication, data validation, session management, and integration with third-party APIs. This architecture ensures that sensitive operations such as password updates, health record modifications, and AI prompt handling are processed securely on the server rather than directly from the client, reducing exposure to potential attacks like injection or unauthorized access (OpenJS Foundation, 2017).

A primary benefit of Node.js is its event-driven and non-blocking I/O approach allows for great efficiency and scalability when managing several concurrent requests. This is critical for features like real-time chat responses and scheduled email reminders. Additionally, middleware support in Express allows for modular implementation of functionalities such as JWT-based authentication, rate limiting, request logging, and error handling, ensuring robustness and maintainability throughout the project lifecycle. With strong community support, extensive npm packages, and compatibility with modern web standards, Node.js serves as a reliable and efficient backbone for PawHub's backend infrastructure.

3.5.6 Supabase

Figure 3.7: Supabase Backend Tool (asierr.dev, 2024)

Supabase will be chosen as the back-end platform for data storage, user authentication, and real-time synchronization. As an open-source Firebase alternative, Supabase will provide a scalable and developer-friendly suite of tools, including a PostgreSQL database, RESTful APIs, and real-time listeners. For PawHub, Supabase will handle user accounts, pet profiles, and health records with seamless integration into the React Native frontend.

Supabase Auth will be used to manage secure login and registration using email and password. It will also provide session management features, ensuring that user data can be accessed securely without the need to build a custom authentication backend. Its integration with JWT tokens and role-based permissions will enable fine-tuned access control over pet data and user-specific resources (Supabase, no date).

In terms of database functionality, Supabase's real-time PostgreSQL database will allow for efficient CRUD operations on pet and health-related data. The Supabase client library will work seamlessly within the React Native environment, making it easier to connect the frontend and backend without the complexity of a traditional server architecture. Its hosted dashboard will provide insights into database activity, logs, and performance metrics, supporting scalability and debugging.

3.5.7 OpenRouter AI

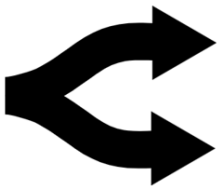


Figure 3.8: OpenRouter AI API (OpenRouter Logo PNG Vector (SVG) Free Download, 2025)

The OpenRouter AI API will be used to power PawHub's AI chatbot, offering advanced natural language processing capabilities through access to a wide range of leading large language models (LLMs). OpenRouter serves as a unified platform that connects to multiple powerful AI models, such as GPT-4, Claude, and Mixtral, allowing flexible and intelligent conversational experiences without the need to build custom models from scratch.

By integrating OpenRouter's API via HTTP requests, PawHub will be able to interpret user queries about pet care, generate accurate and context-aware replies, and provide seamless real-time interaction within the app. OpenRouter's support multiple models within one API key aligns well with PawHub's aim to achieve different AI models within one application and it also allow prompt engineering that will help with fine-tuning of how the chatbot responds, making interactions more specific to pet care needs.

Additionally, OpenRouter offers competitive pricing, a generous free tier for developers, and the flexibility to select different AI models according to usage scenarios, making it highly suitable for PawHub's development that requires multiple models and scaling phases. Its clear documentation and growing developer community ensure reliable support during the integration process. OpenRouter will serve as a robust and scalable solution for building intelligent dialogue systems in PawHub, providing users with trusted, pet-specific guidance instantly (Principles - OpenRouter's Core Values, 2025).

3.5.8 Resend

Figure 3.9: Resend Email API (Resend, 2025)

Resend will be integrated into PawHub as the email delivery service responsible for sending automated health reminders to users based on upcoming pet care events such as vaccinations, deworming schedules, and vet check-ups. Designed specifically for developers, Resend offers a simple, reliable, and cost-effective solution for transactional email delivery, making it an ideal choice over traditional SMTP providers or bulk email services unsuitable for personalized notifications.

A major advantage of Resend is its developer-first approach, featuring clean documentation, instant setup with API keys, and real-time delivery analytics through its dashboard. It also provides detailed logs for troubleshooting failed deliveries and supports email verification to ensure deliverability and compliance with anti-spam policies. Unlike other email platforms that require complex configuration or hidden costs at scale, Resend offers a generous free tier and transparent pricing, fitting perfectly within the project's zero-budget constraint (Managing Emails - Resend, 2025).

Furthermore, Resend integrates smoothly with Node.js via lightweight HTTP clients like Axios, enabling quick implementation without bloating the codebase. Its reliability and ease of use make it a powerful tool for enhancing user accountability and preventive pet care. By automating timely reminders through Resend, PawHub reinforces responsible pet ownership and delivers added value beyond basic digital record-keeping.

3.5.9 Figma

Figure 3.10: Figma (Interino, 2022)

Figma will serve as the primary tool for designing the PawHub prototype screens and user interface. The goal is to develop an app that feels smooth, intuitive, and user-friendly for all pet owners, and Figma supports this objective by enabling design visualization prior to development (Figma Design – Figma Learn - Help Center, 2024).

As a cloud-based platform with real-time collaboration capabilities, Figma facilitates rapid design iteration across key screens such as Login, Register, Home, Pet Dashboard, AI Chatbot, and Profile. Its ability to support both low and high-fidelity mockups allows for flexible experimentation with layouts and ensures a seamless user journey throughout the app.

Despite its design-oriented nature, Figma remains highly accessible to those with a development background. Its drag-and-drop interface, reusable components, and built-in prototyping tools make it easy to simulate the look and feel of the application. Once the design is finalized, it will act as a visual reference during the development phase in Visual Studio Code, guiding the implementation of each React Native component and ensuring visual consistency and usability aligned with the original design objectives.

3.6 Project Plan

This section outlines the overall project planning strategy. The WBS breaks down the project into manageable tasks and sub-tasks to ensure systematic progress and clear role distribution. The Gantt chart presents the scheduling of these tasks across the project duration, highlighting key milestones, dependencies, and deadlines to ensure effective time management and project tracking.

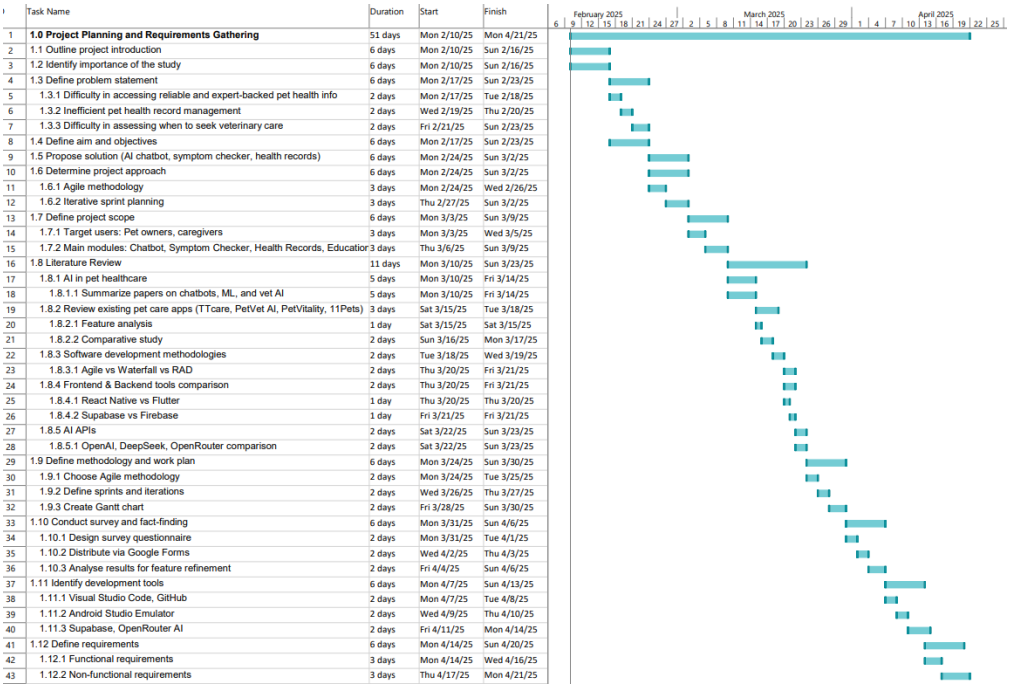
3.6.1 Work Breakdown Structure (WBS)

Figure 3.11: Work Breakdown Structure Diagram

3.6.2 Gantt Chart

The Gantt chart presents the scheduling of these tasks across the project duration, highlighting key milestones, dependencies, and deadlines to ensure effective time management and project tracking.

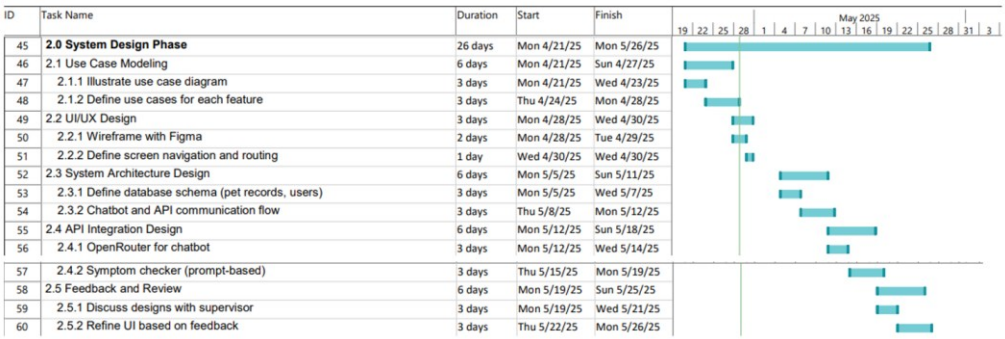
3.6.2.1 Project Planning and Requirements Gathering



	Task Name	Duration	Start	Finish
1	**1.0 Project Planning and Requirements Gathering**	51 days	Mon 2/10/25	Mon 4/21/25
2	1.1 Outline project introduction	6 days	Mon 2/10/25	Sun 2/16/25
3	1.2 Identify importance of the study	6 days	Mon 2/10/25	Sun 2/16/25
4	1.3 Define problem statement	6 days	Mon 2/17/25	Sun 2/23/25
5	1.3.1 Difficulty in accessing reliable and expert-backed pet health info	2 days	Mon 2/17/25	Tue 2/18/25
6	1.3.2 Inefficient pet health record management	2 days	Wed 2/19/25	Thu 2/20/25
7	1.3.3 Difficulty in assessing when to seek veterinary care	2 days	Fri 2/21/25	Sun 2/23/25
8	1.4 Define aim and objectives	6 days	Mon 2/17/25	Sun 2/23/25
9	1.5 Propose solution (AI chatbot, symptom checker, health records)	6 days	Mon 2/24/25	Sun 3/2/25
10	1.6 Determine project approach	6 days	Mon 2/24/25	Sun 3/2/25
11	1.6.1 Agile methodology	3 days	Mon 2/24/25	Wed 2/26/25
12	1.6.2 Iterative sprint planning	3 days	Thu 2/27/25	Sun 3/2/25
13	1.7 Define project scope	6 days	Mon 3/3/25	Sun 3/9/25
14	1.7.1 Target users: Pet owners, caregivers	3 days	Mon 3/3/25	Wed 3/5/25
15	1.7.2 Main modules: Chatbot, Symptom Checker, Health Records, Educatior	3 days	Thu 3/6/25	Sun 3/9/25
16	1.8 Literature Review	11 days	Mon 3/10/25	Sun 3/23/25
17	1.8.1 AI in pet healthcare	5 days	Mon 3/10/25	Fri 3/14/25
18	1.8.1.1 Summarize papers on chatbots, ML, and vet AI	5 days	Mon 3/10/25	Fri 3/14/25
19	1.8.2 Review existing pet care apps (TTcare, PetVet AI, PetVitality, 11Pets)	3 days	Sat 3/15/25	Tue 3/18/25
20	1.8.2.1 Feature analysis	1 day	Sat 3/15/25	Sat 3/15/25
21	1.8.2.2 Comparative study	2 days	Sun 3/16/25	Mon 3/17/25
22	1.8.3 Software development methodologies	2 days	Tue 3/18/25	Wed 3/19/25
23	1.8.3.1 Agile vs Waterfall vs RAD	2 days	Thu 3/20/25	Fri 3/21/25
24	1.8.4 Frontend & Backend tools comparison	2 days	Thu 3/20/25	Fri 3/21/25
25	1.8.4.1 React Native vs Flutter	1 day	Thu 3/20/25	Thu 3/20/25
26	1.8.4.2 Supabase vs Firebase	1 day	Fri 3/21/25	Fri 3/21/25
27	1.8.5 AI APIs	2 days	Sat 3/22/25	Sun 3/23/25
28	1.8.5.1 OpenAI, DeepSeek, OpenRouter comparison	2 days	Sat 3/22/25	Sun 3/23/25
29	1.9 Define methodology and work plan	6 days	Mon 3/24/25	Sun 3/30/25
30	1.9.1 Choose Agile methodology	2 days	Mon 3/24/25	Tue 3/25/25
31	1.9.2 Define sprints and iterations	2 days	Wed 3/26/25	Thu 3/27/25
32	1.9.3 Create Gantt chart	2 days	Fri 3/28/25	Sun 3/30/25
33	1.10 Conduct survey and fact-finding	6 days	Mon 3/31/25	Sun 4/6/25
34	1.10.1 Design survey questionnaire	2 days	Mon 3/31/25	Tue 4/1/25
35	1.10.2 Distribute via Google Forms	2 days	Wed 4/2/25	Thu 4/3/25
36	1.10.3 Analyse results for feature refinement	2 days	Fri 4/4/25	Sun 4/6/25
37	1.11 Identify development tools	6 days	Mon 4/7/25	Sun 4/13/25
38	1.11.1 Visual Studio Code, GitHub	2 days	Mon 4/7/25	Tue 4/8/25
39	1.11.2 Android Studio Emulator	2 days	Wed 4/9/25	Thu 4/10/25
40	1.11.3 Supabase, OpenRouter AI	2 days	Fri 4/11/25	Mon 4/14/25
41	1.12 Define requirements	6 days	Mon 4/14/25	Sun 4/20/25
42	1.12.1 Functional requirements	3 days	Mon 4/14/25	Wed 4/16/25
43	1.12.2 Non-functional requirements	3 days	Thu 4/17/25	Mon 4/21/25

Figure 3.12: Project Planning and Requirements Gathering Gantt Chart

3.6.2.2 System Design Phase



ID	Task Name	Duration	Start	Finish
45	**2.0 System Design Phase**	26 days	Mon 4/21/25	Mon 5/26/25
46	2.1 Use Case Modeling	6 days	Mon 4/21/25	Sun 4/27/25
47	2.1.1 Illustrate use case diagram	3 days	Mon 4/21/25	Wed 4/23/25
48	2.1.2 Define use cases for each feature	3 days	Thu 4/24/25	Mon 4/28/25
49	2.2 UI/UX Design	3 days	Mon 4/28/25	Wed 4/30/25
50	2.2.1 Wireframe with Figma	2 days	Mon 4/28/25	Tue 4/29/25
51	2.2.2 Define screen navigation and routing	1 day	Wed 4/30/25	Wed 4/30/25
52	2.3 System Architecture Design	6 days	Mon 5/5/25	Sun 5/11/25
53	2.3.1 Define database schema (pet records, users)	3 days	Mon 5/5/25	Wed 5/7/25
54	2.3.2 Chatbot and API communication flow	3 days	Thu 5/8/25	Mon 5/12/25
55	2.4 API Integration Design	6 days	Mon 5/12/25	Sun 5/18/25
56	2.4.1 OpenRouter for chatbot	3 days	Mon 5/12/25	Wed 5/14/25
57	2.4.2 Symptom checker (prompt-based)	3 days	Thu 5/15/25	Mon 5/19/25
58	2.5 Feedback and Review	6 days	Mon 5/19/25	Sun 5/25/25
59	2.5.1 Discuss designs with supervisor	3 days	Mon 5/19/25	Wed 5/21/25
60	2.5.2 Refine UI based on feedback	3 days	Thu 5/22/25	Mon 5/26/25

Figure 3.13: System Design Phase Gantt Chart

3.6.2.3 Development Phase

ID	Task Name	Duration	Start	Finish
62	**3.0 Development Phase**	61 days	Mon 5/26/25	Sun 8/17/25
63	**3.1 Sprint 1 – UI/UX Development & Navigation Setup**	16 days	Mon 5/26/25	Sun 6/15/25
64	3.1.1 Design and develop core UI screens	6 days	Mon 5/26/25	Sun 6/1/25
65	3.1.1.1 Login and Register pages	2 days	Mon 5/26/25	Tue 5/27/25
66	3.1.1.2 Home and Navigation Tabs	2 days	Wed 5/28/25	Thu 5/29/25
67	3.1.1.3 Pet Dashboard and Profile pages	2 days	Fri 5/30/25	Sat 5/31/25
68	3.1.2 Implement navigation flow between screens	2 days	Sun 6/1/25	Mon 6/2/25
69	3.1.3 Create low-fidelity wireframes in Figma for reference	2 days	Tue 6/3/25	Wed 6/4/25
70	3.1.4 Conduct UI testing and adjust screen layout	2 days	Thu 6/5/25	Fri 6/6/25
71	3.1.5 Present to supervisor for design feedback	1 day	Sat 6/7/25	Sat 6/7/25
72	3.1.6 Refine layout based on feedback	2 days	Sun 6/8/25	Mon 6/9/25
73	3.1.7 Ensure responsive design across device sizes	3 days	Tue 6/10/25	Thu 6/12/25
74	3.1.8 Refine UI animations, padding, and transitions	2 days	Fri 6/13/25	Sun 6/15/25
75				
76	**3.2 Sprint 2 – Authentication, Database & Pet Profile Features**	16 days	Mon 6/16/25	Sun 7/6/25
77	3.2.1 Set up Supabase authentication (Register/Login flow)	2 days	Mon 6/16/25	Tue 6/17/25
78	3.2.2 Integrate Supabase user sessions	2 days	Wed 6/18/25	Thu 6/19/25
79	3.2.3 Set up Supabase database schema	2 days	Fri 6/20/25	Sun 6/22/25
80	3.2.3.1 User profile data	1 day	Fri 6/20/25	Fri 6/20/25
81	3.2.3.2 Pet profile structure	2 days	Sat 6/21/25	Sun 6/22/25
82	3.2.4 Develop CRUD operations for pet profiles	2 days	Mon 6/23/25	Tue 6/24/25
83	3.2.5 Test user flow	2 days	Wed 6/25/25	Thu 6/26/25
84	3.2.6 Fix authentication & session handling bugs	2 days	Fri 6/27/25	Sat 6/28/25
85	3.2.7 Develop UI for health record management	2 days	Sun 6/29/25	Mon 6/30/25
86	3.2.8 Integrate Supabase to store & retrieve records	2 days	Tue 7/1/25	Wed 7/2/25
87	3.2.9 Add features for vaccination, notes, vet visits	2 days	Thu 7/3/25	Fri 7/4/25
88	3.2.10 Build multi-pet profile viewing	1 day	Sat 7/5/25	Sat 7/5/25
89	3.2.11 Validate data integrity between pet and user	1 day	Sun 7/6/25	Sun 7/6/25
90				
91	**3.3 Sprint 3 – AI Integration & Symptom Checker**	16 days	Mon 7/7/25	Sun 7/27/25
92	3.3.1 Connect OpenRouter API to chatbot interface	2 days	Mon 7/7/25	Tue 7/8/25
93	3.3.2 Design chatbot prompt flow	2 days	Wed 7/9/25	Thu 7/10/25
94	3.3.3 Handle loading, error, and fallback states	2 days	Fri 7/11/25	Sat 7/12/25
95	3.3.4 Style chat interface to match app theme	2 days	Sun 7/13/25	Mon 7/14/25
96	3.3.5 Test responses to general pet care queries	2 days	Tue 7/15/25	Wed 7/16/25
97	3.3.6 Refactor prompt logic for better accuracy	2 days	Thu 7/17/25	Fri 7/18/25
98	3.3.7 Design and implement Symptom Checker form	2 days	Sat 7/19/25	Sun 7/20/25
99	3.3.8 Connect prompt to OpenRouter for symptom advice	2 days	Mon 7/21/25	Tue 7/22/25
100	3.3.9 Include dropdowns, input fields for symptom data	2 days	Wed 7/23/25	Thu 7/24/25
101	3.3.10 Add educational content module (e.g., FAQs, articles)	2 days	Fri 7/25/25	Sun 7/27/25
102				
103	**3.4 Sprint 4 – Final Features, Feedback & Testing**	16 days	Mon 7/28/25	Sun 8/17/25
104	3.4.1 Implement offline support with local storage (if feasible)	3 days	Mon 7/28/25	Wed 7/30/25
105	3.4.2 Add feedback module (ratings/comments)	2 days	Thu 7/31/25	Fri 8/1/25
106	3.4.3 Conduct a final UX test with pet owners for feedback	2 days	Sat 8/2/25	Sun 8/3/25
107	3.4.4 Conduct integration testing across all modules	3 days	Mon 8/4/25	Wed 8/6/25
108	3.4.5 Bug fixing across navigation, records, and API calls	3 days	Thu 8/7/25	Sat 8/9/25
109	3.4.6 Optimize database queries and loading speed	2 days	Sun 8/10/25	Mon 8/11/25
110	3.4.7 Prepare app for demo deployment	3 days	Tue 8/12/25	Thu 8/14/25
111	3.4.8 Submit final working prototype for evaluation	2 days	Fri 8/15/25	Sun 8/17/25

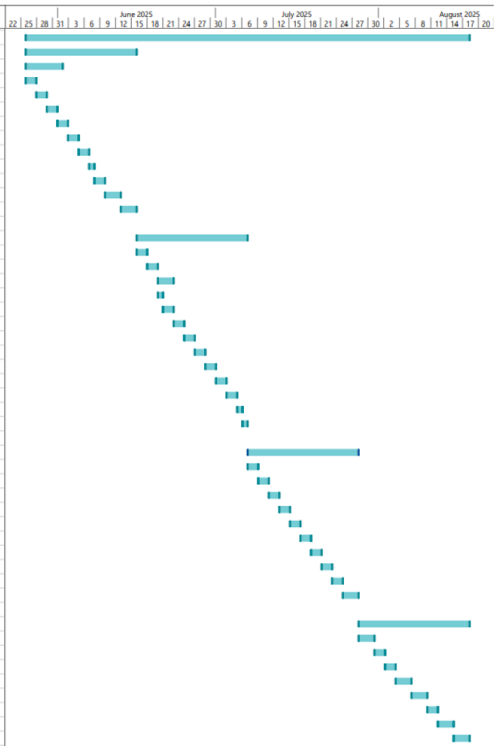


Figure 3.14: Development Phase Gantt Chart

3.6.2.4 Testing Phase

ID	Task Name	Duration	Start	Finish
113	**4.0 Testing Phase**	21 days	Mon 8/18/25	Sun 9/14/25
114	4.1 Unit Testing	5 days	Mon 8/18/25	Fri 8/22/25
115	4.1.1 Test UI components and logic	3 days	Mon 8/18/25	Wed 8/20/25
116	4.1.2 Test API responses	2 days	Sat 8/2/25	Mon 8/4/25
117	4.2 Integration Testing	4 days	Sat 8/23/25	Wed 8/27/25
118	4.2.1 Ensure smooth frontend-backend integration	4 days	Sat 8/23/25	Wed 8/27/25
119	4.3 Usability Testing	3 days	Thu 8/28/25	Mon 9/1/25
120	4.3.1 Gather feedback from users (pet owners)	2 days	Thu 8/28/25	Fri 8/29/25
121	4.3.2 Update UX/UI accordingly	1 day	Sun 8/31/25	Sun 8/31/25
122	4.4 Performance Testing	3 days	Tue 9/2/25	Thu 9/4/25
123	4.4.1 Assess load speed, responsiveness	3 days	Tue 9/2/25	Thu 9/4/25
124	4.5 Security Testing	2 days	Fri 9/5/25	Sun 9/7/25
125	4.5.1 Validate login, data protection	2 days	Fri 9/5/25	Sun 9/7/25
126	4.6 Bug fixing	6 days	Mon 9/8/25	Sun 9/14/25

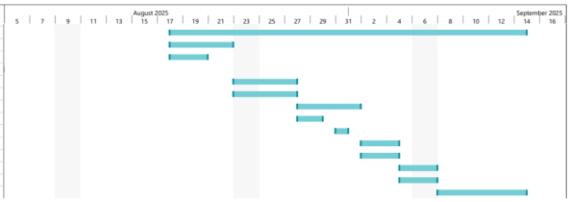


Figure 3.15: Testing Phase Gantt Chart

3.6.2.5 Closing Phase

ID	Task Name	Duration	Start	Finish
128	**5.0 Project Closing**	6 days	Mon 9/15/25	Sun 9/21/25
129	5.1 Final code optimization and polish	2 days	Mon 9/15/25	Tue 9/16/25
130	5.2 Prepare final documentation	2 days	Wed 9/17/25	Thu 9/18/25
131	5.3 Final report compilation	1 day	Fri 9/19/25	Fri 9/19/25
132	5.4 Poster and demo video preparation	1 day	Sat 9/20/25	Sat 9/20/25
133	5.5 Submit deliverables	1 day	Sun 9/21/25	Sun 9/21/25
134	5.6 Final presentation and evaluation	1 day	Sun 9/21/25	Sun 9/21/25

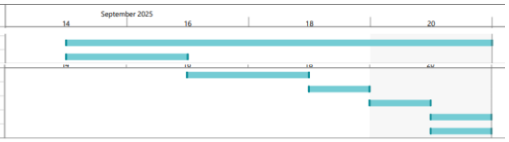


Figure 3.16: Closing Phase Gantt Chart

CHAPTER 4

PROJECT INITIAL SPECIFICATIONS

4.1 Introduction

In this section, data collected through questionnaires are analyse and preliminary research is presented to identify user needs and system expectations for the PawHub application. Based on these findings, the requirements are written, the use case diagram are drawn, and based on these specifications, the use case description, and user interface designs are developed and included in this chapter. Overall, this chapter serves as a blueprint for the application's core features and user experience design.

4.2 Facts Finding

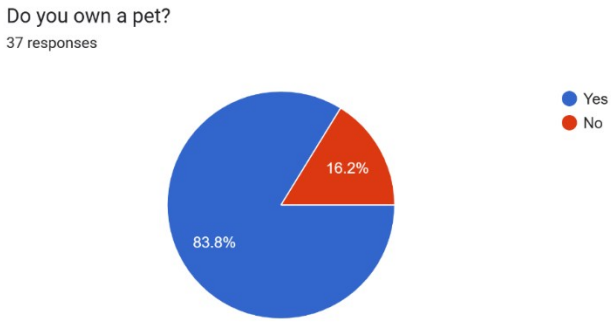


Figure 4.1: Target Users of Survey

This section in my report presents an in-depth analysis of pet owner's current practices, preferences, and challenges in managing pet health, gathered through a comprehensive survey. The survey sample, composed entirely of active pet owners, with 83.8% (37 respondents) confirming pet ownership, validates the relevance of our study and underscores a strong inclination towards digital solutions for pet health management. These findings reveal significant gaps in current offerings and highlight a growing demand for an integrated, user-friendly platform that meets the real needs of pet owners, thereby guiding the design and development of PawHub to enhance timely and effective care for their pets.

4.2.1 Section 1: General Information

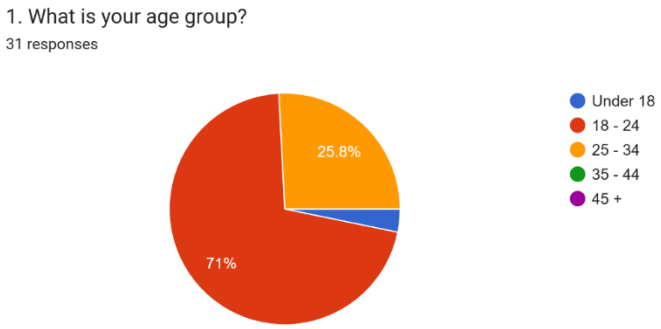


Figure 4.2: Survey Question 1, Age Group

The survey data indicate that the majority of respondents are young adults, with 71% aged between 18 and 24, and an additional 25.8% falling within the 25–34 age bracket. This concentration in the younger demographic suggests that the target user base is likely to be tech-savvy, comfortable with mobile apps, and open to adopting innovative digital solutions for managing pet care. Such a youthful audience is often receptive to modern, interactive interfaces and may value features that integrate seamlessly with their digital lifestyles.

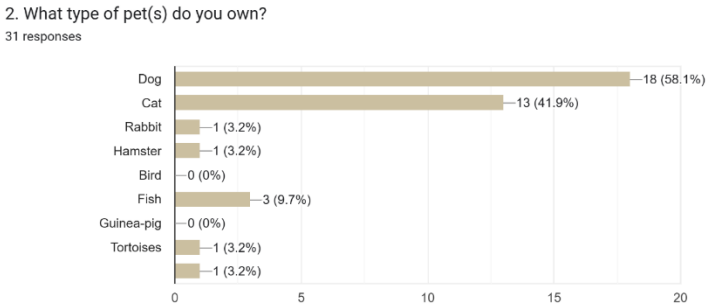


Figure 4.3: Survey Question 2, Pet Types

In terms of pet types, the survey shows that dogs are the most popular, owned by 58.1% of respondents, while 41.9% own cats. There is also a smaller percentage of respondents who own fish, birds, and other pets. This distribution implies that while the app should primarily cater to dog and cat owners by offering features and content tailored to these animals, it should also remain flexible enough to accommodate the needs of owners with less common pet types, ensuring inclusivity and broad appeal.

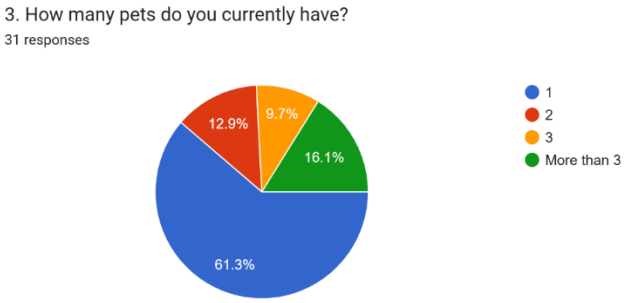


Figure 4.4: Survey Question 3, Number of pets

The survey reveals that the majority of respondents (61.3%) own just one pet, which suggests that many users may prefer a simple and straightforward user interface for managing a single pet's profile. However, with 12.9% owning two pets, 9.7% owning three, and 16.1% owning more than three pets, there is a clear indication that the app should support multi-pet management. This feature would enable users with multiple pets to easily switch between profiles and maintain comprehensive health records for each animal, thereby increasing the app's overall usability and appeal.

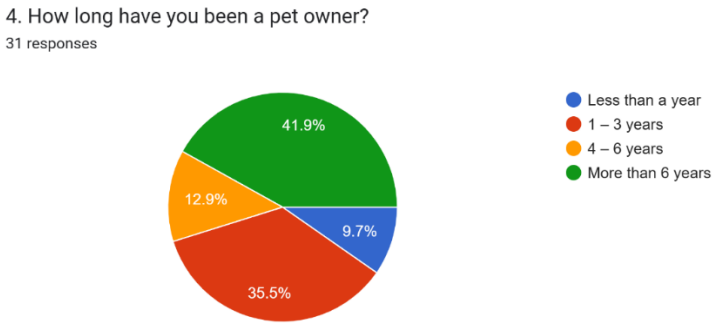


Figure 4.5: Survey Question 4, Pet Ownership Experience

Respondents experience levels with pet ownership vary significantly. The largest group, comprising 41.9% of respondents, has been pet owners for more than six years, suggesting they are likely to be well-informed and may demand more advanced features or nuanced insights regarding pet care. Meanwhile, 35.5% have 1–3 years of experience, indicating a need for educational content and guidance to help newer pet owners make informed decisions. With 12.9% having 4–6 years and 9.7% less than a year of experience, the app should ideally offer a balance of advanced functionalities

for experienced owners and user-friendly, educational features to assist those who are relatively new to pet care.

4.2.2 Section 2: Current Pet Care Practices

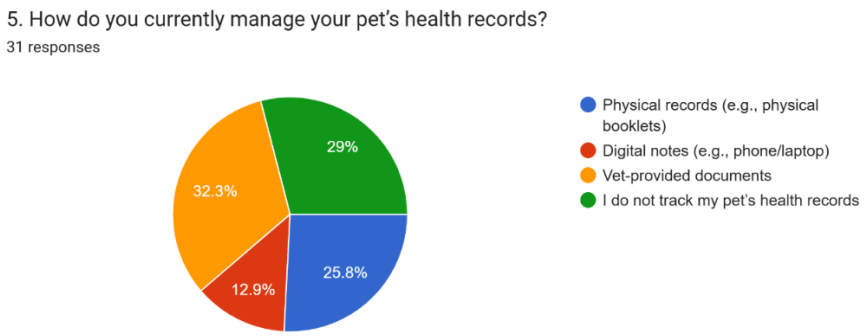


Figure 4.6: Survey Question 5, Current Pet Health Management

Out of 31 respondents, 25.8% rely on physical records like booklets, while 12.9% use digital notes on devices such as phones or laptops. A significant 32.3% depend on documents provided by veterinarians, and 29% indicated that they do not track their pet's health records at all. These results suggest a varied approach to record-keeping, reflecting both traditional and modern practices. The fact that nearly one-third of respondents do not track their records at all underscores a clear opportunity for a centralized digital solution like PawHub to simplify health record management and ensure that critical information is readily accessible.

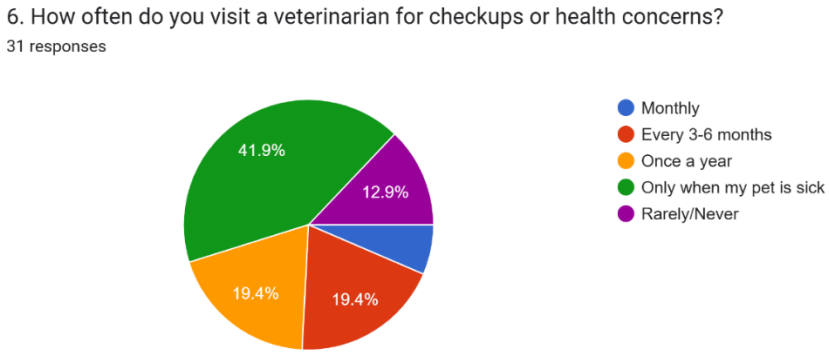


Figure 4.7: Survey Question 6, Vet Visits for checkup or health concerns

Among the 31 respondents, 6.5% visit a veterinarian monthly, 19.4% do so every 3-6 months, and another 19.4% attend annual checkups. However, the largest group, 41.9%, only visits a vet when their pet is sick, and 12.9% reported that they rarely or never make vet visits. This distribution indicates that while routine checkups are followed by some, a substantial portion of pet owners delay veterinary visits until a problem arises. Such insights highlight the potential benefit of an app that encourages proactive health management through regular reminders and alerts, potentially reducing the reliance on reactive, emergency-based vet visits.

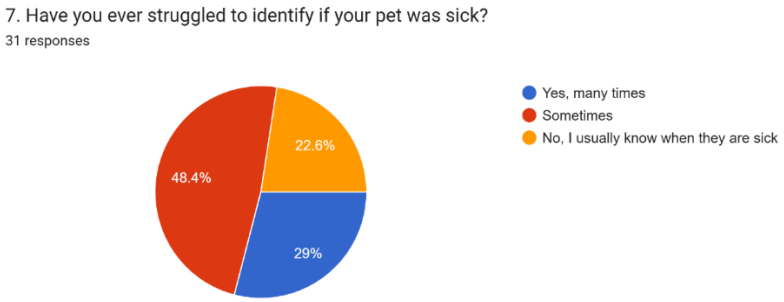


Figure 4.8: Survey Question 7, Struggles to identify if pet is sick

When asked about the difficulty of identifying sickness in their pets, 29% of respondents stated that they have experienced this issue many times, 48.4% said it happens sometimes, and 22.6% mentioned that they usually know when their pet is sick. This data reveals that a significant majority, over three-quarters of respondents face challenges in accurately identifying health issues in their pets. This finding underscores the need for an intelligent, AI-powered symptom diagnosis tool within PawHub that could assist pet owners in early detection and prompt action, ultimately leading to better health outcomes for their pets.

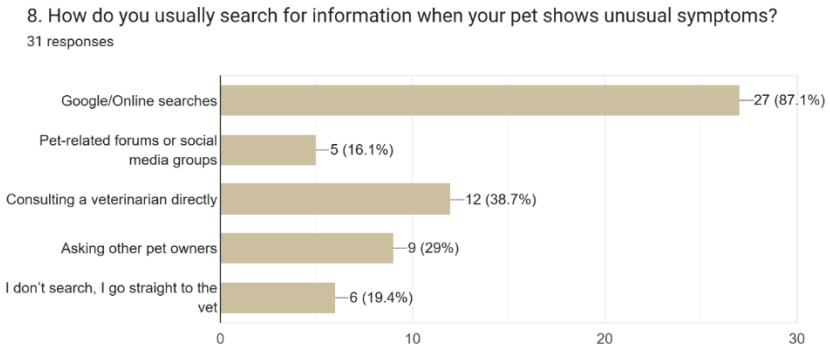


Figure 4.9: Survey Question 8, Ways to search unusual symptoms

The responses indicate that a large majority of 87.1% rely on Google or other online searches to gather information when their pet exhibits unusual symptoms. Additionally, 38.7% consult a veterinarian directly, 29% ask other pet owners, and only 16.1% utilize pet-related forums or social media groups. These findings suggest that while pet owners frequently turn to online sources, which may vary in reliability, a significant number still value professional guidance and peer advice. This reinforces the need for a trusted, integrated solution like PawHub that provides accurate AI-driven insights, thereby reducing the reliance on less consistent online information and unnecessary google searches.

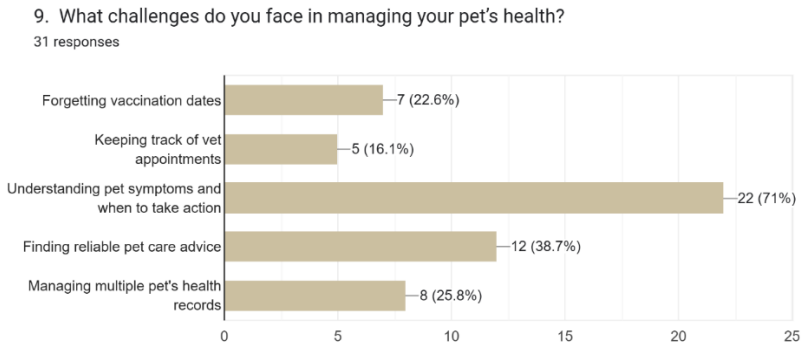


Figure 4.10: Survey Question 9, Challenges faced in managing pet's health

This question revealed several common struggles faced by pet owners, with the most prominent issue being understanding pet symptoms and knowing when to take action, cited by 71% of respondents. This suggests that many owners feel uncertain or lack the confidence to assess their pet's health,

which could lead to delayed or inappropriate responses to illnesses. Additionally, 38.7% of respondents stated they struggle with finding reliable pet care advice, highlighting the difficulty of navigating the vast amount of sometimes conflicting information available online. A notable 25.8% mentioned the challenge of managing multiple pets' health records, emphasizing a need for a system that can consolidate and organize health data in one place. Other common issues include forgetting vaccination dates (22.6%) and keeping track of vet appointments (16.1%). These findings clearly indicate that pet owners face both informational and organizational difficulties, underlining the necessity for an app like PawHub, which can deliver trustworthy insights, reminders, and comprehensive record management features tailored to individual pet needs.

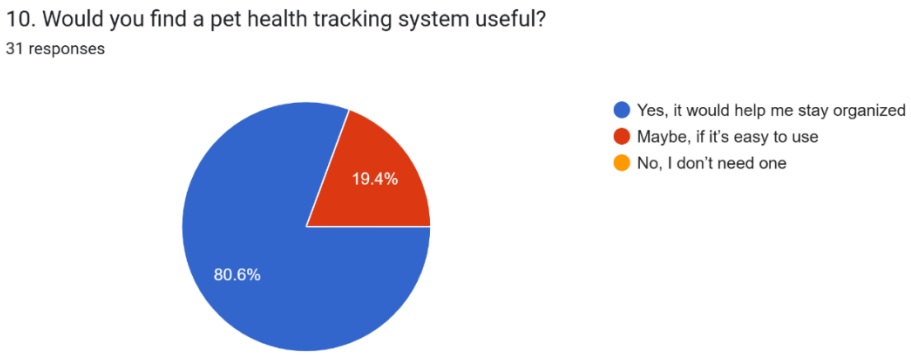


Figure 4.11: Survey Question 10, Pet Health Tracking System Usefulness

Most participants (80.6%) expressed that a pet health tracking system would help them stay organized, showing strong demand for such a digital tool. A further 19.4% said they might find it useful if it is easy to use, indicating that usability and intuitive design are crucial for adoption. Significantly, no respondents rejected the idea of a tracking system, reinforcing the notion that most pet owners are open to technological solutions, especially if they can simplify and improve how pet health is managed. This feedback strongly supports the integration of tracking features in PawHub, such as, vet visit logs, symptom monitoring, and multi-pet support, all within a user-friendly interface.

4.2.3 Section 3: AI Chatbot & App Features

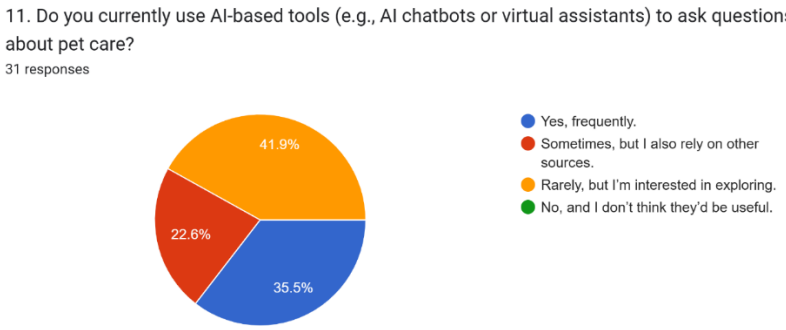


Figure 4.12: Survey Question 11, AI-based Virtual Assistant Usage

This question aimed to gauge the familiarity and openness of respondents toward AI tools in pet care. About 35.5% stated they frequently use AI-based tools, indicating a growing acceptance of modern technology in everyday pet care routines. Another 22.6% use them occasionally alongside other resources, showing that AI is often used in combination with traditional sources such as vets or online forums. Interestingly, 41.9% said they rarely use AI but are open to trying it, suggesting that while AI hasn't become mainstream in this context yet, there is strong interest and potential for adoption. Importantly, no one rejected the usefulness of AI outright, which highlights a positive user sentiment towards integrating such tools into a pet care app.

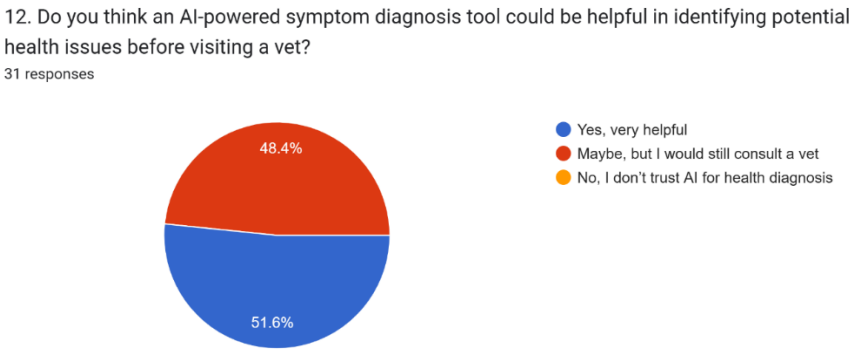


Figure 4.13: Survey Question 12, AI-based Symptoms Diagnosis Tool Usage

The response was overwhelmingly positive, with 51.6% saying it would be very helpful, and the remaining 48.4% agreeing it might be useful,

though they would still consult a vet. Notably, no one expressed distrust in using AI for preliminary diagnosis, suggesting that users are open to AI tools as a first line of assistance or a support system before seeking professional care. This confirms that there is a strong interest in using AI to bridge the gap between noticing symptoms and taking action, giving reassurance to pet owners who may otherwise delay or struggle with decision-making.

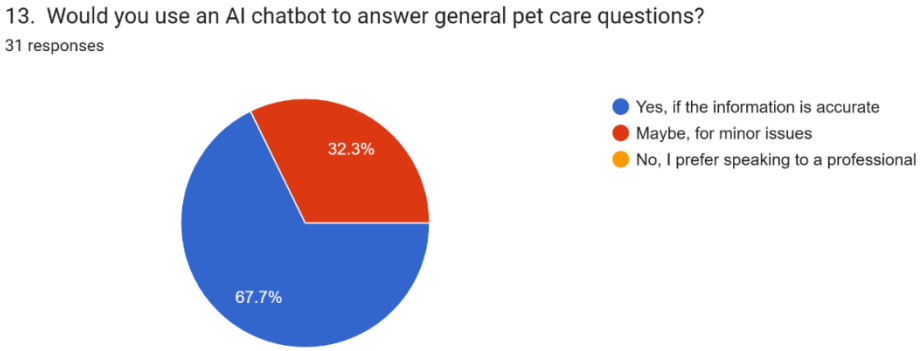


Figure 4.14: Survey Question 13, AI-based Virtual Assistant Usage for General Pet Care Questions

A large majority (67.7%) responded yes, provided that the chatbot delivers accurate information, while 32.3% indicated they would use it for minor issues. Again, none rejected the idea of using an AI chatbot, showing widespread acceptance of such tools, especially for low-risk or general informational pet care queries. This points to the need for PawHub to prioritize data accuracy and reliability, as trust in the information source is key for continued usage and user satisfaction.

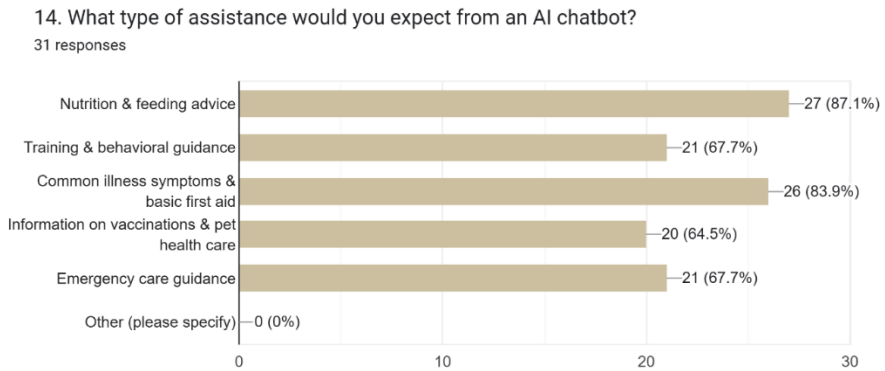


Figure 4.15: Survey Question 14, AI-based Virtual Assistant Usage

This multi-choice question highlighted the specific functionalities users desire from an AI chatbot. Nutrition and feeding advice (87.1%), common illness symptoms and basic first aid (83.9%), and emergency care guidance (67.7%) were among the most requested features. Additionally, training and behavioural guidance (67.7%) and information on vaccinations and general pet health care (64.5%) also ranked high. These results reflect a wide range of needs, from routine care to critical health concerns, emphasizing the importance of developing a comprehensive and multi-functional AI assistant in PawHub. Users expect more than just casual chat, they want expert-level educational resources across diverse pet health topics.

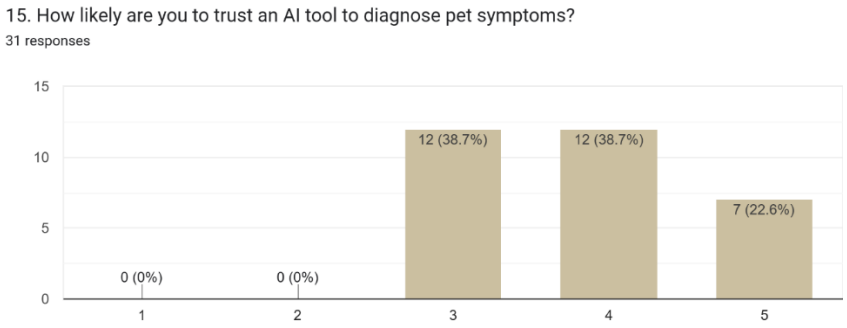


Figure 4.16: Survey Question 15, User Trust on AI Tool For Symptoms Diagnosis

Trust levels are relatively high, with most responses clustering around the mid-to-high end of the scale. 38.7% selected “3”, indicating a neutral but open stance, the same percentage selected “4”, showing strong confidence in AI tools. Moreover, 22.6% rated it “5”, showing very high trust in AI-powered symptom diagnosis. Importantly, no respondents chose “1” or “2”, indicating that nobody is fundamentally opposed to trusting AI in this role. This shows well for PawHub, suggesting that the more accurate and transparent the tool becomes, the more user confidence it can build over time.

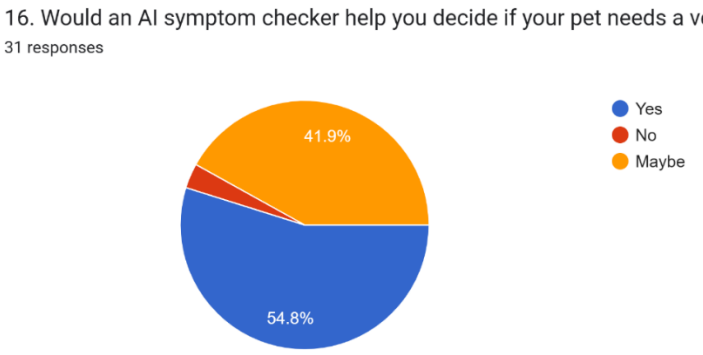


Figure 4.17: Survey Question 16, AI Symptom Checker

More than half (54.8%) agreed that such a tool would help them decide, while 41.9% said maybe, again pointing to openness dependent on accuracy and reliability. Only 3.2% rejected the usefulness of an AI symptom checker outright. This highlights the opportunity for PawHub to serve as a decision-support tool, helping users interpret symptoms and make timely decisions on seeking veterinary care, especially useful for owners unsure whether a symptom is urgent.

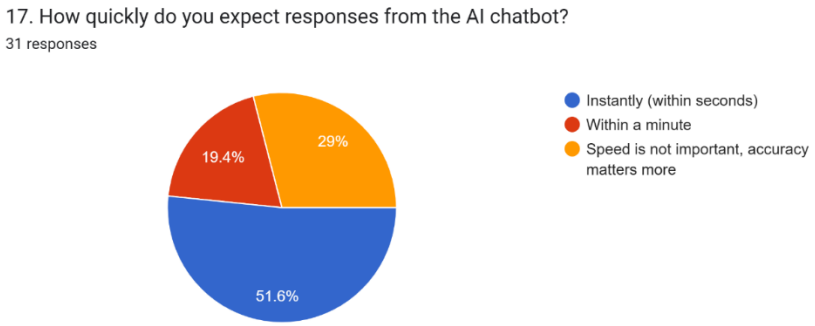


Figure 4.18: Survey Question 17, AI-based Virtual Assistant Response Rate

Users had mixed but insightful expectations about response time. 51.6% preferred instant responses, reflecting the need for real-time interaction in cases of stress or urgency. 19.4% were comfortable with a delay of up to a minute, likely valuing slightly delayed but thoughtful responses. Interestingly, another 51.6% emphasized that “accuracy matters more than speed”, which reinforces the importance of providing correct and helpful information over quick, generic answers. These insights suggest PawHub should aim for a

balance, prioritize speed, when possible, but never at the expense of accuracy and user trust.

4.2.4 Section 4: Pet Health Record Management Preferences

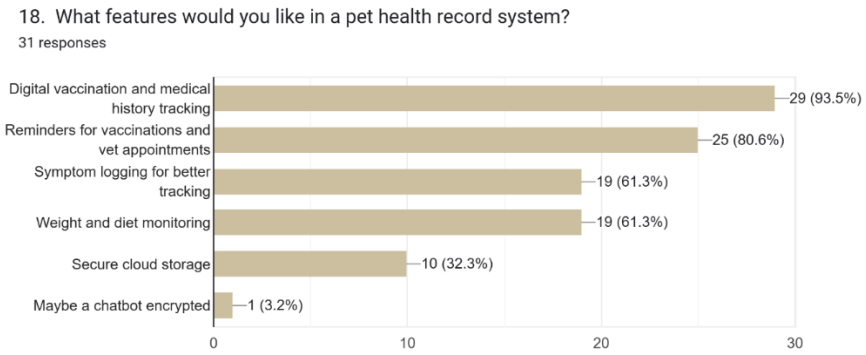


Figure 4.19: Survey Question 18, Preferred features in a Pet Health Record System

When asked about preferred features for a digital pet health record system, the most in-demand feature was digital vaccination and medical history tracking, with 93.5% of respondents selecting it. Reminders for vaccinations and vet appointments followed closely at 80.6%, indicating that users prioritize staying up to date on their pet's health needs. Symptom logging and weight/diet monitoring were each selected by 61.3% of respondents, showing that many pet owners want more detailed tracking tools. Additionally, 32.3% valued secure cloud storage for accessing data across devices, and a small group (3.2%) suggested integrating an encrypted chatbot for privacy and convenience.

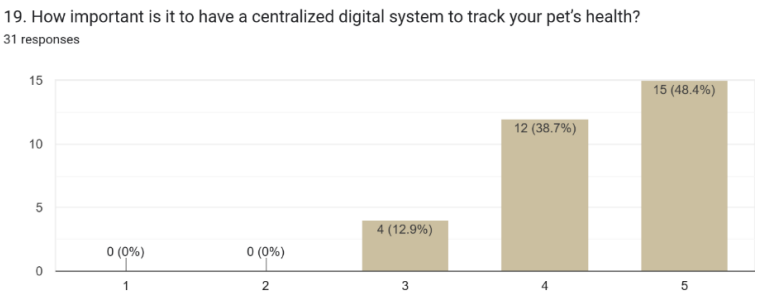


Figure 4.20: Survey Question 19, Importance of Centralized Digital System

The majority of respondents considered a centralized digital system highly important for managing their pet's health. Specifically, 48.4% rated its importance as 5 (very important), while 38.7% gave it a 4, making up a combined 87.1% who strongly support the idea. The remaining 12.9% rated it a 3, indicating moderate interest, while none of the participants rated it as unimportant (1 or 2). This shows a strong preference for a unified and digital health tracking solution among pet owners.

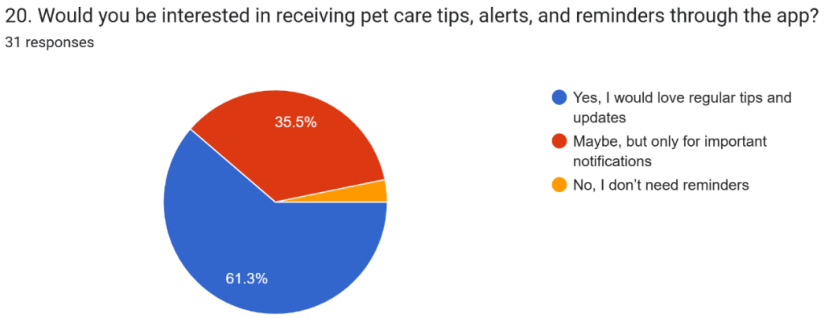


Figure 4.21: Survey Question 20, Interest in Receiving Pet Care Tips, Alerts, and Reminders

Regarding engagement through alerts and updates, 61.3% of participants stated they would love to receive regular pet care tips and notifications through the app. Another 35.5% expressed interest in receiving updates, but only for important matters such as vaccinations or health concerns. Just 3.2% mentioned that they do not require reminders. Overall, this indicates a clear interest in having timely, relevant notifications delivered through the app to help with responsible pet care management.

4.2.5 Section 5: App Usability and Feature Preferences

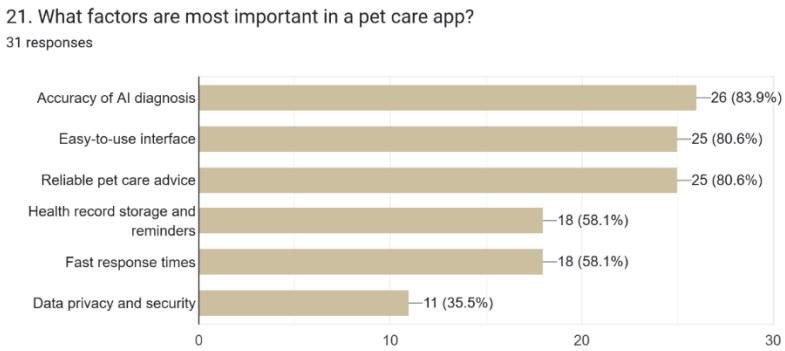


Figure 4.22: Survey Question 21, Most Important Factors in Pet Care App

When identifying key priorities for a pet care app, 83.9% of respondents ranked the accuracy of AI diagnosis as the most important factor. This was followed closely by an easy-to-use interface and reliable pet care advice, both of which were selected by 80.6% of participants, highlighting the need for simplicity and trustworthy information. Additionally, 58.1% valued health record storage and reminders, and the same percentage emphasized the importance of fast response times. Data privacy and security were also a concern for 35.5% of users, suggesting that while functionality is important, protecting user data should not be overlooked.

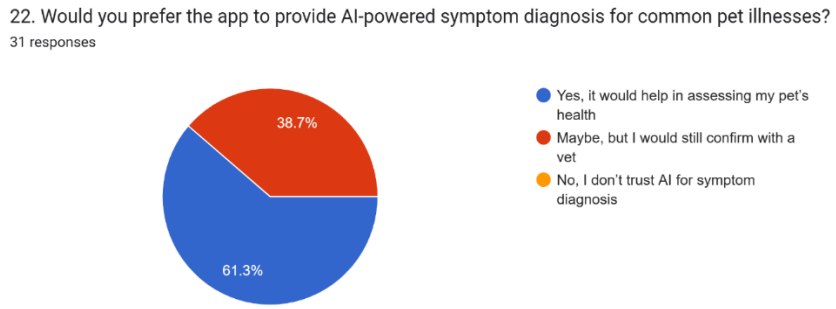


Figure 4.23: Survey Question 22, Preference for an AI-powered Symptom Diagnosis Tool

A majority of respondents (61.3%) expressed that they would prefer the app to include AI-powered symptom diagnosis to help assess their pet's health. Another 38.7% responded "maybe," showing openness to the idea while emphasizing that they would still confirm with a veterinarian.

Importantly, none of the respondents rejected the idea outright, which reflects a high level of trust or curiosity in the role of AI in supporting pet health assessments.

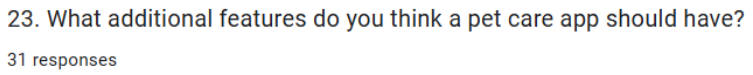


Figure 4.24: Survey Question 23, Additional Features Suggestion

Table 4.1: Additional Features User Responses (Open ended question)

Pet locations if they go missing	Reminders
photo record feature for pet so that it is easier to have a lost and found if the pet accidentally lost	Nearest animal clinics? And maybe it's ratings? Could do like a collaboration type thing
Keeping individual record and profile for each pet that user own. Categorising them can be a help to those who owns more that one type of pet.	Perhaps a call/communication center to connect veterinarians/trained professionals with consumers virtually
Maybe we could have specialists online for chatting with the users regarding their pets' behaviours.	Maybe a social hub for pet parents to share tips, photos of their pet and personal stories.
Nutritional plans, behavioural training tips etc	Pet communities for those which shares location
A profile photo for your pet !	Location for the nearest veterinarian
Real time update and notification	Facts
Chatbot encrypted. Maybe, real-time advice from a vet and also identifying the nearest animal hospital in case of any emergency.	General knowledge/educational stuff (importance of neutering/vaccination/safety tips or precautions to take after vaccination)
where I can take picture and upload to ask om diagnosis.	Honestly most things have been covered!
multiple language can be accessed	Locations for near by clinic
Lost pet alert system	Reminders

Participants suggested a wide range of features to improve the overall functionality and usefulness of the pet care app. Several respondents recommended features to help in emergencies, such as lost pet alert systems, pet location tracking, and real-time notifications. Others proposed having individual pet profiles, photo records, and categorization for multiple pets. There were also suggestions for real-time vet consultations, access to nearby clinics with ratings, and AI-based image diagnosis tools. Some participants wanted multilingual access, social features like a pet parent community, educational content, and behavioural training tips. These ideas reflect a desire for both practical tools and community engagement features that support day-to-day pet care. Many of the features suggested are already in line with your planned functionality for PawHub, like AI advice, pet profiles, and vet communication. Others such as social features, lost pet systems, clinic locators, and education tools can be considered for future iterations to enhance user value and experience.

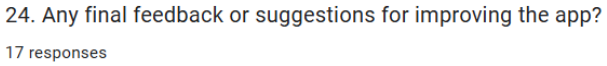


Figure 4.25: Survey Question 24, Final Feedback or Suggestion for improving the application

Table 4.2: Final Feedback or Suggestion user responses (Open ended question)

If there was a way to gauge people's trust in AI, like maybe a health department certification or ethics certification that ensures users they can rely on the app for accurate information	Keep it simple and user friendly which can also be used by older generations which aren't really good with technology.
Customization options, In-App chat with Vet, Offline mode	Can apply real veterinarian in the apps to increase the accountability
Allow for other pet owners to add each other on the app	Might be useful to profile for each pet, as some people have multiple

	pets to take care of.
More educational/fun facts	Have multiple pets account accessible
Overall, very good !	Nice app idea ! All the best :)

Final feedback from users was mostly positive and constructive. Many emphasized the importance of having multiple pet account access and individual profiles, especially for users with more than one pet. Simplicity and user-friendliness were recurring themes, especially to ensure that the app is accessible to older generations. Several participants suggested offline access, customization options, and real veterinarian integration to enhance trust and credibility. There were also ideas like in-app vet chats, friend-adding features among pet owners, and even certification labels to validate the app's reliability. Overall, the feedback was encouraging, with multiple users expressing support and enthusiasm for the app concept. These suggestions emphasize the importance of user-centric design, credibility, and reliability. Integrating feedback from real users into future development such as vet verification, multiple pet profiles, and simplicity can set PawHub apart in a growing market of pet care apps.

4.3 Requirements Specification

This section defines the functionalities of the PawHub mobile application. The requirements were derived from the key features of the application and aligned with the objectives of promoting accessible, AI-powered pet care management.

4.3.1 Functional Requirements

Table 4.3: Functional Requirements

ID	Functional Requirements
F001	Pet owners must be able to create a new account to access the app.
F002	Pet owners must be able to log into their existing account to access the app.
F003	Pet owners should have the ability to view and update their personal profile information.
F004	The system must support pet owners to manage their pet profile.
F005	Pet owners should have the ability to add pets, view their pet profile, update their information and delete multiple pet profiles.
F006	The system must allow pet owners to add, view and delete health records for each pet.
F007	The system should automatically notify pet owners with reminders if the health record date is in the future.
F008	The system should provide an AI-powered chatbot that answers pet care inquiries (nutrition, training, behaviour, etc.) in real time.
F009	Pet owners must be able to toggle between three AI models in the chatbot for varied response styles.
F010	The system should allow pet owners to enter pet symptoms and receive an AI-generated analysis indicating if urgent veterinary care is needed.
F011	The system should provide pet care resources such as articles, tips, and guides.
F012	The system should allow pet owners to give feedback on the app's functionality and usability.

4.3.2 Non-Functional Requirements

Table 4.4: Non-Functional Requirements

ID	Non-Functional Requirements	Category
NF001	Under normal conditions, the system must respond to user input in less than two seconds.	Performance
NF002	For both tech-savvy and non-tech-savvy users, the front end of the application must be easy to use, visually appealing, and clear.	Usability
NF003	The system must ensure secure data transmission and storage, protecting user credentials and pet health data.	Security
NF004	The system's code should be modular, well-documented, and easily extendable for future updates and feature additions.	Maintainability
NF005	The system must allow 1000 users and data without significant performance degradation.	Scalability

4.4 Use Case Diagram

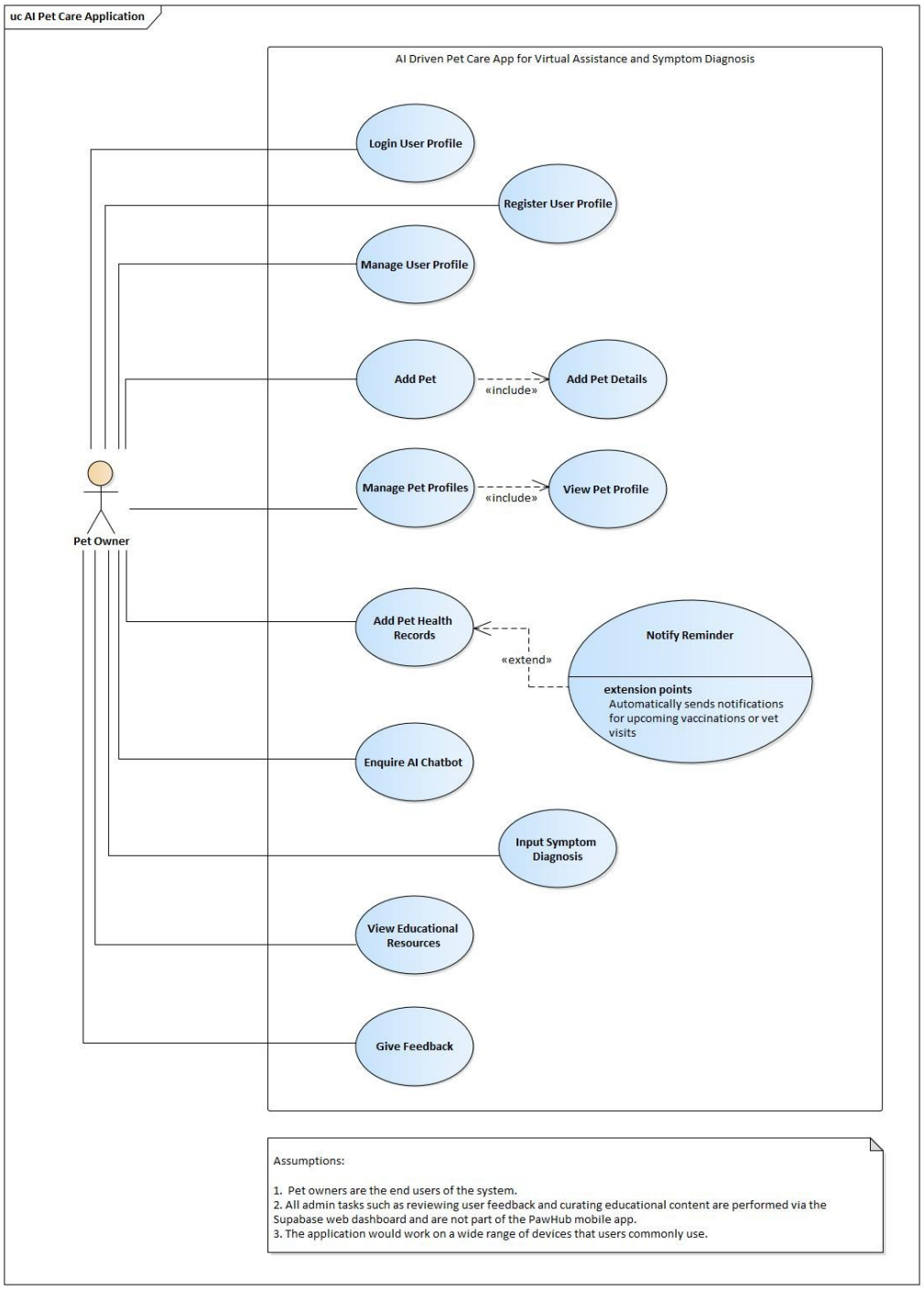


Figure 4.26: Use Case Diagram

4.5 Use Case Description

Based on requirements specifications and use case diagram, the use case description are designed and included in this section. Overall, this section shows the flows of event between users (pet owners) and the system for every use case features.

4.5.1 UC001: Login User Profile

Table 4.5: UC001 Login User Profile

<table>
<tr><td colspan="2">Use Case Name: Login User Profile</td><td>ID:
UC001</td><td>Importance Level:
High</td></tr>
<tr><td colspan="2">Primary Actor: Pet Owner</td><td colspan="2">Use Case Type: Detail, Real</td></tr>
<tr><td colspan="4">Stakeholders and Interests:
Pet Owners – want secure and quick access to their account.</td></tr>
<tr><td colspan="4">Brief Description: This use case describes the process of allowing an existing user to access their account by verifying their login information.</td></tr>
<tr><td colspan="4">Trigger: A user wants to log into their existing account.</td></tr>
<tr><td colspan="4">Relationships:
Association : Pet Owner
Include : N/A
Extend : N/A
Generalization: N/A</td></tr>
<tr><td colspan="4">Normal Flow of Events:
1. The user presses the “Login” function.
2. The system ask the user to give their email and password.
3. The system checks the completion of data entered by the user. Continue to S-1.
4. The system validates the format of the information written by the</td></tr>
</table>

user. Continue to S-2.

5. The system checks if they user credentials match the database. Continue to S-3.
6. The system brings the user to the home page.

Sub-flows**:**

S-1: Perform 3.1 or 3.2

3.1 If the information is empty, the system will show an error message of incomplete data. Continue to flow no 2.

3.2 If the information is complete, proceed to flow no 4.

S-2: Perform 4.1 or 4.2

4.1 If all the information is in the correct format, proceed to flow no.5.

4.2 If the information format is incorrect, the system displays an error message beside and ask to re-enter again. Continue to flow no.2.

S-3: Perform 5.1 or 5.2

5.1 If the credentials match the records in the database. proceed to flow no.6.

5.2 If the credentials are not found in the database, the system displays an error message. Continue to flow no.2.

Alternate/Exceptional Flows:

- A message advising the user to try again later is displayed in the event that a system fault arises during database validation.

4.5.2 UC002: Register User Profile

Table 4.6: Register User Profile

Use Case Name: **Register User Profile**	ID: **UC002**	Importance Level: **High**
Primary Actor: **Pet Owner**	Use Case Type: **Detail, Real**	
Stakeholders and Interests: **Pet Owners – want to create a new profile easily and securely.**		
Brief Description: **This use case explains how to let a new user to register by giving the necessary information.**		
Trigger: **A new user wants to create an account.**		
Relationships: Association : **Pet Owner** Include : **N/A** Extend : **N/A** Generalization: **N/A**		
Normal Flow of Events: 1. The user picks the the "Register" function. 2. The system prompts the user to enter personal details (e.g., email, password, name, phone number). 3. The system checks the completion of data entered by the user. Continue to S-1. 4. The system validates the format of the entered information. Continue to S-2. 5. The system checks for duplicate account existence in the database. Continue to S-3. 6. The system creates the user account.		

Sub-flows**:**

S-1: Perform 3.1 or 3.2

3.1 A warning message indicating incomplete data is displayed by the system if any required field is left empty. Continue to flow no 2.

3.2 If all required fields are complete, proceed to flow no 4.

S-2: Perform 4.1 or 4.2

4.1 If all the information is in the correct format, proceed to flow no.5.

4.2 If the information format is incorrect, the system displays an error message beside and ask to re-enter again. Continue to flow no.2.

S-3: Perform 5.1 or 5.2

5.1 If no duplicate account exists. proceed to flow no.6.

5.2 If an account with the provided email already exists, the system will show an error message and prompts for a different email. Continue to flow no.2.

Alternate/Exceptional Flows:

- A warning message is displayed and the user is urged to try again later in the event that a system error occurs during account creation.

4.5.3 UC003: Manage User Profile

Table 4.7: Manage User Profile

Use Case Name: **Manage User Profile**	ID: **UC003**	Importance Level: **High**
Primary Actor: **Pet Owner**	Use Case Type: **Detail, Real**	
Stakeholders and Interests: **Pet Owners – want quick and organized access to their pet's profile and information.**		
Brief Description: **This use case explains how to let a user change their personal information.**		
Trigger: **A logged-in user wants to update their profile details.**		
Relationships: Association : **Pet Owner** Include : **N/A** Extend : **N/A** Generalization: **N/A**		
Normal Flow of Events: 1. The user presses the "Manage User Profile" button. 2. The current profile details of the user is shown by the system. 3. The user edits their wanted informations fields. 4. The system checks the completeness of the new data. Continue to S-1. 5. The system validates the format of the new information. Continue to S-2. 6. The system saves the updated profile information.		

Sub-flows**:**

S-1: Perform 4.1 or 4.2

4.1 If the updated information is complete, proceed to flow no.5.

4.2 The system shows an error message and asks the user to fill out all necessary fields if the revised information is not full. Continue to flow no.2.

S-2: Perform 5.1 or 5.2

5.1 If the updated information is in the correct format. proceed to flow no.6.

5.2 The system asks the user to re-enter the right format and shows an error message if the information format is incorrect. Continue to flow no.2.

Alternate/Exceptional Flows:

- If an error while saving the profile, it notifies the user and suggests trying the update again later.

4.5.4 UC004: Add Pet

Table 4.8: UC004: Add Pet

<table>
<tr><td colspan="2">Use Case Name: Add Pet</td><td>ID:
UC004</td><td>Importance Level:
High</td></tr>
<tr><td colspan="2">Primary Actor: Pet Owner</td><td colspan="2">Use Case Type: Detail, Real</td></tr>
<tr><td colspan="4">Stakeholders and Interests:
Pet Owners – want to easily register their pet’s details for health monitoring and management.</td></tr>
<tr><td colspan="4">Brief Description: This use case explains how to allow a user to add a new pet profile by providing necessary pet details.</td></tr>
<tr><td colspan="4">Trigger: A user wants to add a new pet profile to their account.</td></tr>
<tr><td colspan="4">Relationships:
Association : Pet Owner
Include : Add Pet Details
Extend : N/A
Generalization: N/A</td></tr>
<tr><td colspan="4">Normal Flow of Events:
1. The user selects the “Add Pet” function.
2. The user is prompted by the system to input pet information, such as the pet’s name, breed, and age.
3. The system checks for the completeness of the entered data. Continue to S-1.
4. The system validates the format of the pet details. Continue to S-2.
5. The system saves the new pet profile and associates it with the user’s account.</td></tr>
</table>

Sub-flows**:**

S-1: Perform 3.1 or 3.2

3.1 If all required details are provided, proceed to flow no.4.

3.2 If any required pet detail is missing, the system displays an error message indicating incomplete data. Continue to flow no.2.

S-2: Perform 4.1 or 4.2

4.1 If the pet details are in the correct format. proceed to flow no.5.

4.2 If any detail is in the incorrect format, the system displays an error message and asks for re-entry. Continue to flow no.2.

Alternate/Exceptional Flows:

- If the system cannot save the pet profile because of an error, it shows a warning and ask the user to try again later.

4.5.5 UC005: Manage Pet Profile

Table 4.9: UC005: Manage Pet Profile

<table>
<tr><td colspan="2">Use Case Name: Manage Pet Profile</td><td>ID:
UC005</td><td>Importance Level:
High</td></tr>
<tr><td colspan="2">Primary Actor: Pet Owner</td><td colspan="2">Use Case Type: Detail, Real</td></tr>
<tr><td colspan="4">Stakeholders and Interests:
Pet Owners – want to keep their pet information accurate and up to date for better management and tracking.</td></tr>
<tr><td colspan="4">Brief Description: This use case explains the function of allowing a user to update or edit an existing pet profile.</td></tr>
<tr><td colspan="4">Trigger: A user wants to modify details of an existing pet profile.</td></tr>
<tr><td colspan="4">Relationships:
Association : Pet Owner
Include : View Pet Profile
Extend : N/A
Generalization: N/A</td></tr>
<tr><td colspan="4">Normal Flow of Events:
1. The user picks the “Manage Pet Profile” button.
2. A list of pet profiles associated with the user is displayed by the system.
3. The user selects a specific pet profile to edit.
4. The system presents the current pet details for editing.
5. The user modifies the desired fields.
6. The system checks the completeness of the updated information. Continue to S-1.
7. The system validates the format of the updated information. Continue</td></tr>
</table>

to S-2. 8. The system saves the updated pet profile.
Sub-flows**:** **S-1: Perform 6.1 or 6.2** 6.1 If the updated pet details are complete, proceed to flow no.7. 6.2 The system indicates an error and asks for completion if any required fields are left blank. Continue to flow no.4. **S-2: Perform 7.1 or 7.2** 7.1 If the updated details are correctly formatted. proceed to flow no.8. 7.2 The system requests for re-entry and shows an error message if the format is incorrect. Continue to flow no.4.
Alternate/Exceptional Flows: • The user is notified and advised to try again later if the system detects an error when saving updates.

4.5.6 UC006: Add Pet Health Records

Table 4.10: UC006: Add Pet Health Records

Use Case Name**: Add Pet Health Records**	ID**:** **UC006**	Importance Level**:** **High**
Primary Actor**: Pet Owner**	Use Case Type**: Detail, Real**	
Stakeholders and Interests**:** **Pet Owners – want to document their pet's health records efficiently and receive reminders for future vaccinations or appointments.**		
Brief Description**: This use case explains how to add a pet's health record, including an extension that will send out a reminder if future dates (such a vaccine or veterinary appointment) are added.**		
Trigger**: A user wants to record new health information for their pet.**		
Relationships: Association **: Pet Owner** Include **: N/A** Extend **: Notify Reminder** Generalization**: N/A**		
Normal Flow of Events**:** 1. The user selects the "Add Pet Health Records" function. 2. The user is prompted by the system to submit information about their pet health record, such as the date, type, and name of their vaccinations. 3. The system checks for completeness of the entered data. Continue to S-1. 4. The system validates the format and correctness of the date and other details. Continue to S-2.		

5. The system checks if the entered date is in the future. Continue to S-3.
6. The system triggers the Notify Reminder extension.
7. The system saves the health record and associates it with the pet's profile.

Sub-flows**:**

S-1: Perform 3.1 or 3.2

3.1 If all fields are complete, proceed to flow no.4.

3.2 The system indicates incomplete data with an error message if any necessary field is left empty. Continue to flow no.2.

S-2: Perform 4.1 or 4.2

4.1 If the information is in the correct format, proceed to flow no.5.

4.2 If any field is incorrectly formatted, the system shows an error warning and the user is asked to enter the information again. Continue to flow no.2.

S-3: Perform 5.1 or 5.2

5.1 If the health record includes a future date, the system automatically schedules a reminder for the event. proceed to flow no.6.

5.2 If the date is not in the future, no reminder is scheduled, and the system proceeds to save the record. proceed to flow no.7.

Alternate/Exceptional Flows:

- If an error occurs during the reminder scheduling or saving process, the system displays an error message and suggests trying again later.

4.5.7 UC007: Enquire AI Chatbot

Table 4.11: UC007: Enquire AI Chatbot

<table>
<tr><td colspan="2">Use Case Name: Enquire AI Chatbot</td><td>ID:
UC007</td><td>Importance Level:
High</td></tr>
<tr><td colspan="2">Primary Actor: Pet Owner</td><td colspan="2">Use Case Type: Detail, Real</td></tr>
<tr><td colspan="4">Stakeholders and Interests:
Pet Owners – want fast, AI-powered responses to general pet care questions to support daily care decisions.</td></tr>
<tr><td colspan="4">Brief Description: This use case explains the interaction with the AI chatbot to ask questions and receive pet care advice.</td></tr>
<tr><td colspan="4">Trigger: A user wants to get advice or information about pet care from the AI chatbot.</td></tr>
<tr><td colspan="4">Relationships:
Association : Pet Owner
Include : N/A
Extend : N/A
Generalization: N/A</td></tr>
<tr><td colspan="4">Normal Flow of Events:
1. The user selects the “AI Chatbot” option.
2. The system opens the AI chat interface, including a dropdown menu for model selection.
3. The user selects one of the available AI models.
4. The user enters a pet care-related question into the input box.
5. The system performs a preliminary validation of the input (e.g., checking for non-empty and valid characters). Continue to S-1.
6. The system constructs an API request that includes both the selected</td></tr>
</table>

AI model and the user's query.

7. The system sends the API request to the corresponding OpenRouter AI model endpoint based on the user's selected model.
8. The system waits for the response from the external AI API.
9. Upon receiving the response, the system processes and formats the returned text for clarity and display.
10. The system displays the AI-generated response to the user in the chat interface.

Sub-flows**:**

S-1: Perform 5.1 or 5.2

5.1 If the user's input is valid (non-empty and acceptable format), proceed to flow no.6.

5.2 If the input is invalid (empty or contains disallowed characters), the system will show a warning and ask the user to enter the query again. Continue to flow no.4.

Alternate/Exceptional Flows:

A-1: API Timeout or Failure

- If the external AI API fails to respond within the expected time or returns an error, the system displays a message indicating that the service is temporarily unavailable and suggests using another model.

A-2: Malformed API Response

- If the AI API returns an incorrect or partial result, the system alerts the user and records the error for future analysis, with the option to try the request again.

4.5.8 UC008: Input Symptom Diagnosis

Table 4.12: UC008: Input Symptom Diagnosis

<table>
<tr><td colspan="2">Use Case Name: Input Symptom Diagnosis</td><td>ID:
UC008</td><td>Importance Level:
High</td></tr>
<tr><td colspan="2">Primary Actor: Pet Owner</td><td colspan="2">Use Case Type: Detail, Real</td></tr>
<tr><td colspan="4">Stakeholders and Interests:
Pet Owners – want to receive fast and reliable suggestions when their pet shows signs of illness.</td></tr>
<tr><td colspan="4">Brief Description: This use case explains the function of allowing a user to input pet symptoms and receive a preliminary diagnosis whether the pet needs immediate medical attention.</td></tr>
<tr><td colspan="4">Trigger: A user observes unusual symptoms in their pet and wants a preliminary diagnosis.</td></tr>
<tr><td colspan="4">Relationships:
Association : Pet Owner
Include : N/A
Extend : N/A
Generalization: N/A</td></tr>
<tr><td colspan="4">Normal Flow of Events:
1. The user selects the “Symptom Diagnosis” button.
2. The user is prompted by the system to enter the symptoms they have noticed.
3. The user inputs some details of the symptoms faced by their pet.
4. The system checks for completeness of the entered data. Continue to S-1.
5. The system validates the format and relevance of the symptoms.</td></tr>
</table>

Continue to S-2.

6. The system processes the input and generates a preliminary diagnosis on whether the pet need immediate medical attention.
7. The system displays the preliminary diagnosis and any recommended next steps.

Sub-flows**:**

S-1: Perform 4.1 or 4.2

4.1 If all required symptom details are provided, proceed to flow no.5.

4.2 If details are incomplete, the system displays an error message and asks for additional information. Continue to flow no.2.

S-1: Perform 5.1 or 5.2

5.1 If the symptoms are clearly described and in the correct format. proceed to flow no.6.

5.2 If the description is incomplete, the system asks the user to clarify or add more details. Continue to flow no.2.

Alternate/Exceptional Flows:

- If the system is unable to provide a clear diagnosis, it may recommend the user consult a veterinary professional immediately.

4.5.9 UC009: View Educational Resources

Table 4.13: UC009: View Educational Resources

Use Case Name**: View Educational Resources**	ID**:** **UC009**	Importance Level**:** **High**
Primary Actor**: Pet Owner**	Use Case Type**: Detail, Real**	
Stakeholders and Interests**:** **Pet Owners – want easy access to trusted pet care articles and guides to improve their knowledge and care practices.**		
Brief Description**: This use case explains how to let a user to browse and view educational resources related to pet care.**		
Trigger**: A user wants to learn more about pet care through articles or training guides.**		
Relationships: Association **: Pet Owner** Include **: N/A** Extend **: N/A** Generalization**: N/A**		
Normal Flow of Events**:** 1. The user selects the "Educational Resources" option. 2. The system displays a list or menu of available educational resources after web scraping. 3. The user browses or uses the search/filter functionality to find relevant topics. 4. The user selects a specific resource to view. 5. The system displays the detailed content of the selected resource.		

Sub-flows**:**
Alternate/Exceptional Flows: • If no resources match the user's search criteria, the system displays a "No educational resources found" message with suggestions to broaden the search.

4.5.10 UC010: Give Feedback

Table 4.14: UC010: Give Feedback

Use Case Name**: Give Feedback**	ID**:** **UC010**	Importance Level**:** **High**
Primary Actor**: Pet Owner**	Use Case Type**: Detail, Real**	
Stakeholders and Interests**:** **Pet Owners – want to share their experiences, report issues, or suggest improvements to help enhance the app's quality.**		
Brief Description**: This use case describes the process of allowing a user to submit feedback or report their experience with the application.**		
Trigger**: A user decides to provide feedback about their experience using the application.**		
Relationships: Association **: Pet Owner** Include **: N/A** Extend **: N/A** Generalization**: N/A**		
Normal Flow of Events**:** 1. The user selects the "Feedback" option. 2. The user is prompted by the system to submit a report, comments, or ideas. 3. The user inputs their feedback. 4. The system checks for completeness of the feedback submission. Continue to S-1. 5. The system records and stores the feedback.		

Sub-flows**:** **S-1: Perform 4.1 or 4.2** 4.1 If the feedback is complete, proceed to flow no.5. 4.2 The system shows an error notice and asks the user to finish the submission if the feedback is lacking any necessary information. Continue to flow no.2.
Alternate/Exceptional Flows: • The user is notified and given the option to try again later if the system detects an error when recording the feedback.

4.6 User Interface (UI) Prototype

This section presents a simple wireframe prototype developed using Figma for the PawHub application. The prototype illustrates the overall layout, navigation flow, and key interface elements designed to ensure a user-friendly and accessible experience. It serves as a visual guide for the app's structure before moving into full development.

Figure 4.27: Overall PawHub Prototype

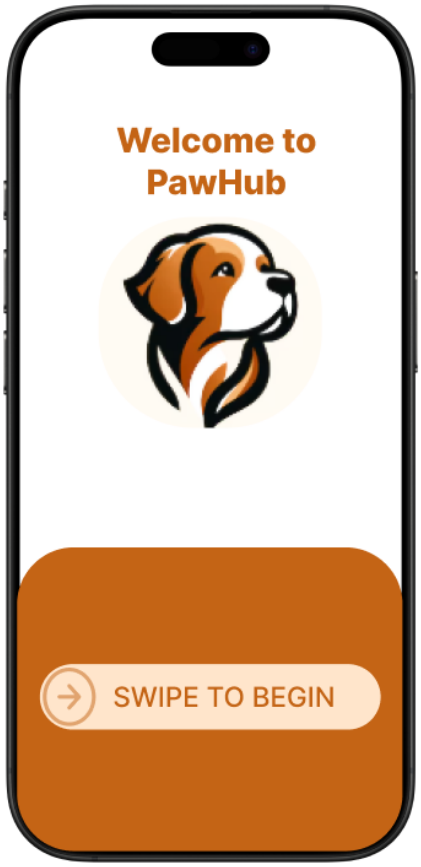


Figure 4.28: Welcome Screen

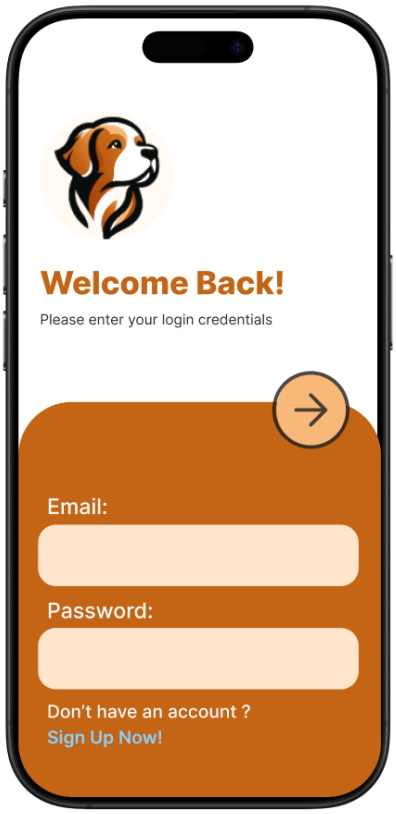


Figure 4.29: Login Screen

Figure 4.30: Register Screen

Figure 4.31: Home Screen and Navigation

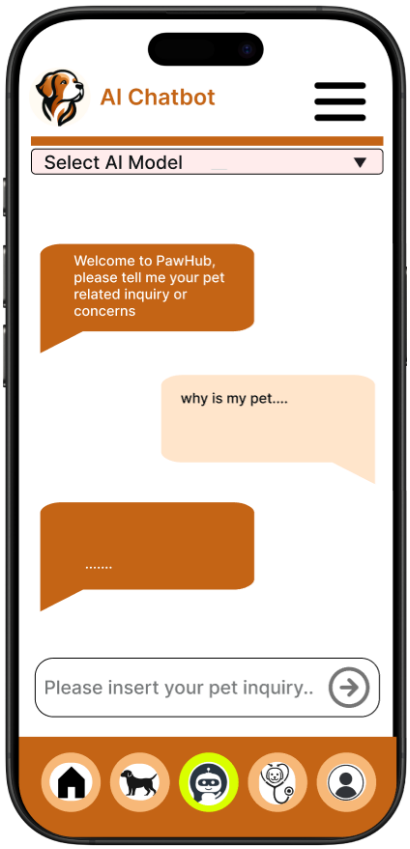


Figure 4.32: AI Chatbot Screen

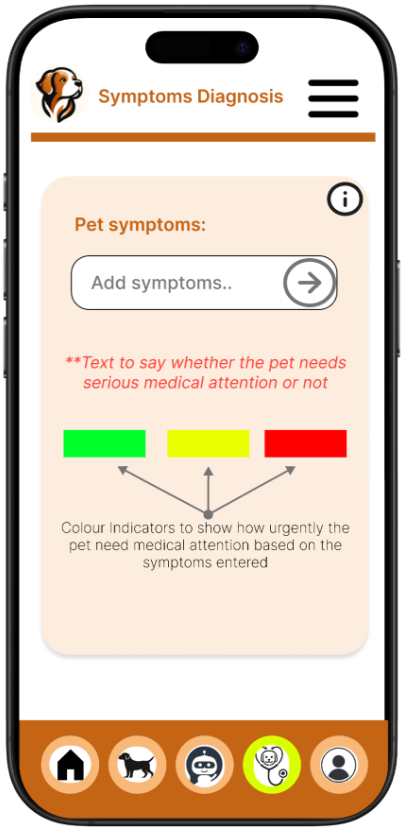


Figure 4.33: Symptom Diagnosis Screen

Figure 4.34: Pet Profile Management Screen

Figure 4.35: User Profile Management Screen

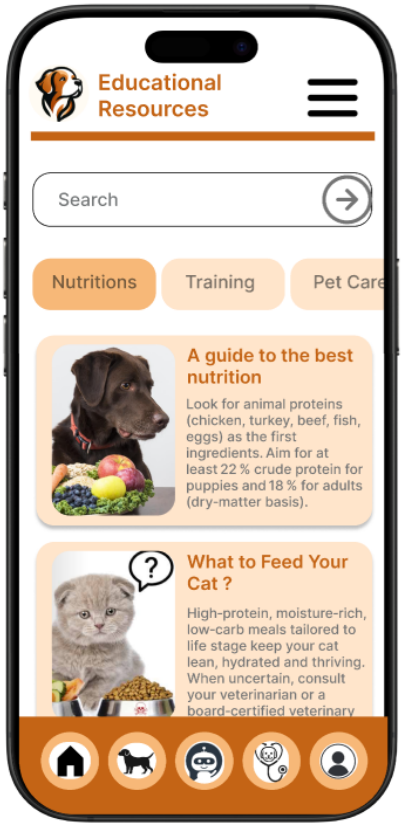


Figure 4.36: Educational Resources Screen

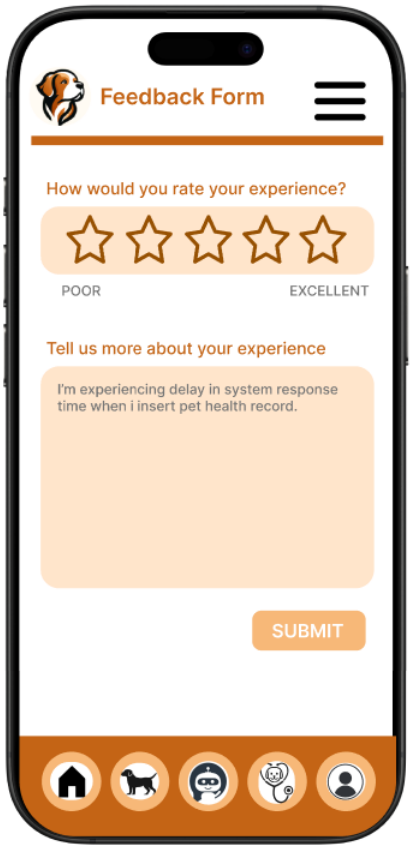


Figure 4.37: Feedback Screen

CHAPTER 5

SYSTEM DESIGN

5.1 Introduction

In this chapter the system architecture, database diagram, data dictionary, API, Endpoints, Data Flow Diagram (ERD), Activity Diagram, Mobile Application Design Principles and User Interface Design were all covered in detail. To show the relationships and structuring of data, it contains Activity Diagrams, Data Flow Diagrams (DFD), Entity Relationship Diagrams (ERD), and a data dictionary. The user interface decisions are explained using the principles of mobile application design which focuses on responsiveness, consistency, and usability. Additionally, layouts for the application's user interface are used to show the application's navigation, layout, and overall user experience.

5.2 System Architecture

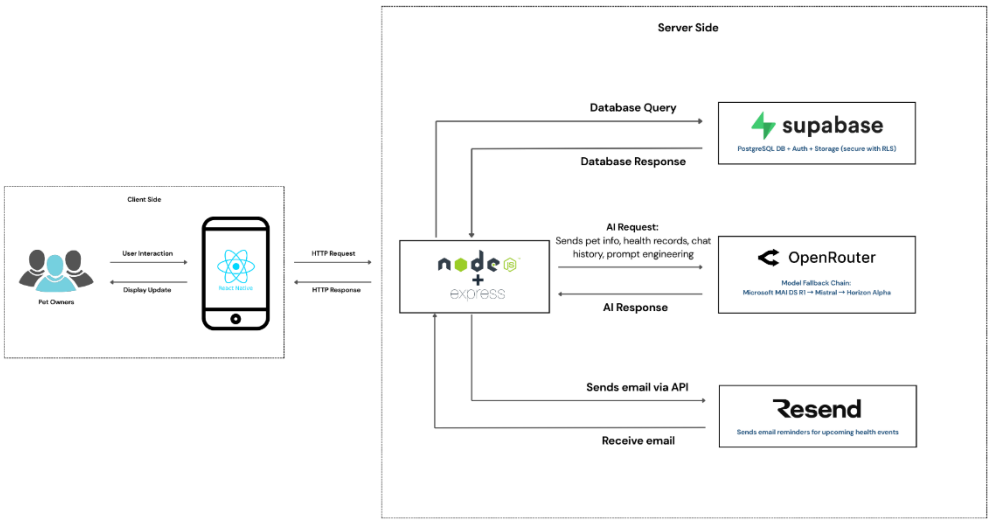


Figure 5.1: System Architecture

The PawHub system architecture is a well-structured, secure, and intelligent mobile application ecosystem designed to assist pet owners in managing their pets' health through AI-powered virtual assistance. At the client side, the React Native mobile app served as the user interface, allowing pet owners to

interact with features such as symptom diagnosis, AI chatbot conversations, pet profile management, and health record tracking. All user requests are securely transmitted to the Node.js Express backend server, which acts as the central control layer, handling authentication, business logic, and communication with external services.

The backend ensured data integrity and security by validating requests, enforcing rate limits, and using JWT-based authentication to verify user identity. It communicates with Supabase, which functions as the primary database and authentication provider, storing user profiles, pet information, health records, symptom history, chat logs, and feedback, to guarantee that users may only access their own data, all are safeguarded by Row Level Security (RLS). For AI-powered functionality, the backend integrates with OpenRouter.ai, sending structured prompts that include pet details and symptoms to advanced language models such as Microsoft MAI DS R1, Mistral, and Horizon Alpha. The system employed a Model Control & Prioritization (MCP) strategy, automatically falling back to alternative models if the primary one fails, ensuring reliable and continuous AI responses.

When a user submits symptoms, the AI returns a diagnosis in a strictly formatted structure Diagnosis, Severity, Recommendation, Possible Causes, and Additional Notes which is parsed by the backend and displayed in a clean, user-friendly format within the app. Additionally, the system uses Resend to send automated email reminders for upcoming health events, such as vaccinations, by checking the email_reminders table at regular intervals and triggering emails one day before the scheduled date. This layered architecture separating the frontend, backend, database, and external AI/email services ensures scalability, maintainability, and strong security, making PawHub a robust, real-world-ready pet care solution that effectively combines modern full-stack development with artificial intelligence to deliver practical value to pet owners.

5.3 Database Design

The PawHub application used a secure, relational database built on Supabase (PostgreSQL) to store user, pet, health, and AI interaction data. The profiles table links to Supabase Auth for secure user identification, while keeping profile data like username and avatar separate. The pets table stores pet details and is linked to users via foreign key, supporting multiple pets per user. Health records, including vaccinations and check-ups, are tracked in the health_records table with optional email reminders managed by the email_reminders table.

AI-powered features were supported by symptom_history, which saves AI-generated diagnoses with severity levels, and chat_history, which logs chatbot interactions for continuity and user feedback. Educational content is stored in the articles table, populated weekly by scraping trusted sources like AKC. User feedback is collected in the feedback table to support app improvements. All tables were protected by Row Level Security (RLS) standards, making sure users may only view their own data. This design ensures data integrity, privacy, and scalability, forming a solid foundation for current functionality and future enhancements.

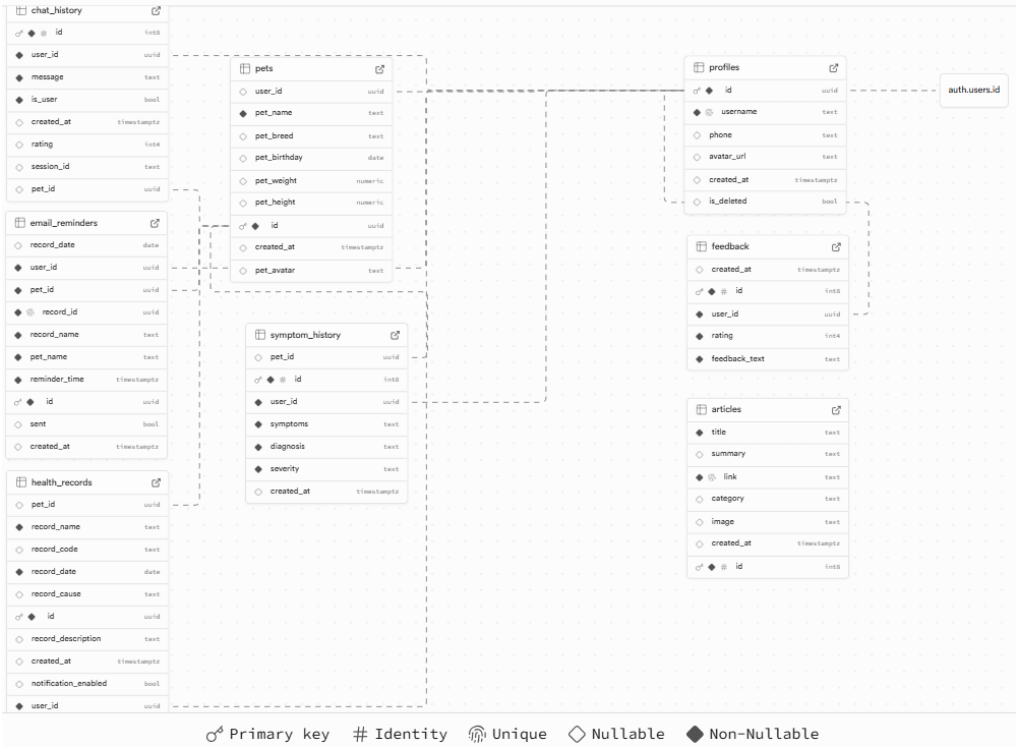


Figure 5.2: ERD Diagram

5.3.1 Data Dictionary

5.3.1.1 auth.users.id

This table is managed by Supabase Auth. It is not directly accessed by the app, only through secure authentication flows.

Table 5.1: Authenticated Users Table

Field	Type	Null	Default	Description
id	uuid	NO	-	User's unique identification
email	text	NO	-	User's Email address
encrypted_password	text	NO	-	Password hash stored securely (not visible or accessible)

5.3.1.2 profiles

Stores user account information linked to Supabase Auth.

Table 5.2: Users Information Table

Field	Type	Null	Default	Description
id	uuid	NO	-	Primary key; references auth.users(id)
username	text	NO	-	Unique username chosen by the user
phone	text	YES	NULL	User's contact number
avatar_url	text	YES	NULL	URL to profile picture in Supabase Storage
created_at	timestamptz	NO	timezone('utc'::text, now())	Account creation timestamp
is_deleted	boolean	NO	false	Flag to mark soft-deleted accounts

Constraints:

- profiles_pkey: Primary key on id
- profiles_id_fkey: Foreign key to auth.users(id)

5.3.1.3 pets

Stores pet profile information owned by a user.

Table 5.3: Pets Information Table

Field	Type	Null	Default	Description
user_id	uuid	YES	NULL	Owner of the pet
pet_name	text	NO	-	Name of the pet
pet_breed	text	YES	NULL	Type of the pet
pet_birthday	date	YES	NULL	Date of birth
pet_weight	numeric	YES	NULL	Weight in kilograms
pet_height	numeric	YES	NULL	Height in centimeters
id	uuid	NO	gen_random_uuid()	Unique pet identifier
created_at	timestamptz	NO	timezone('utc'::text, now())	Timestamp when pet was added
pet_avatar	text	YES	NULL	URL to pet's profile image

Constraints:

- pets_pkey: Primary key on id
- pets_user_id_fkey: Foreign key to profiles(id)

5.3.1.4 health_records

Tracks medical events such as vaccinations, check-ups, and treatments.

Table 5.4: Pets Health Record Information Table

Field	Type	Null	Default	Description
pet_id	uuid	YES	NULL	Reference to the pet
record_name	text	NO	-	Name of the health event (e.g., "Vaccination")
record_code	text	YES	NULL	Optional medical code
record_date	date	NO	-	Scheduled or past date of the event
record_cause	text	YES	NULL	Reason for the visit (e.g., "Routine check-up")
id	uuid	NO	gen_random_uuid()	Unique record ID
record_description	text	YES	NULL	Notes or details about the event
created_at	timestamptz	NO	timezone('utc'::text, now())	Timestamp when record was created
notification_enabled	boolean	NO	true	Whether to send email reminder

Constraints:

- health_records_pkey: Primary key on id
- health_records_pet_id_fkey: Foreign key to pets(id)
- health_records_user_id_fkey: Foreign key to profiles(id)

5.3.1.5 symptom_history

Stores AI-generated diagnoses from user-submitted symptoms.

Table 5.5: Pets AI Symptoms Diagnosis History Table

Field	Type	Null	Default	Description
pet_id	uuid	YES	NULL	Reference to the pet
id	bigint	NO	GENERATED ALWAYS AS IDENTITY	Auto-incrementing ID
user_id	uuid	NO	-	Owner who submitted the symptoms
symptoms	text	NO	-	User's input (e.g., "vomiting, lethargic")
diagnosis	text	NO	-	AI-generated diagnosis
severity	text	NO	-	Risk level: low, moderate, or high
created_at	timestamptz	NO	now()	Timestamp of diagnosis

Constraints:

- symptom_history_pkey: Primary key on id
- symptom_history_user_id_fkey: Foreign key to profiles(id)
- symptom_history_pet_id_fkey: Foreign key to pets(id)

5.3.1.6 chat_history

Stores messages from the AI chatbot for continuity and rating.

Table 5.6: AI Chatbot Message History Table

Field	Type	Null	Default	Description
id	bigint	NO	GENERATED ALWAYS AS IDENTITY	Auto-incrementing message ID
user_id	uuid	NO	-	User who sent the message

message	text	NO	-	Message content
is_user	boolean	NO	-	trueif from user,falseif from AI
created_at	timestamptz	NO	now()	Timestamp of message
rating	integer	YES	NULL	User rating (1–5 stars) for bot message
session_id	text	YES	NULL	Group messages by chat session
pet_id	uuid	YES	NULL	Contextual pet for AI response

Constraints:

- chat_history_pkey: Primary key on id
- chat_history_user_id_fkey: Foreign key to profiles(id)
- chat_history_pet_id_fkey: Foreign key to pets(id)

5.3.1.7 email_reminders

Manages scheduled email notifications for upcoming health events.

Table 5.7: Email Reminders Table

Field	Type	Null	Default	Description
record_date	date	YES	NULL	Date of the health event
user_id	uuid	NO	-	Recipient user
pet_id	uuid	NO	-	Pet associated with the reminder
record_id	uuid	NO	-	Unique; referenceshealth_records(id)
record_name	text	NO	-	Name of the event (e.g., “Dental Check-up”)
pet_name	text	NO	-	Name of the pet (denormalized for email)
reminder_time	timestamptz	NO	-	When the email should be sent (e.g., 1 day before)

id	uuid	NO	gen_random_uuid()	Unique reminder ID
sent	boolean	NO	false	Whether the email has been sent
created_at	timestamptz	NO	now()	When reminder was created

Constraints:

- email_reminders_pkey: Primary key on id
- email_reminders_user_id_fkey: Foreign key to profiles(id)
- email_reminders_pet_id_fkey: Foreign key to pets(id)
- record_id is unique per user

5.3.1.8 feedback

Stores user feedback for app improvement.

Table 5.8: User Feedback Table

Field	**Type**	**Null**	**Default**	**Description**
created_at	timestamptz	NO	CURRENT_TIMESTAMP	When feedback was submitted
id	bigint	NO	GENERATED ALWAYS AS IDENTITY	Auto-incrementing ID
user_id	uuid	NO	-	Submitter
rating	integer	NO	-	1 to 5 stars
feedback_text	text	NO	-	User's written feedback

Constraints:

- feedback_pkey: Primary key on id
- feedback_user_id_fkey: Foreign key to profiles(id)
- rating must be between 1 and 5

5.3.1.9 articles

Stores curated pet care articles scraped from external sources (e.g., AKC).

Table 5.9: Web-scraped Articles Table

Field	Type	Null	Default	Description
title	text	NO	-	Article title
summary	text	YES	NULL	Brief description
link	text	NO	-	Unique URL to original article
category	text	YES	NULL	Topic (e.g., “Training”, “Nutrition”)
image	text	YES	NULL	Thumbnail image URL
created_at	timestamptz	NO	now()	When article was added to database
id	bigint	NO	GENERATED ALWAYS AS IDENTITY	Auto-incrementing ID

Constraints:

- articles_pkey: Primary key on id
- link must be unique

5.4 API Endpoints

To enable communication between the React Native frontend and backend services, the PawHub application made use of a secure RESTful API developed with Node.js and Express.js. JWT-based authentication middleware safeguards all endpoints, guaranteeing that only authorized users can access or alter data. Rate limiting was implemented to prevent abuse, with specific limits for authentication (20 attempts per 15 minutes), general usage (200 requests per 15 minutes), and chat functionality (20 requests per minute). The API integrates with Supabase for data persistence, OpenRouter for AI-powered symptom diagnosis, and Resend for automated email reminders. All requests required a valid Bearer token obtained during login, and responses were returned in JSON format.

5.4.1 Authentication Endpoints

Table 5.10: Authentication Endpoints

Method	Endpoint	Description
POST	/auth/login	Gets the JWT token after authenticating the user with their email and password.
POST	/auth/register	Creates a new user account and corresponding profile entry
POST	/auth/forgot-password	Initiates password reset flow via email

5.4.2 Profile Management Endpoints

Table 5.11: Profile Management Endpoints

Method	Endpoint	Description
GET	/auth/profile	Retrieves user profile information including username and avatar URL
PUT	/profile	Updates user's profile information.
POST	/profile/avatar	Uploads a new profile picture from a multipart/form-data request
POST	/profile/avatar/default	Sets a predefined default avatar by

		updating the avatar_url with the URL of one of the four default user images hosted in Supabase Storage.
DELETE	/profile	Soft deletes the user account by setting the is_deleted flag to true in the Profiles table.

5.4.3 Pet Management Endpoints

Table 5.12: Pet Management Endpoints

Method	**Endpoint**	**Description**
GET	/pets	Retrieves all pets belonging to the authenticated user
POST	/pets	Creates a new pet profile or updates an existing one
DELETE	/pets/:id	Deletes a pet and all associated health records

5.4.4 Health Records Endpoints

Table 5.13: Health Records Endpoints

Method	**Endpoint**	**Description**
GET	/health-records/:petId	Retrieves all health records for a specific pet
POST	/health-records	Creates a new health record with optional email reminder
DELETE	/health-records/:id	Deletes a specific health record

5.4.5 Feedback Endpoints

Table 5.14: Feedback Endpoints

Method	Endpoint	Description
GET	/feedback	Retrieves all feedback for logged-in user from database
POST	/feedback	Submits or updates user feedback with rating
DELETE	/feedback/:id	Deletes a feedback entry

5.4.6 Symptom Diagnosis Endpoints

Table 5.15: Symptom Diagnosis Endpoints

Method	Endpoint	Description
POST	/symptom/diagnose	Sends symptoms to AI model and returns structured diagnosis
GET	/symptom-history	Retrieves user's past symptom diagnoses
POST	/symptom-history	Saves a new symptom diagnosis to history
DELETE	/symptom-history/:id	Deletes a specific symptom history entry

5.4.7 AI Chatbot Endpoints

Table 5.16: AI Chatbot Endpoints

Method	Endpoint	Description
GET	/chat/models	Retrieves list of available AI models for chat
GET	/chat/history	Retrieves chat history for current session
POST	/chat/message	Sends user message to AI and returns response
PUT	/chat/rate/:id	Submits user rating for a chat message
DELETE	/chat/session/:sessionId	Clear a chat session
GET	/chat/sessions	Get user's chat sessions

5.4.8 Other Endpoints

Table 5.17: Other Endpoints

Method	Endpoint	Description
GET	/articles	Retrieves web scraped educational articles from database
GET	/home	Retrieves consolidated data for home screen display
GET	/health	Health check endpoint returning service status

5.5 Data Flow Diagram (DFD)

The PawHub system's information flow is shown by the Data Flow Diagram (DFD), which shows the data flow between users, processes, and the database. It gives a clear picture of the system's data storage, external entities, and functional processes, making it possible to understand how various parts work together. In order to guarantee that all data transactions were managed effectively and securely, the DFD also helps in identifying the pathways via which user inputs are processed, stored, and retrieved. The system's general functioning and detailed process flows can be seen in both the Level 0 and Level 1 diagrams.

5.5.1 Context Diagram

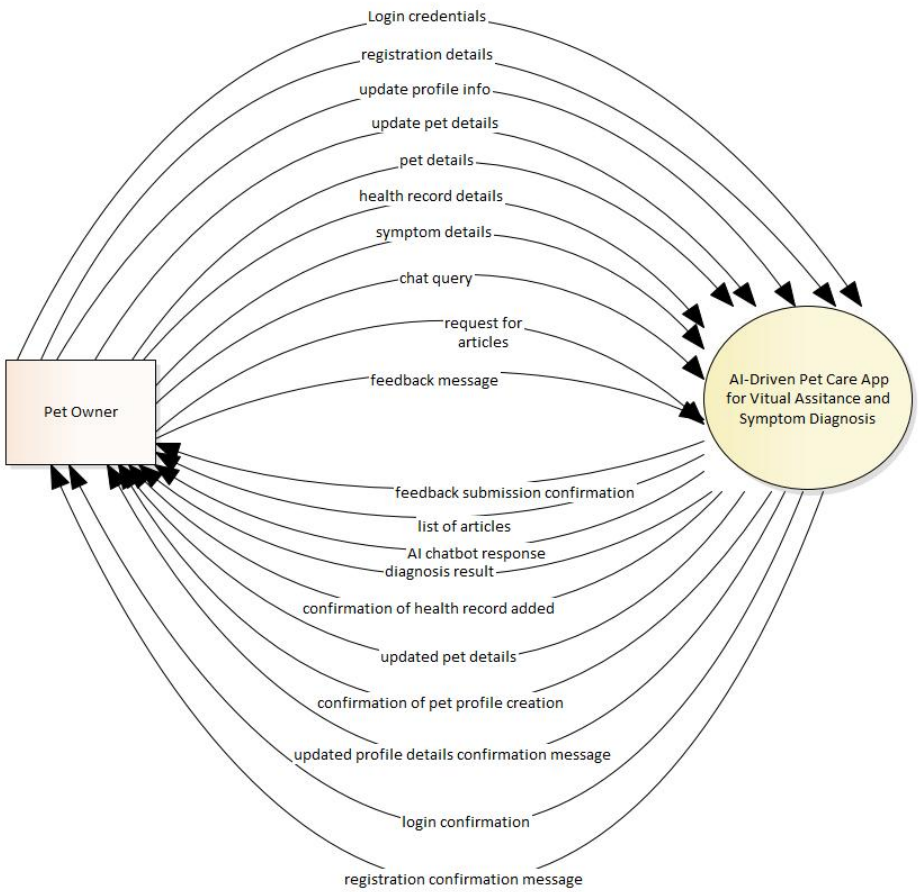


Figure 5.3: Context Diagram

5.5.2 Level-0 Diagram

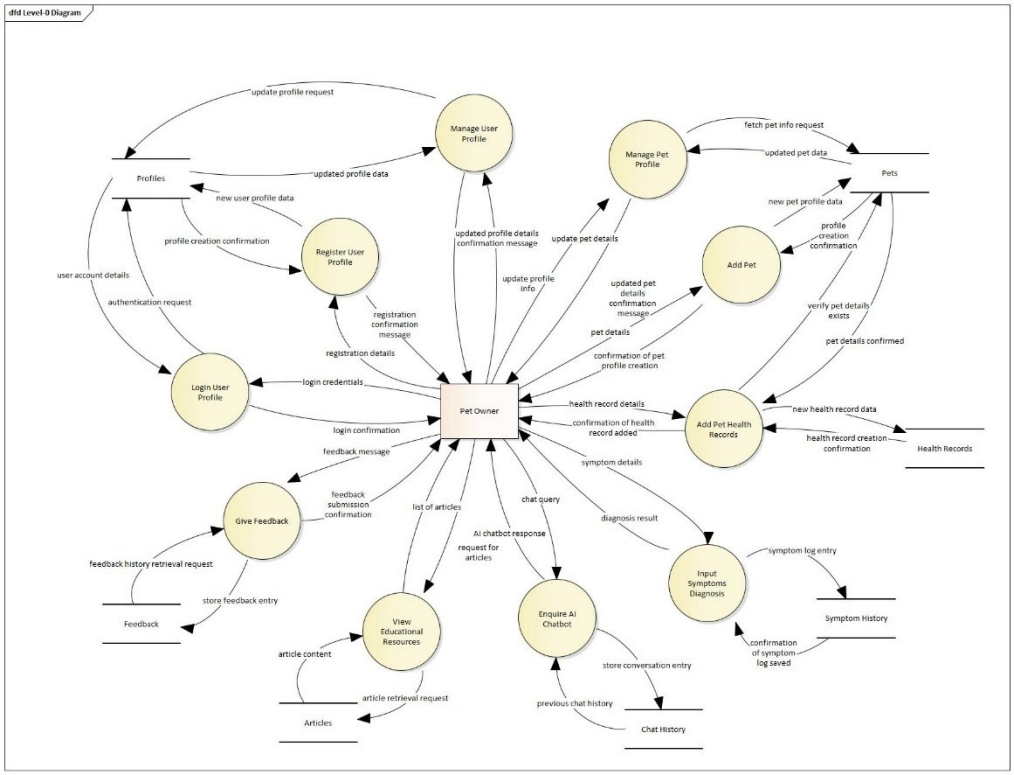


Figure 5.4: Level-0 Diagram

5.5.3 Level-1 Diagram

The PawHub system's primary functions were shown in further detail in the Level 1 Data Flow Diagram, which builds upon the Level 0 diagram's general flow. It interacts with external entities and the system's databases, breaking down certain important processes into smaller ones. This degree of detail helped in explaining the sequential data flow, illustrating the ways in which data is collected, processed, stored, and retrieved to provide the application's intended functionality.

5.5.3.1 Enquire AI Chatbot Level-1 Diagram

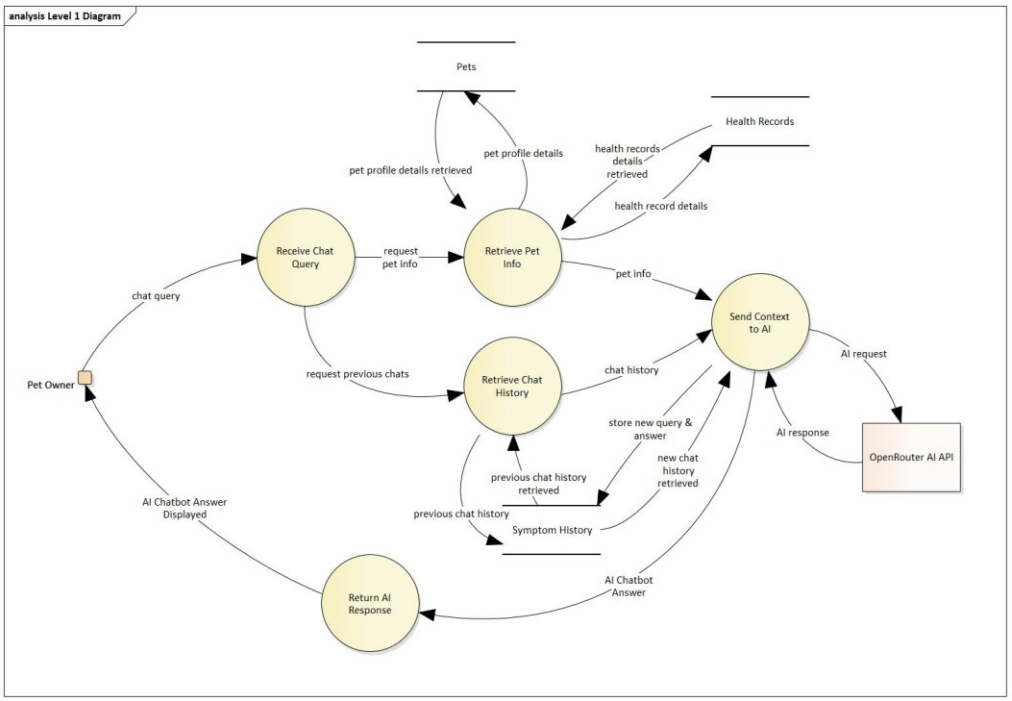


Figure 5.5: Enquire AI Chatbot Level-1 Diagram

5.5.3.2 Input Symptom Diagnosis Level-1 Diagram

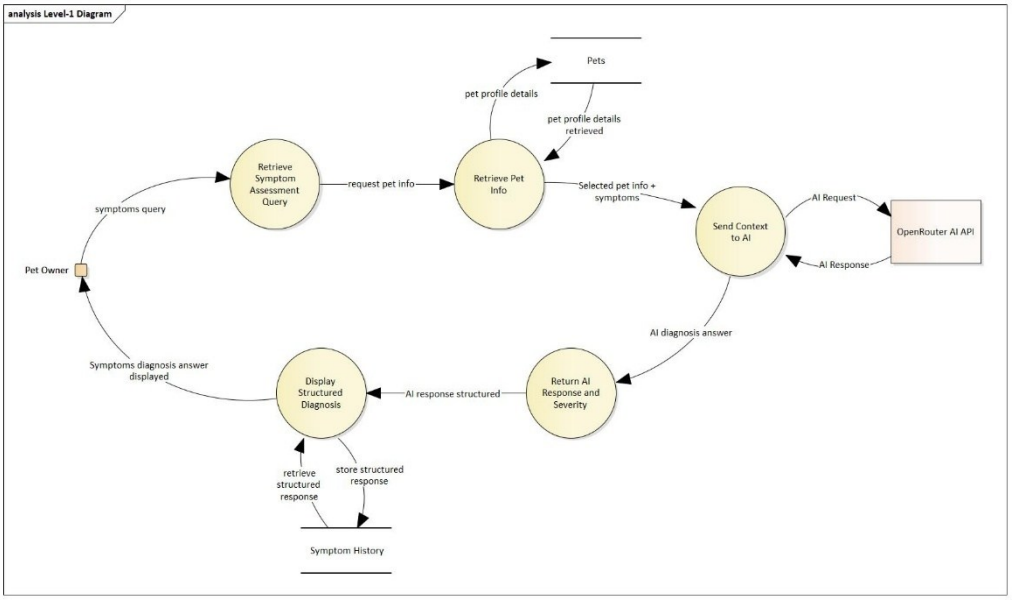


Figure 5.6: Input Symptom Diagnosis Level-1 Diagram

5.5.3.3 Add Pet Health Records Level-1 Diagram

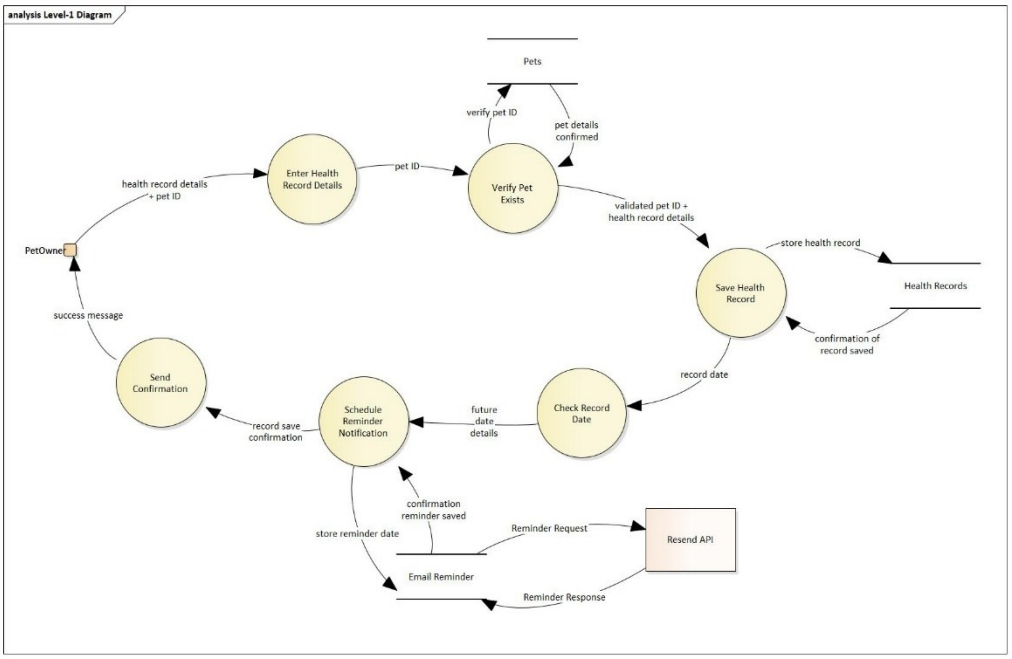


Figure 5.7: Add Pet Health Records Level-1 Diagram

5.6 Activity Diagram

Activity diagrams were used in PawHub to show the process of important user interactions across the main modules. These diagrams used a standardized UML structure to show action sequences, decision points, validations, and system responses. The detailed activity diagrams for the main features were shown in the next sections. These provide an illustration of the operational workflows that are necessary to comprehend the functional behavior of the application.

5.6.1 Login Activity Diagram

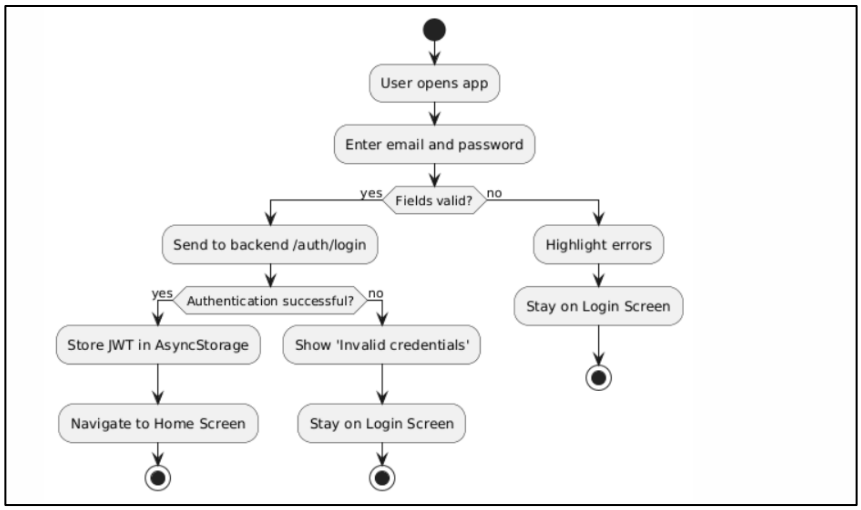


Figure 5.8: Login screen activity diagram

5.6.2 Register Activity Diagram

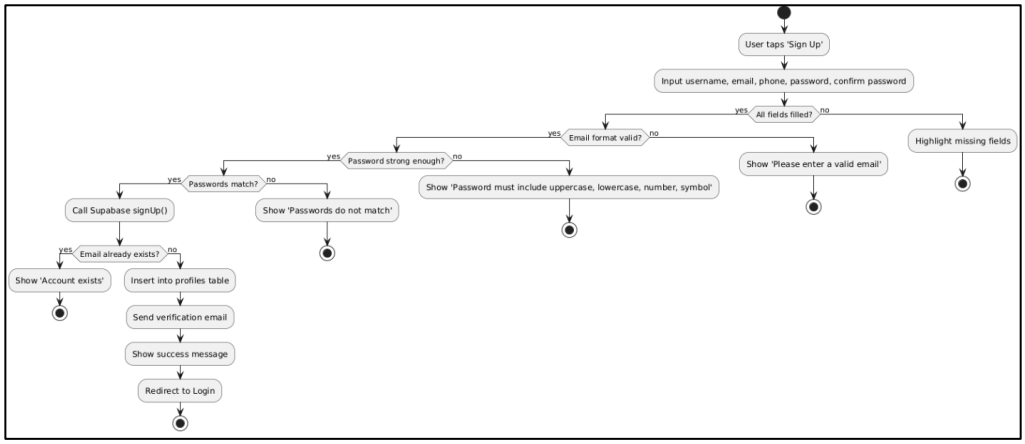


Figure 5.9: Register screen activity diagram

5.6.3 Forgot Password Activity Diagram

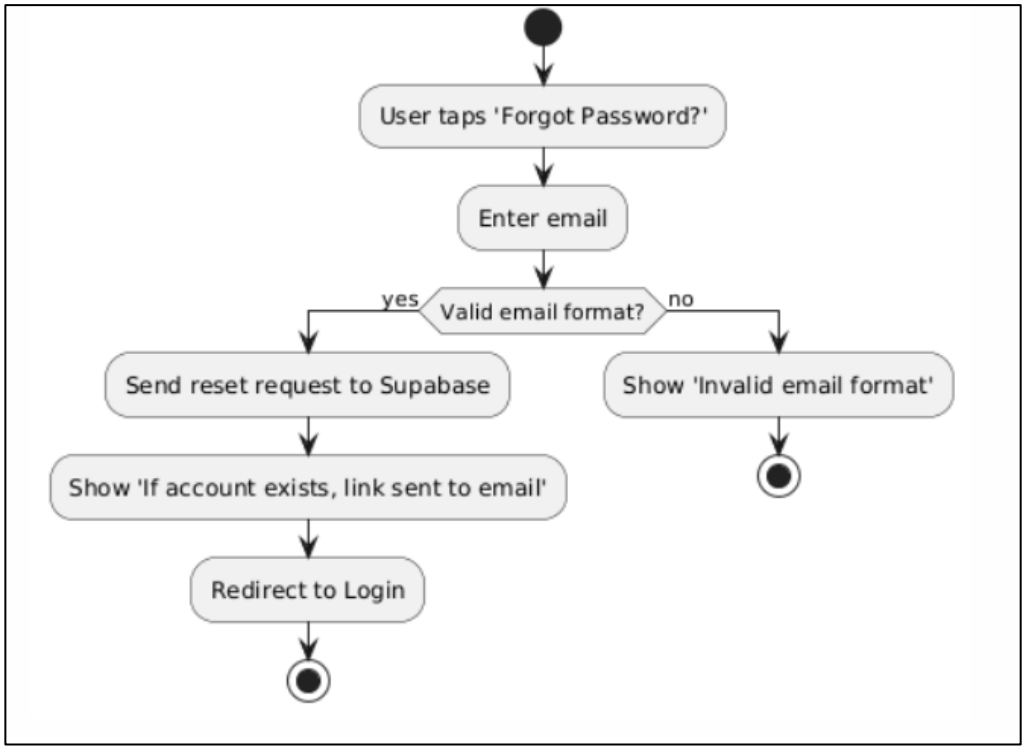


Figure 5.10: Forgot screen activity diagram

5.6.4 Add Pet Activity Diagram

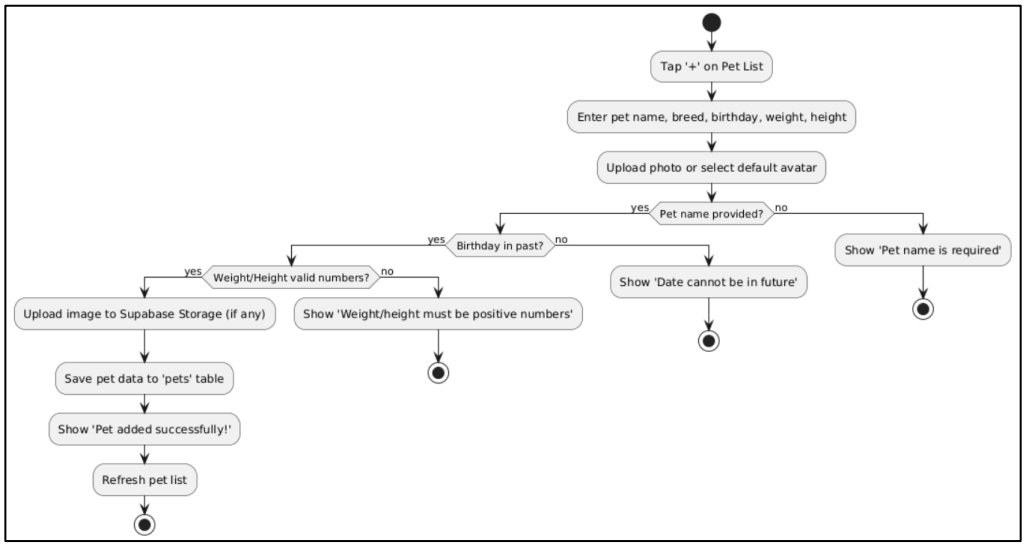


Figure 5.11: Pet Management Screen activity diagram - Add pet

5.6.5 Add Health Record With Reminder Activity Diagram

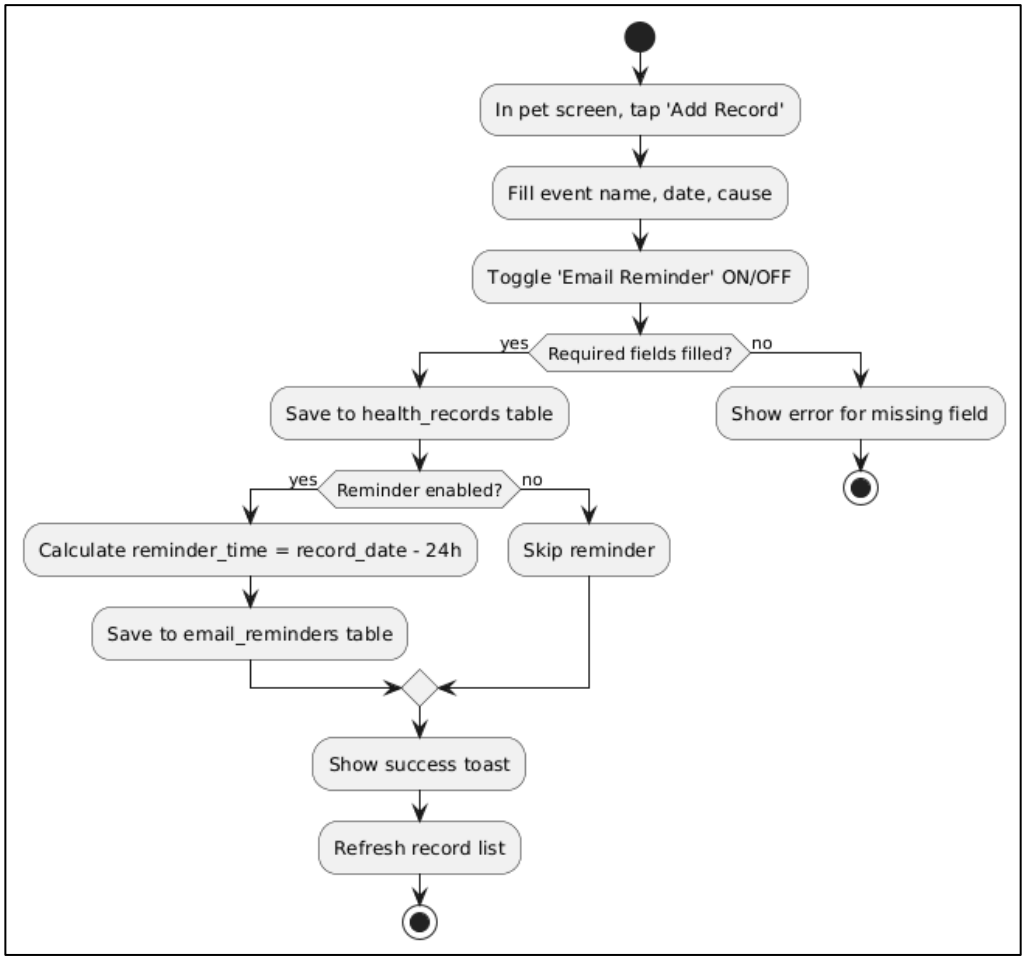


Figure 5.12: Pet Management Screen activity diagram - Add health record with reminder

5.6.6 AI Chatbot Activity Diagram

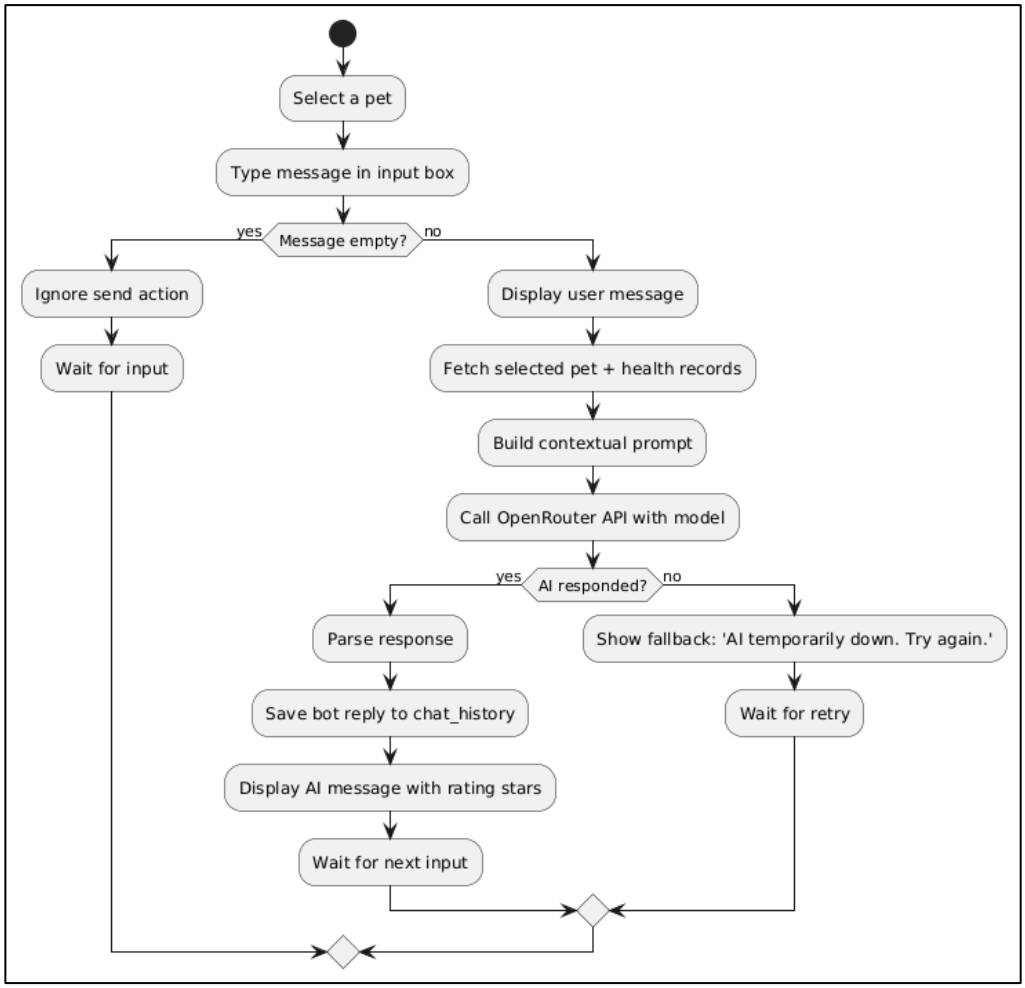


Figure 5.13: AI Chatbot Screen activity diagram

5.6.7 Symptoms Diagnosis Activity Diagram

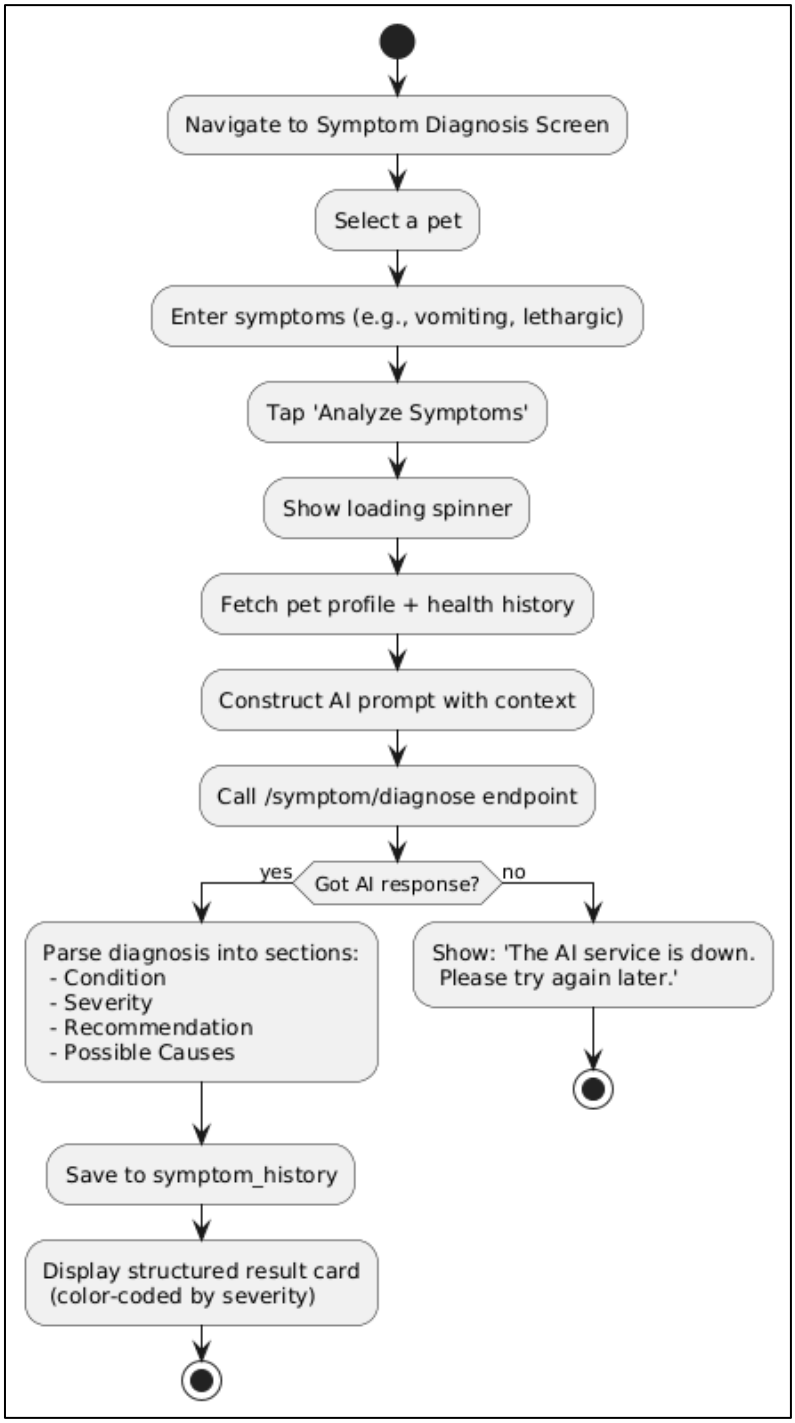


Figure 5.14: Symptom Diagnosis Screen activity diagram

5.6.8 Education Article Activity Diagram

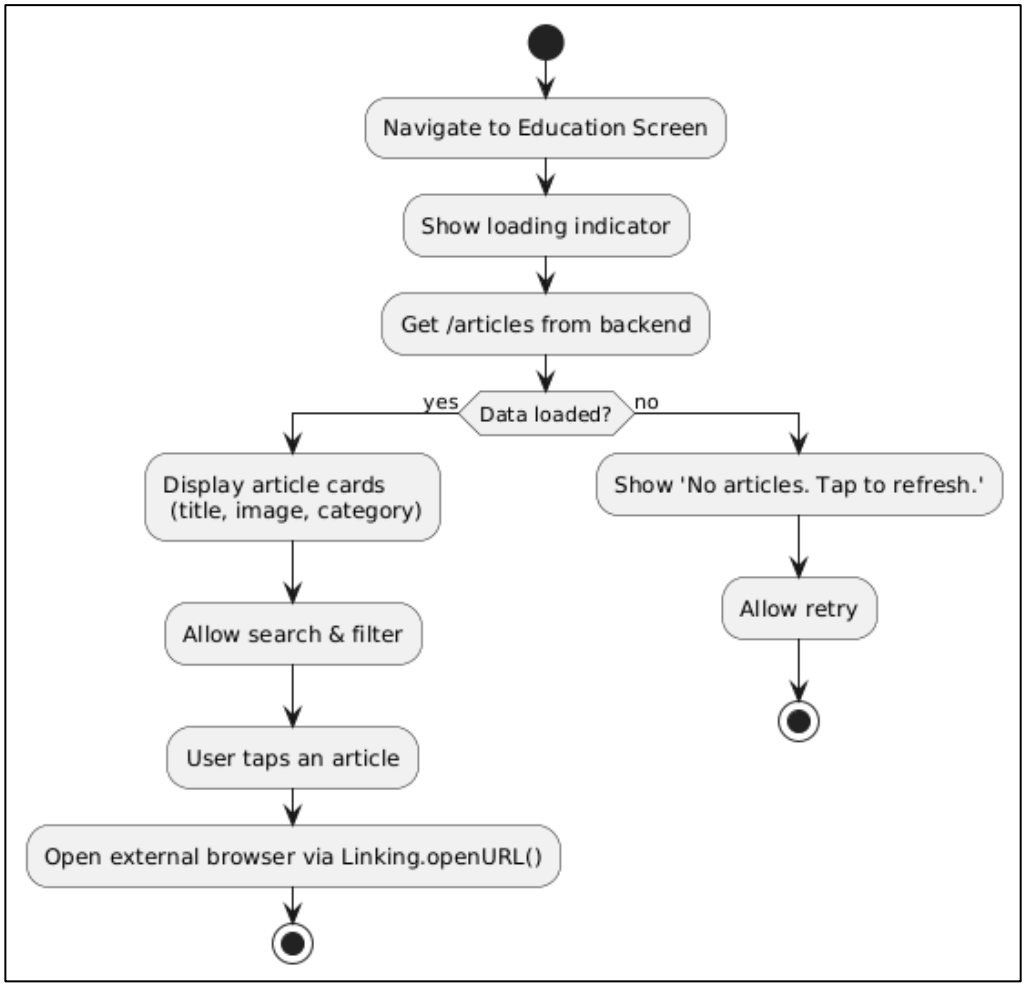


Figure 5.15: Education Screen activity diagram

5.6.9 Feedback Activity Diagram

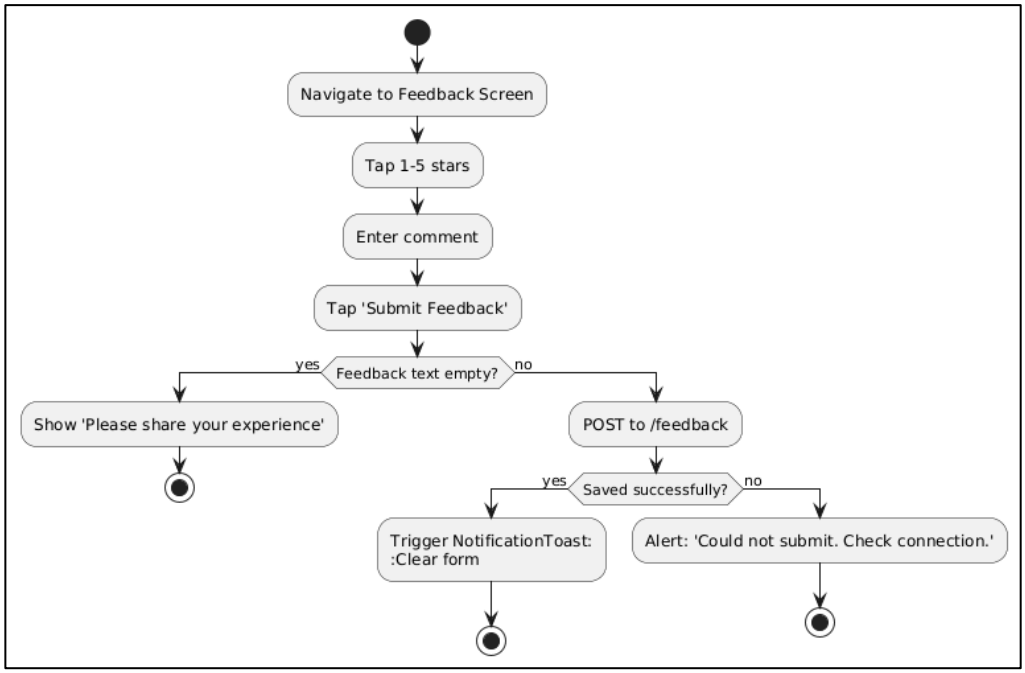


Figure 5.16: Feedback Screen activity diagram

5.6.10 Profile Management Activity Diagram

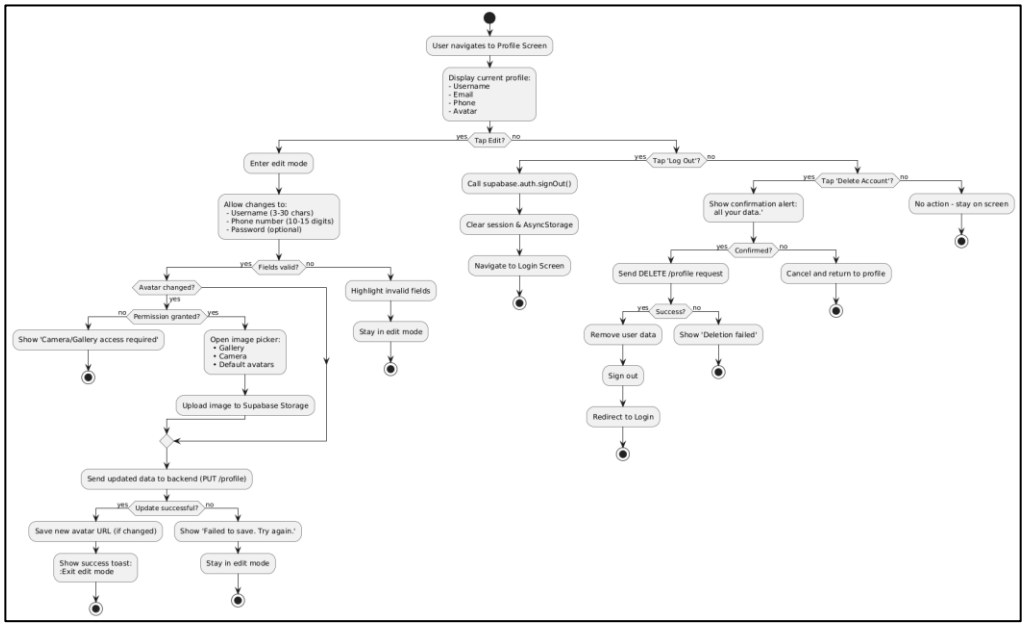
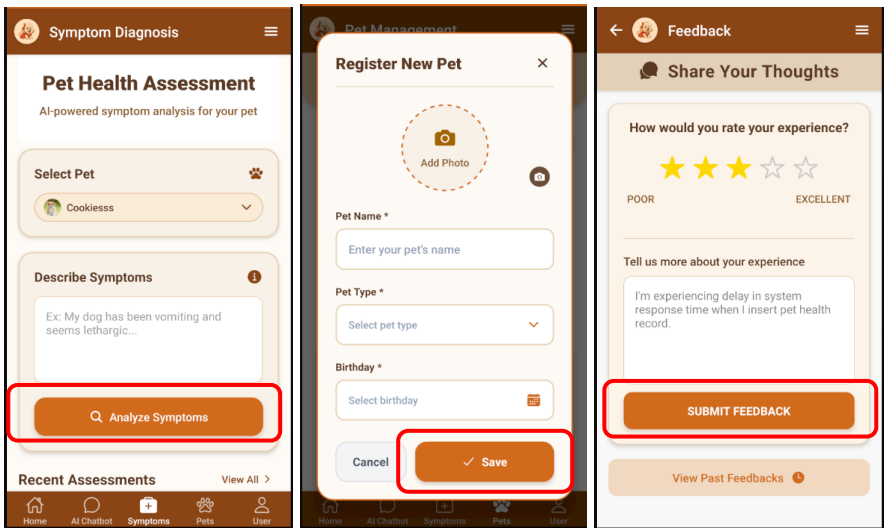


Figure 5.17: Profile Management Screen activity diagram

5.7 Mobile Application Design Principles

The design of PawHub was guided by established human-computer interaction (HCI) principles, particularly Shneiderman's Eight Golden Rules of Interface Design, to ensure an intuitive, efficient, and user-centered experience for pet owners managing their animals' health. These principles were systematically applied throughout the app's core workflows from selecting a pet and entering symptoms to viewing AI-generated diagnoses and managing health records, resulting in a cohesive, trustworthy, and professional mobile interface.

i. Strive for Consistency

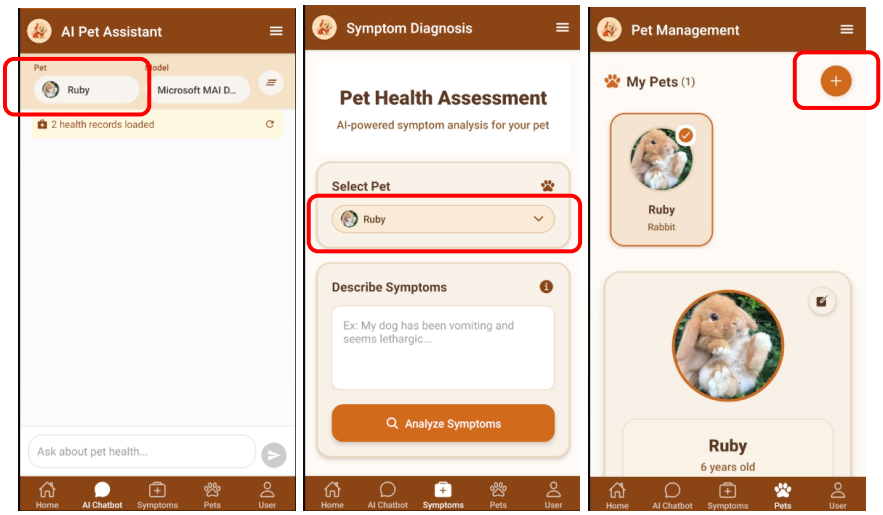


(a) (b) (c)

Figure 5.18(a)(b)(c): Strive for Consistency

PawHub maintained strict visual and functional consistency across all screens. The primary action button color, input field styling, typography hierarchy, icon usage, and modal layouts were uniform whether the user is adding a pet, submitting feedback, or viewing a diagnosis report. This consistency reduced cognitive load, allowing users to navigate confidently without relearning interactions. For example, the "Analyze Symptoms" button appears identically on the Symptom Diagnosis screen as the "save" button does on the Pet Management form to add pet, both use the same rounded corner, shadow, and color scheme, reinforcing familiarity.

ii. Enable Frequent Users to Use Shortcuts



(a) (b) (c)

Figure 5.19(a)(b)(c): Enable Frequent Users to Use Shortcuts

To support power users who interacts with the app daily, PawHub incorporated contextual shortcuts. When a user has only one pet, it is auto-selected upon opening the Symptom Diagnosis screen or AI Chatbot, eliminating a mandatory selection step. Additionally, the floating "+" button on the Pet Management screen allows instant access to new pet profile creation without navigating through menus.

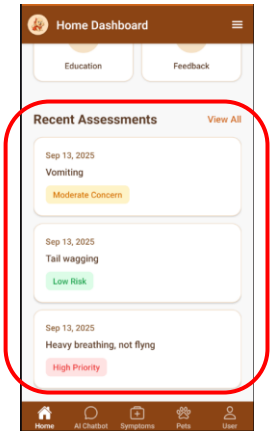
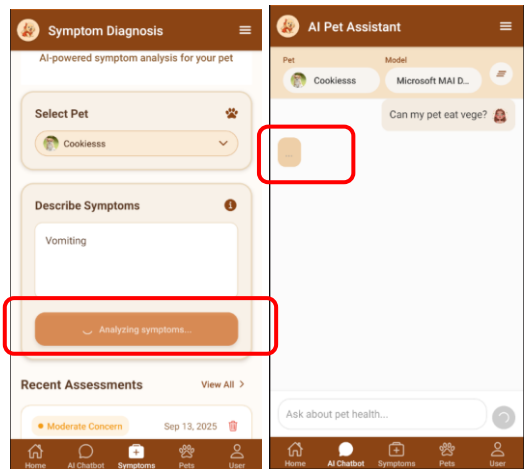


Figure 5.20: Enable Frequent Users to Use Shortcuts - Prominent Display

Recent assessments were prominently displayed in the home view, enabling quick review without drilling into full history, accelerating common tasks while preserving discoverability for new users.

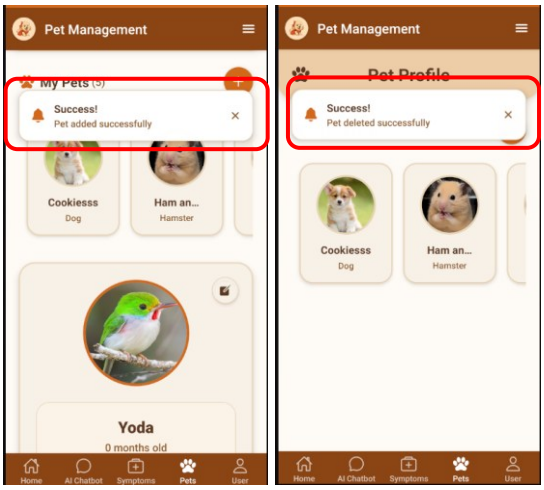
iii. Offer Informative Feedback



(a) (b)

Figure 5.21(a)(b): Offer Informative Feedback - loading spinner

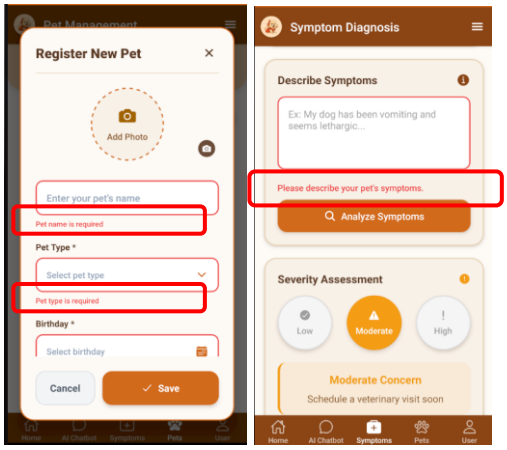
Every user action triggered immediate, meaningful feedback. When submitting symptoms or feedback, a loading spinner with the text appeared instantly, signaling system activity. When a user interacted with the AI Chatbot, the 3 dots were displayed to show the AI is typing.



(a) (b)

Figure 5.22(a)(b): Offer Informative Feedback - Success Toast

Upon successful creation or deletion of a pet, health record or assesment, a subtle toast notification slided in confirming success without interruption.

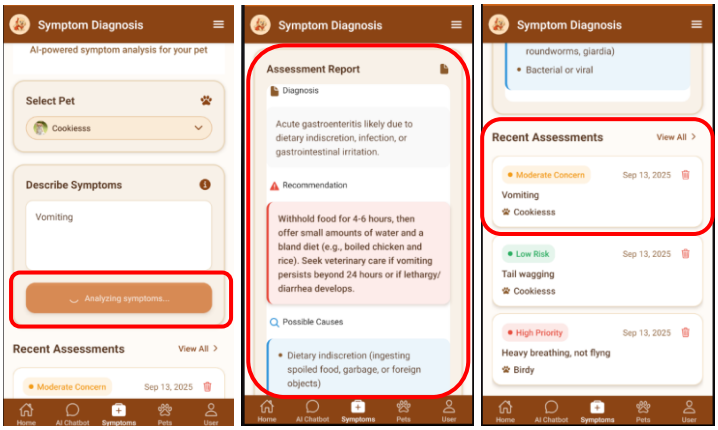


(a) (b)

Figure 5.23(a)(b): Offer Informative Feedback - Input Validation

Input validation was handled inline, empty fields turned red with descriptive error messages below them, ensuring users understand what went wrong without confusion or frustration. These layered feedback mechanisms kept users informed at every stage of interaction.

iv. Design Dialogs to Yield Closure

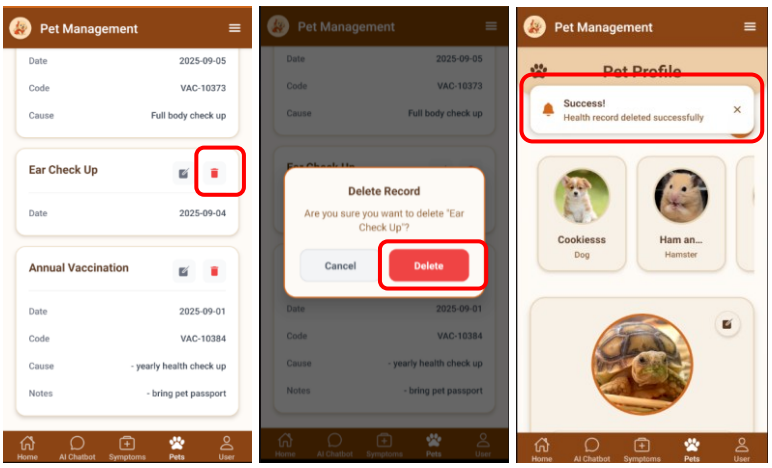


(a) (b) (c)

Figure 5.24(a)(b)(c): Design Dialogs to Yield Closure - Symptom Diagnosis Flow

Each core task followed a clear, closed-loop sequence. In the Symptom Diagnosis flow, users begin by entering symptoms, tap “Analyze”, observed the AI processing state, receive a structured report, and found the result automatically saved in their history. No step feels incomplete, the

final diagnosis card and its presence in the “Recent Assessments” list provided tangible closure.

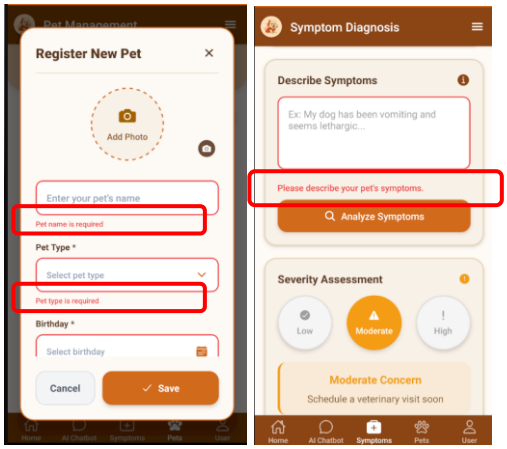


(a) (b) (c)

Figure 5.25(a)(b)(c): Design Dialogs to Yield Closure - Delete Heatlh Record

Similarly, deleting a health record required a confirmation modal followed by immediate visual removal from the list, ensuring users felt confident their action was fully processed.

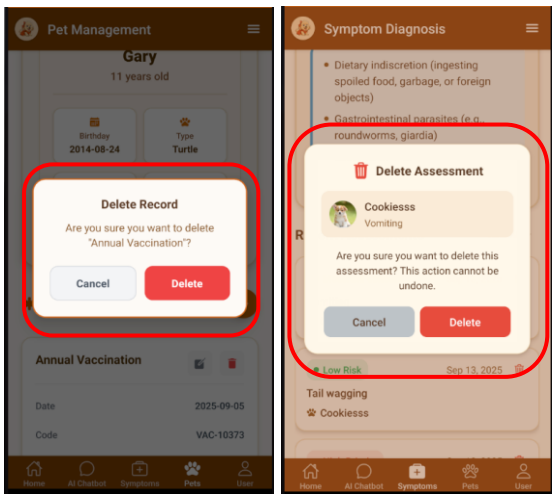
v. Offer Simple Error Handling



(a) (b)

Figure 5.26(a)(b): Offer Simple Error Handling - Input Validation

Errors were prevented and resolved gracefully. Before submitting a request, the app validated that sections are not blank, if they are, the input field turned red with a clear message. This inline approach prevents form submission failures entirely.



(a) (b)

Figure 5.27(a)(b): Offer Simple Error Handling - For irreversible actions

For irreversible actions like deleting a user profile, pet or assessment, a confirmation modal appeared with a destructive button labeled “Delete,” requiring explicit intent. This dual-layered strategy, prevention via validation and recovery via confirmation minimizes user anxiety and data loss.

vi. Permit Easy Reversal of Actions

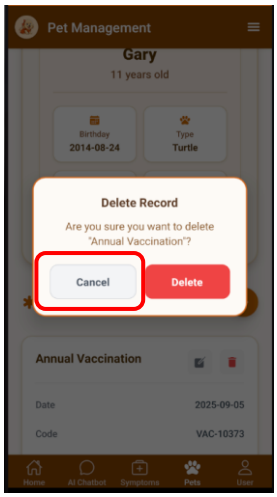
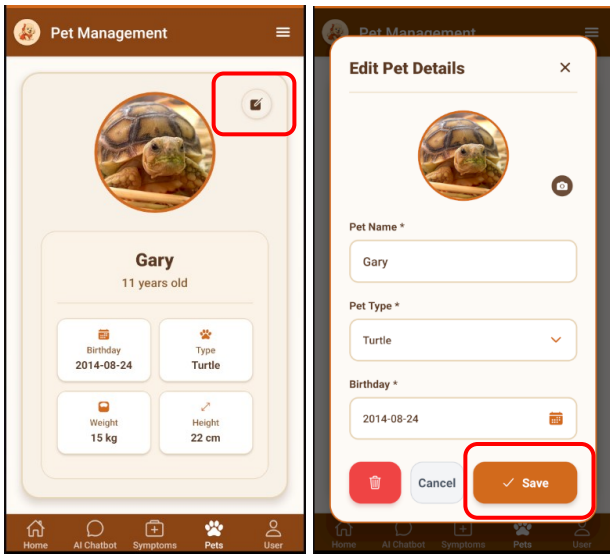


Figure 5.28: Permit Easy Reversal of Actions – Cancel

PawHub prioritizes reversibility to encourage exploration. Deleting any item whether a pet, health record, or symptom assessment triggered a dedicated confirmation modal with clearly labeled “Cancel” and “Delete” buttons. There were no “undo” after deletion, but because all data were stored in the cloud and visible in the History section, users can always re-add information if needed.

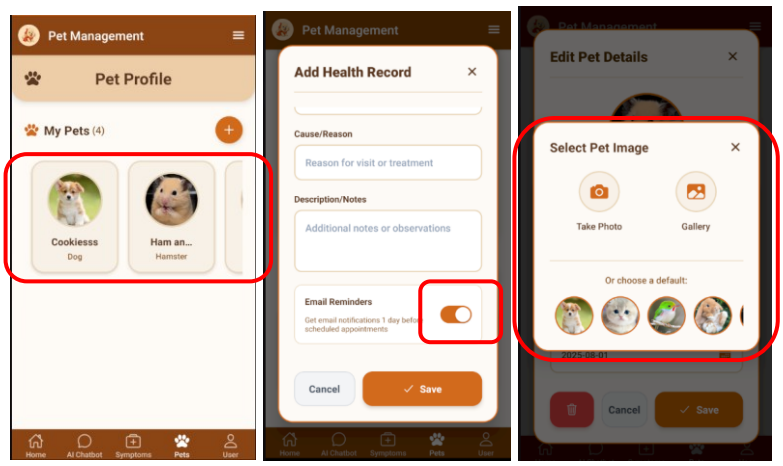


(a) (b)

Figure 5.29: Permit Easy Reversal of Actions - non-destructive until explicitly saved

Furthermore, edits to pets or records are non-destructive until explicitly saved, allowing users to cancel changes mid-flow without consequence.

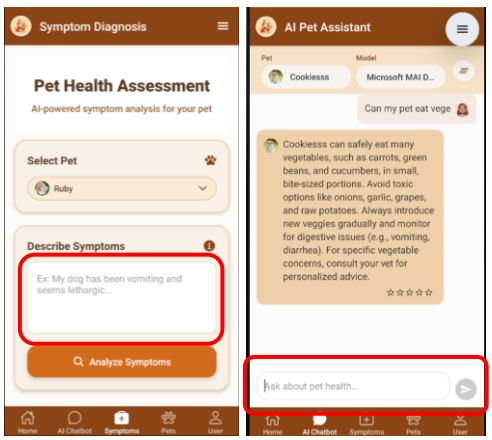
vii. Support Internal Locus of Control



(a) (b) (c)

Figure 5.30(a)(b)(c): Support Internal Locus of Control – Control to decide

Users remained firmly in control of their workflow. They chose which pet to analyze, decide whether to enable email reminders for upcoming appointments, select images from camera or gallery (or use a default), and toggle between viewing recent or full history.



(a) (b)

Figure 5.31(a)(b): Support Internal Locus of Control – AI Never Pre-fills

No automation overrided user intent. For instance, the AI never pre-fills symptoms. Even when the AI service is slow, the app waits patiently rather than auto-retrying or redirecting, preserving user agency and trust.

viii. Reduce Short-Term Memory Load

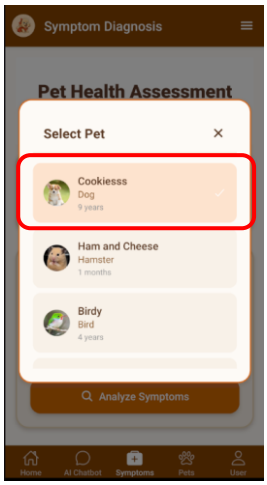


Figure 5.32: Reduce Short-Term Memory Load – Pet Information Populated

PawHub minimized the need for users to remember information across screens. When diagnosing symptoms, the selected pet's name, breed, age, and weight are auto-populated from their profile eliminating manual entry.

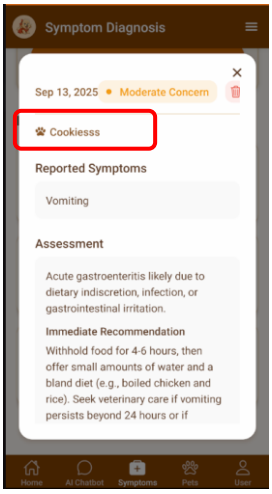
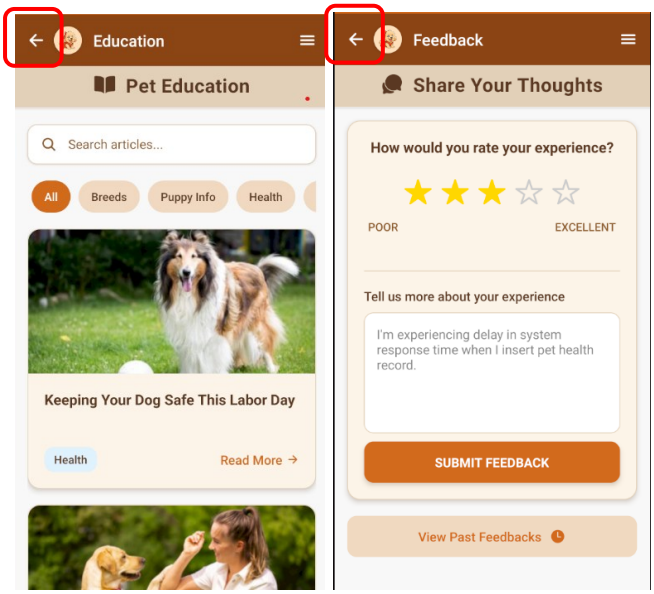


Figure 5.33: Reduce Short-Term Memory Load – Pet Preview

In the History section, each assessment card displayed a concise preview, pet name, date, severity badge, and a truncated symptom summary enough to recall context without opening the full report.



(a) (b)

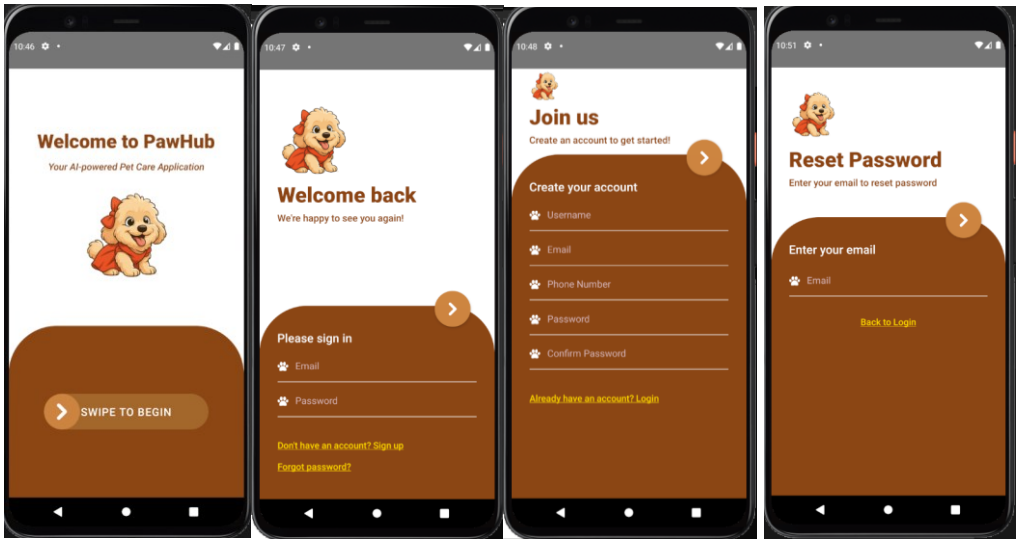
Figure 5.34(a)(b): Reduce Short-Term Memory Load – Consistent Navigation

Navigation were also consistent, back buttons always returned to the previous screen, and breadcrumbs were implied through hierarchical structure rather than explicit labels.

5.8 User Interface (UI) Design

The user interface (UI) of PawHub were designed to be intuitive, accessible, and visually appealing, ensuring a seamless experience for pet owners of all technical levels. The UI supported all core functionalities of the system, including authentication, pet management, AI interaction, and health tracking. Screens were designed with consistent navigation, clear typography, and pet-friendly aesthetics using warm colors and icons.

5.8.1 User Authentication Screens

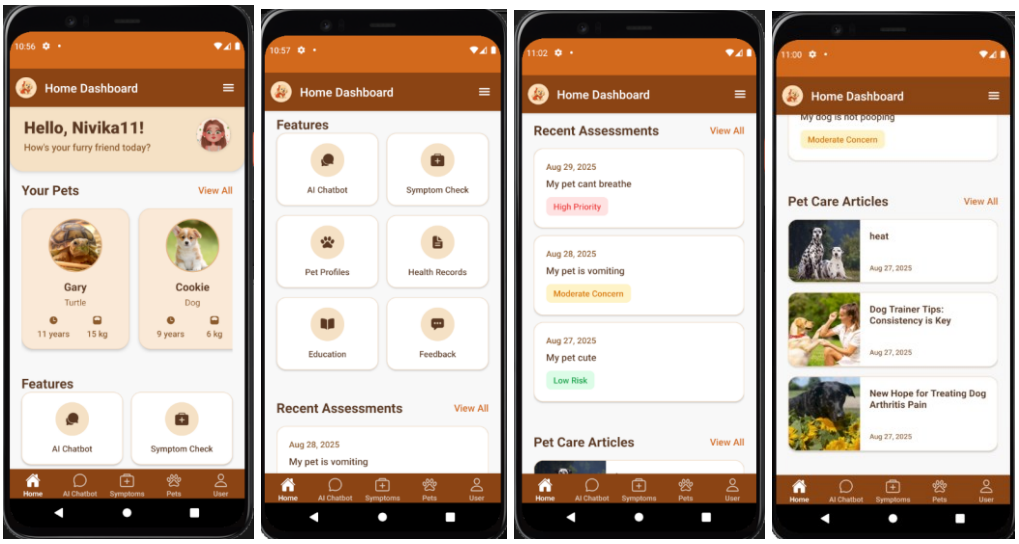


(a) (b) (c) (d)

Figure 5.35(a)(b)(c)(d): User Authentication Screens UI

When a user opened the PawHub app, they were first presented with the Welcome Screen, featuring a swipeable onboarding experience. After swiping through the introduction, it took the user to the Login Screen. The login interface provided two primary options, log in to an existing account or register a new account. For users who have forgotten their password, a “Forgot Password?” link redirects them to the Reset Password Screen from the Login Screen. Upon entering their email, a reset link was sent via Supabase Auth, allowing secure password recovery. All user inputs were validated on the client side (e.g., email format, password strength) before being securely transmitted to the backend and stored in Supabase. This ensured data integrity and a smooth, user-friendly authentication experience.

5.8.2 Homescreen



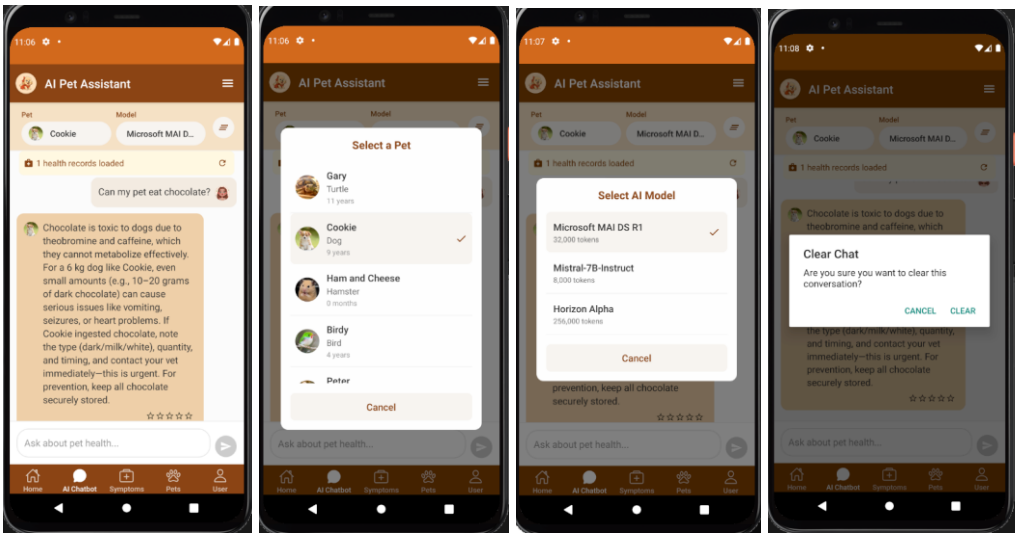
(a) (b) (c) (d)

Figure 5.36(a)(b)(c)(d): Home Screen UI

The homescreen served as the central dashboard of the PawHub application, providing users with a personalized overview of their pets, recent AI assessments, and quick access to all key features. Designed with a warm, pet-friendly aesthetic using soft browns, creams, and card-based layouts, the interface ensured intuitive navigation and a welcoming user experience.

Upon login, the user was greeted with a personalized welcome message (“Hello, [Username]!”) and a profile button that links to the User Profile screen. The main content was organized into clear sections such as Your Pets, displays a horizontal scrollable list of pet cards, each showing the pet’s full detailed information. Tapping a card navigates to the Pet Profile Management screen for detailed management. Features displayed a 2-column layout of the six core features each with an icon and label for easy recognition. Recent Assessments showed up to three recent AI symptom diagnoses with the date, symptoms, and a color-coded severity badge (Low Risk, Moderate Concern, High Priority) based on the diagnosis. Pet Care Articles displayed the latest educational articles from AKC with thumbnails, titles, and publication dates. Tapping an article opened it in the Education screen.

5.8.3 AI Chatbot Screen



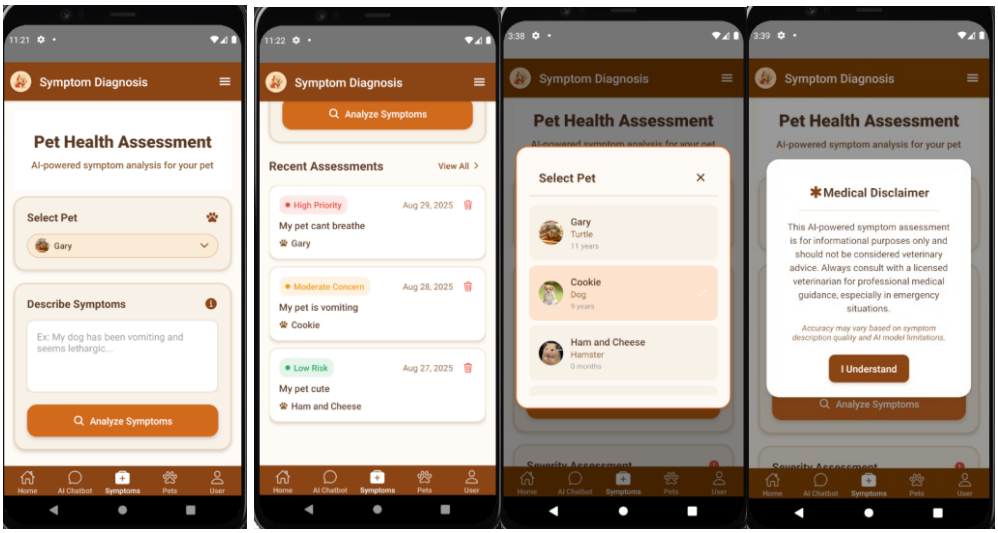
(a) (b) (c) (d)

Figure 5.37(a)(b)(c)(d): AI Chatbot Screen UI

The AI Chatbot Screen was the central hub for interactive pet care support in PawHub, offering a clean, intuitive interface where users could ask questions about pet health, behavior, nutrition, and training. Responses were powered by OpenRouter AI and personalized using the selected pet's profile, including type, age, weight, and medical history. Users could choose between AI models such as Microsoft MAI DS R1 (primary), Mistral-7B-Instruct, and Horizon Alpha enabling model fallback for reliability.

User messages appeared on the right, AI responses on the left with a pet-themed icon. A typing indicator showed when the AI is responding, and users can rate messages with 1–5 stars. The input field included a multi-line text box with a disabled send button during loading to prevent duplicates. If the AI fails, a friendly fallback message appeared, prompting users to check their connection and consult a vet if needed. A "Clear Chat" button cleared the conversation, and a health records badge showed how many records were loaded for context. The screen delivered a responsive, secure, and user-friendly experience while clearly emphasized that the app is a supportive tool, not a substitute for professional veterinary care.

5.8.4 Symptom Diagnosis Screen

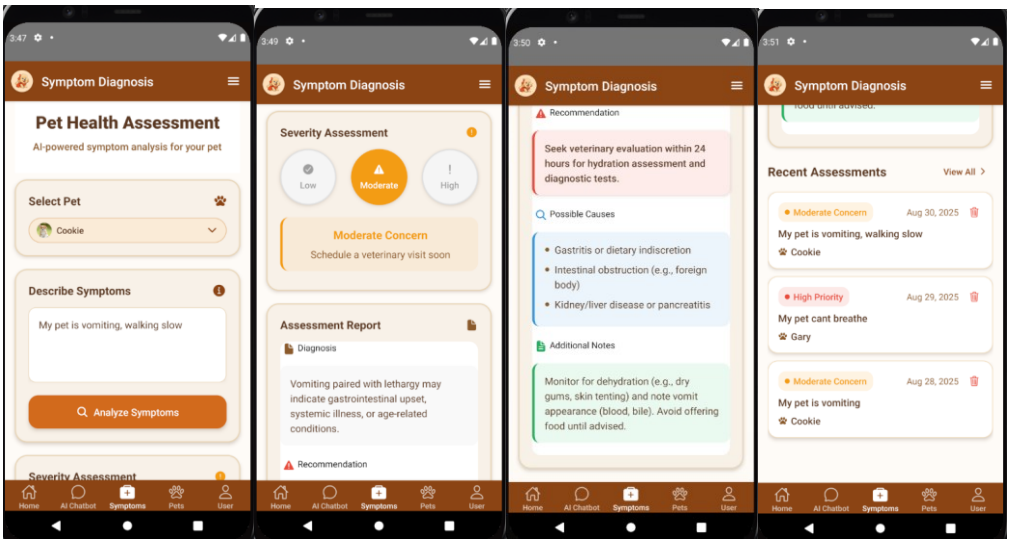


(a) (b) (c) (d)

Figure 5.38(a)(b)(c)(d): Symptom Diagnosis Screen UI

The Symptom Diagnosis Screen provided pet owners with AI-powered health assessments based on user-submitted symptoms. The clean, intuitive interface guided users through three steps, select a pet, describe symptoms, and analyze. A modal allowed pet selection with name, breed, and age displayed for accurate context. Users input symptoms in a multi-line text field, then tap “Analyze Symptoms” to start the AI process, with a loading indicator preventing duplicate submissions.

User input symptoms:

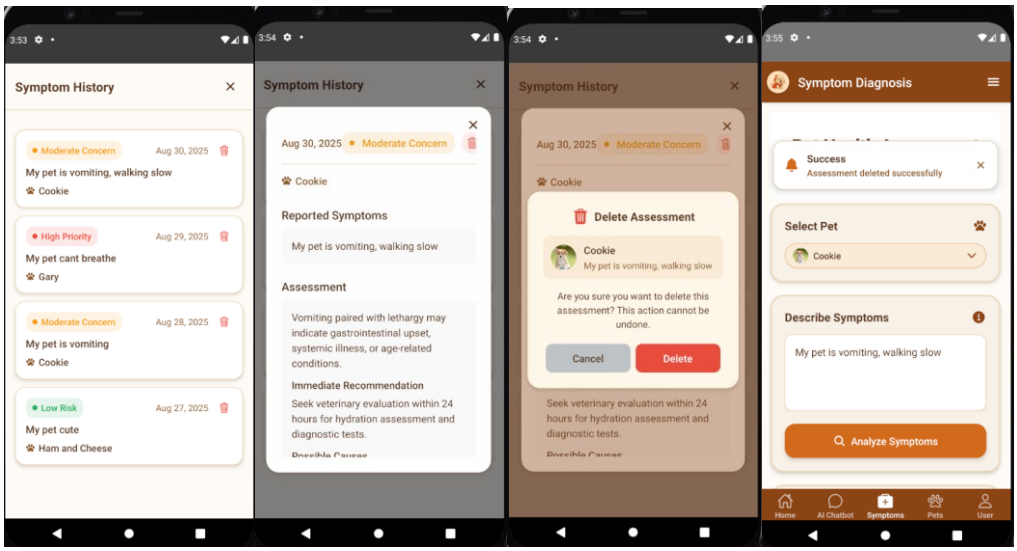


(a) (b) (c) (d)

Figure 5.39(a)(b)(c)(d): Symptom Diagnosis Screen Results UI

Results were presented in a structured format with clear sections: Diagnosis, Recommendation, Possible Causes, and Additional Notes. A color-coded severity badge (green, yellow, red) indicated urgency such as Low Risk, Moderate Concern, or High Priority helping users decide on next steps. A visible disclaimer emphasized that the app supports, but does not replace, professional veterinary care.

Symptom History Modal:

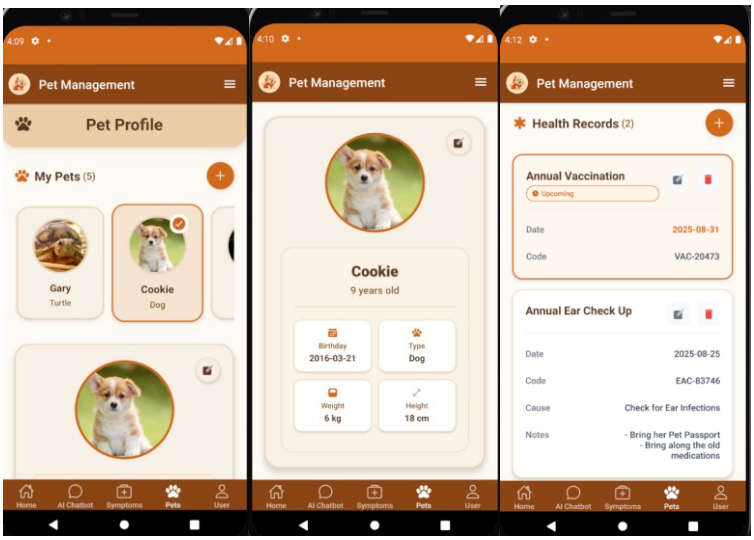


(a) (b) (c) (d)

Figure 5.40(a)(b)(c)(d): Symptom History Modal UI

A "Symptom History" button opened a scrollable modal showing past assessments with date, symptoms, and pet name. Entries could be viewed or deleted, with an empty state shown if no history existed. All data was securely sent to the Node.js backend, enriched with pet details, analyzed via OpenRouter AI, and stored in the symptom_history table. This screen effectively combined AI, personalization, and user-centered design to deliver timely, actionable pet health insights.

5.8.5 Pet Management Screen



(a) (b) (c)

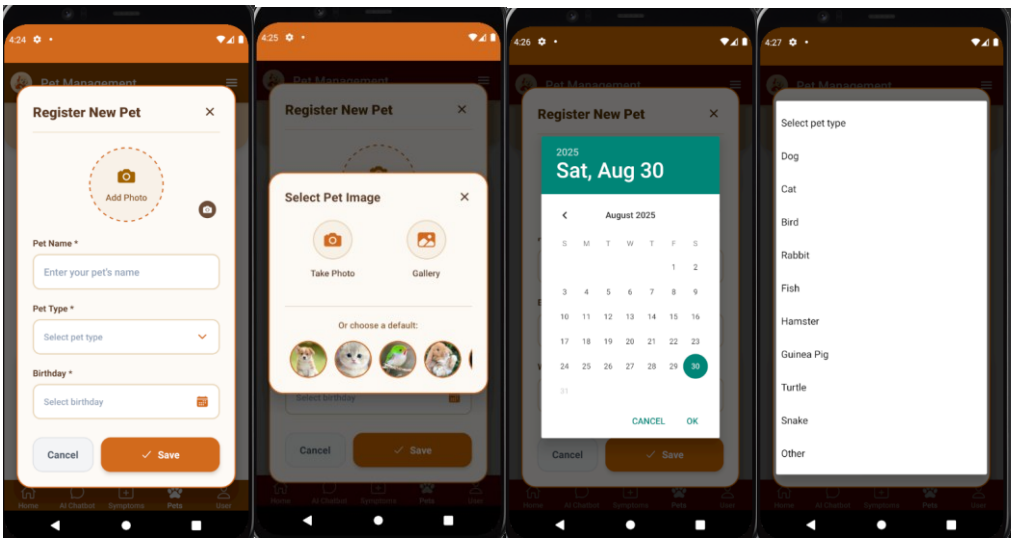
Figure 5.41(a)(b)(c): Pet Management Screen UI

The Pet Management Screen was a central feature of the PawHub application, allowing users to manage their pets' information and health records through a clean, intuitive interface. The screen was divided into two main sections, Pet Profile Management and Health Records Management, enabling users to organize, track, and maintain comprehensive pet health data.

i) CRUD for pet profile management:

On pet profiles, users could carry out complete Create, Read, Update, and Delete (CRUD) actions. To add a new pet, users tap an "+" button to add new pets and were presented with a form to input essential details such as pet name, type, birthday, weight, height, and upload a photo. Form validation ensured accurate data entry, and the new pet was saved securely to Supabase via the backend API. Existing pet profiles were displayed in a scrollable list or card layout, and tapping a pet opened the full profile for viewing or editing. Users could update any field and save changes, or delete a pet profile with confirmation. A visual indicator (e.g., pet avatar and age) enhanced recognition and personalization.

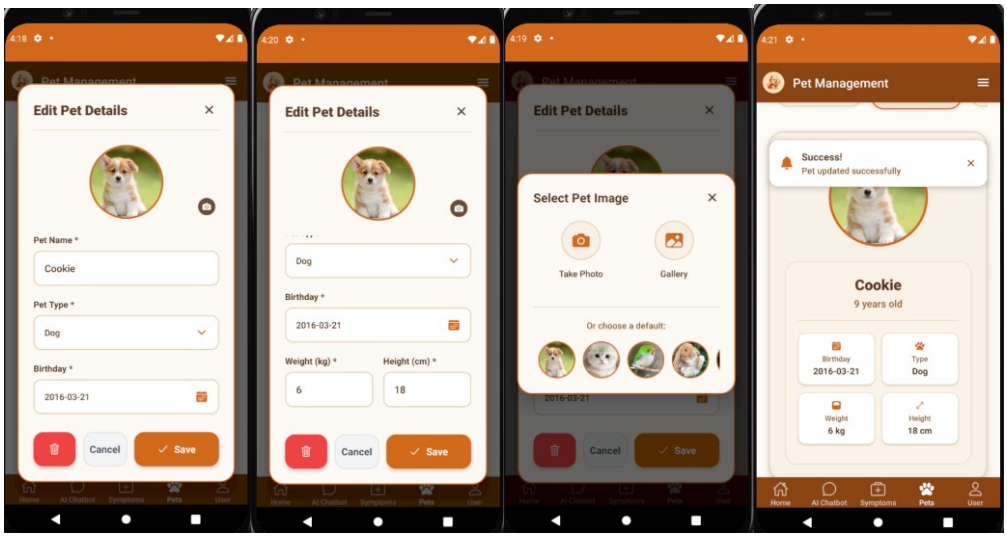
Create:



(a) (b) (c) (d)

Figure 5.42(a)(b)(c)(d): Create Pet Profile Management Screen UI

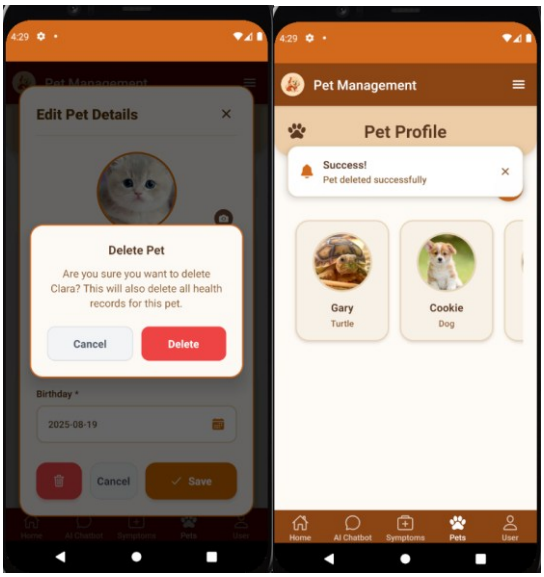
Update:



(a) (b) (c) (d)

Figure 5.43(a)(b)(c)(d): Update Pet Profile Management Screen UI

Delete:

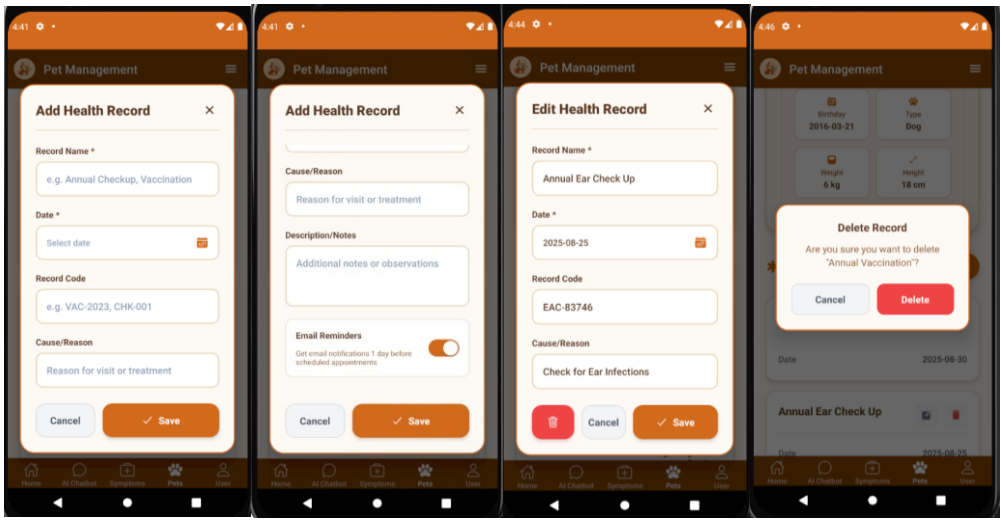


(a) (b)

Figure 5.44(a)(b): Delete Pet Profile Management Screen UI

ii) CRUD for health records of each pet:

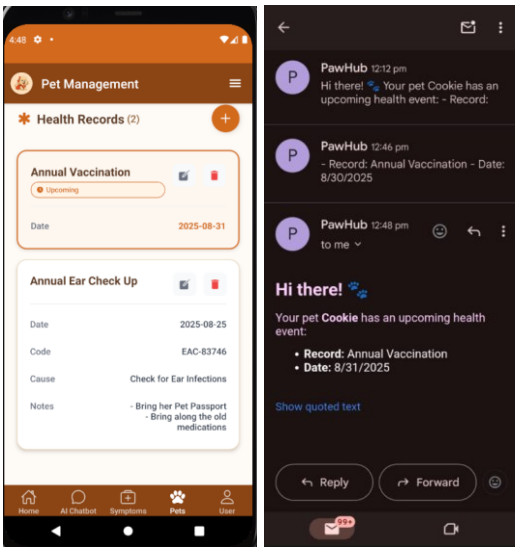
Each pet had an associated Health Records section where users could add, view, edit, or delete medical entries such as vaccinations, check-ups, treatments, or deworming. When adding a record, the form captured the record name, date, cause, description, and code. A toggle allowed users to enable email reminders for future-dated events. Once saved, records were displayed in chronological order with key details visible at a glance. Users could tap any record to view or modify it, or delete it with confirmation.



(a) (b) (c) (d)

Figure 5.45(a)(b)(c)(d): CRUD for Pet Health Record Management Screen UI

Email Reminder for Future Health Events:



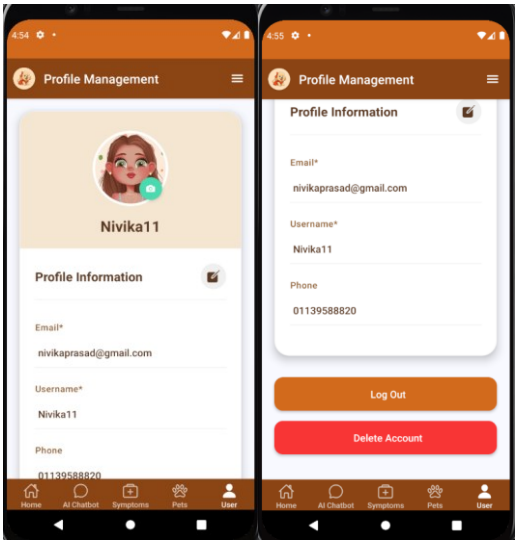
(a) (b)

Figure 5.46(a)(b): Reminder for Pet Health Record Management Screen UI

When a user added a health record with a future date and enabled the notification toggle, the system automatically scheduled an email reminder to be sent one day before the event. This was handled by the backend, which created an entry in the email_reminders table and triggered the Resend email

service daily to check for upcoming reminders. The email included the pet's name, event type, and date, prompting the owner to prepare. This proactive notification system helped prevent missed appointments and supported preventive pet care.

5.8.6 Profile Management Screen

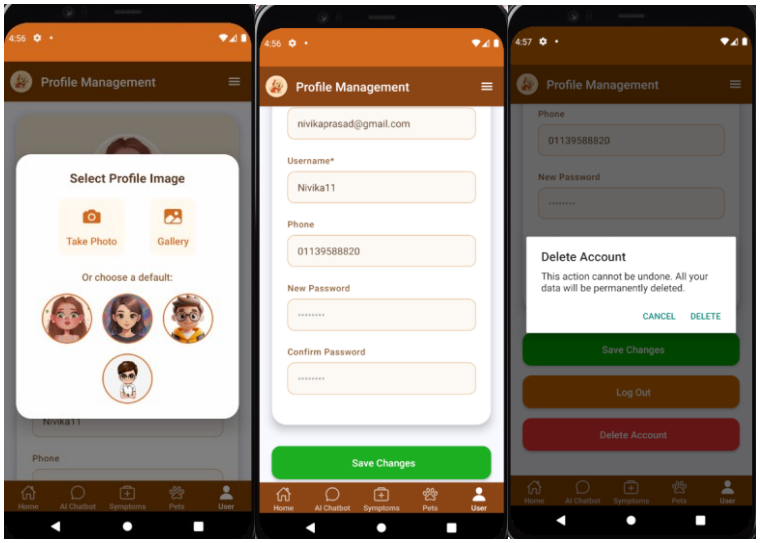


(a) (b)

Figure 5.47(a)(b): Profile Management Screen UI

The Profile Management Screen allowed users to personalize their account settings, update personal information, manage their profile picture, and control their account security. The interface was designed with a clean, card-based layout that separated the avatar section from the profile information, providing a clear and organized user experience.

CRUD in Profile Management:



(a) (b) (c)

Figure 5.48(a)(b)(c): CRUD in Profile Management Screen UI

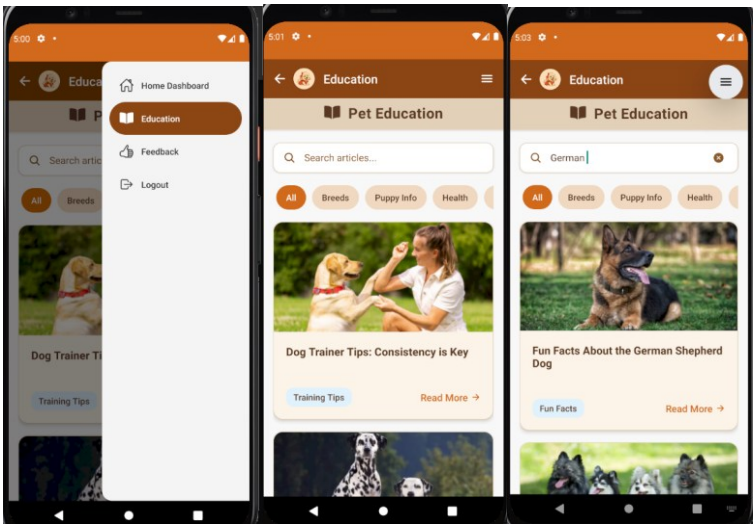
The Profile Management Screen allowed users to update their profile picture and personal information securely. By tapping the avatar, users could choose to take a photo, select an image from their gallery, or pick from predefined default avatars. The selected image was uploaded to Supabase Storage, and the URL was saved in the profiles table. A loading indicator and error handling ensured a smooth experience.

Below, users could edit their email, username, phone number, and password in a form with real-time validation for format and strength. Changes were saved via secure API calls to the Node.js backend, with immediate feedback on success.

Users also had the option to delete their account, which triggered a confirmation alert and, when confirmed, performed a soft delete via the backend API, removing the user's data and signing them out.

Additionally, the "Log Out" button allowed users to securely end their session. The screen used secure authentication flows with JWT tokens stored in AsyncStorage, and all API requests were protected with Bearer token authentication.

5.8.7 Education Screen

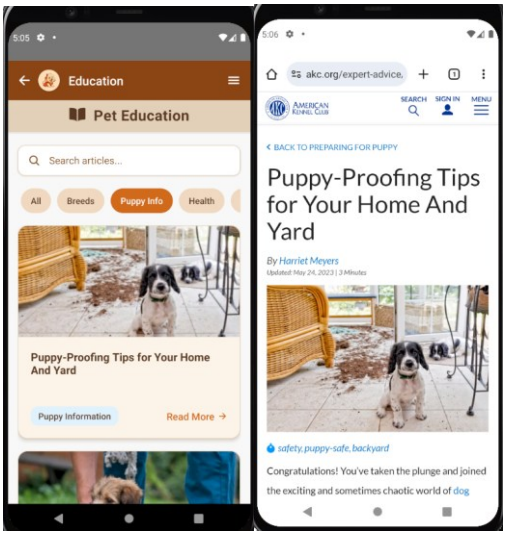


(a) (b) (c)

Figure 5.49(a)(b)(c): Education Screen UI

The Education Screen in PawHub provided users with access to reliable, up-to-date pet care information sourced from trusted authorities such as the American Kennel Club (AKC). This screen featured a scrollable list of curated articles organized by category, including Dog Breeds, Health, Training, Nutrition, and Fun Facts. Each article was displayed in a clean card layout with a thumbnail image, title, category, and publication date, offering users a quick preview before reading.

Click into an article:



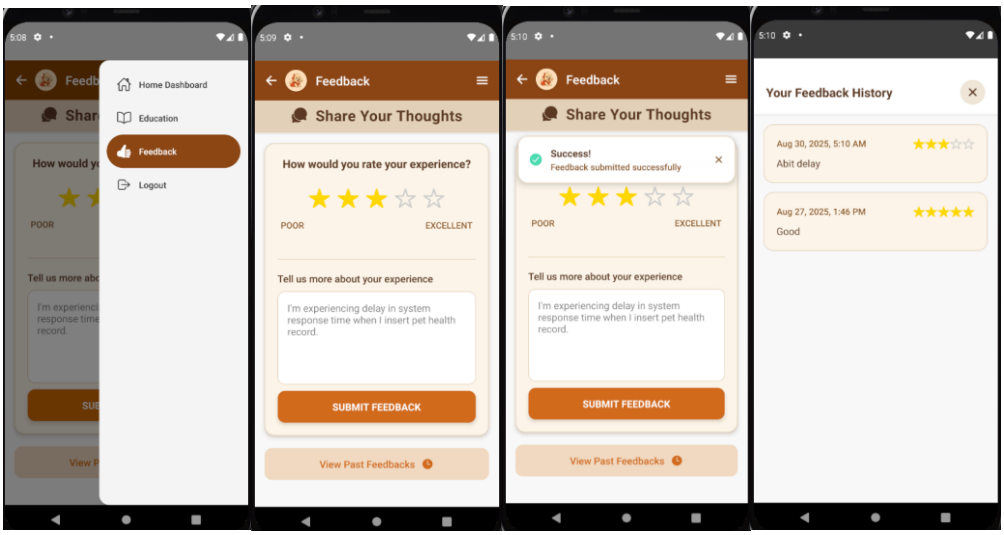
(a) (b)

Figure 5.50(a)(b): Article in Education Screen UI

When a user tapped on an article card, they were redirected to the original webpage in their device's default browser, ensuring they received the most current and accurate information directly from the source. This approach maintained content integrity and avoided copyright issues, while still providing seamless access to expert-backed knowledge.

The content displayed in the app was updated through a web scraper that runs in the terminal by the developer, fetching new articles from AKC's website and storing them in the Supabase database. This ensured the Education Screen remains fresh and relevant. For future enhancements, Github Actions could be implemented to automate the webscraper weekly without requiring manual updates.

5.8.8 Feedback Screen



(a) (b) (c) (d)

Figure 5.51(a)(b)(c)(d): Feedback Screen UI

The Feedback Screen allowed users to share their experience with the PawHub application through a simple, intuitive interface. The screen features a clean, card-based layout with a rating system and text input for detailed feedback. Users beginned by selecting a star rating from 1 to 5, with visual labels indicating "POOR" to "EXCELLENT" to guide their selection. Below, a multi-line text field enabled users to describe their experience, including suggestions or issues encountered.

Once submitted, the feedback was securely sent to the backend via an authenticated API call to the Node.js server, where it is stored in the feedback table in Supabase. A success notification toast appeared to confirm submission. Users could also view their past feedback entries by tapping the "View Past Feedbacks" button, which opened a modal displaying a scrollable list of previous submissions, including the date, rating, and feedback text. If no feedback were submitted, an empty state was shown with a descriptive message. The screen included real-time validation to ensure the feedback text was not empty.

5.9 Conclusion

This chapter showed an overview of the system design of PawHub, an AI-based pet care application designed to support pet owners with virtual assistance, symptom diagnosis, and health management. The system architecture demonstrated a secure, scalable, and well-structured integration of React Native, Node.js, Supabase, OpenRouter, and Resend, ensuring robust communication between frontend and backend components.

The database design was detailed through a complete data dictionary and entity relationships, highlighting data integrity, security through Row Level Security (RLS), and efficient organization of user, pet, health, and AI interaction data. The API endpoints were systematically outlined, showcasing a well-organized RESTful interface that supports all core functionalities with proper authentication and rate limiting.

Additionally, the Data Flow Diagrams (DFD) and Activity Diagram strengthened the application's logical flow and flexibility by clearly visualizing the data flow from user input to AI processing and storage. The user interface design ensured usability, accessibility, and a seamless experience across all screens.

Overall, the system design of PawHub reflected a professional, user-centered approach that successfully integrated modern technologies to deliver a reliable and intelligent pet care solution. This solid foundation enables future enhancements such as push notifications, advanced analytics, or integration with wearable pet devices, positioning PawHub as a scalable and impactful mobile health application.

CHAPTER 6

IMPLEMENTATION

6.1 Introduction

This chapter showed the implementation and integration of PawHub, an AI-powered mobile application designed to assist pet owners with virtual pet care, symptom diagnosis, and health management. The system was developed using a modular, full-stack approach that integrates frontend, backend, database, artificial intelligence, and automated services. Each component was implemented with a strong emphasis on security, usability, and scalability, ensuring a robust and user-friendly experience. The development followed an Agile methodology, enabling iterative refinement and continuous integration.

6.2 Frontend Implementation

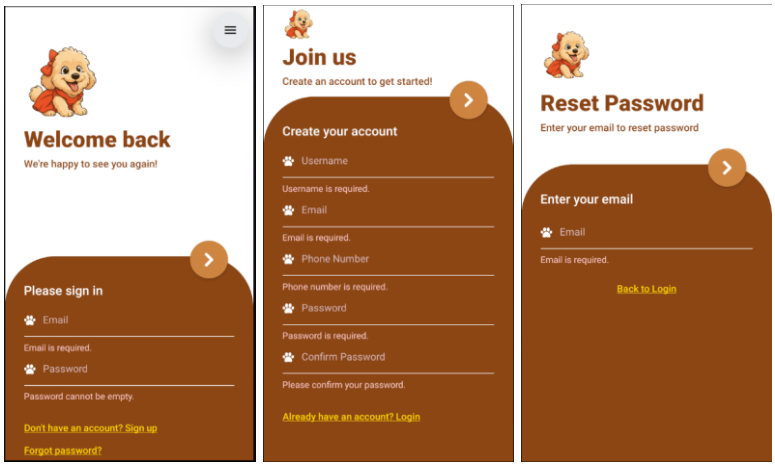
The frontend of PawHub was developed using React Native and tested in Android Studio Emulator, providing a robust, cross-platform foundation that ensured seamless performance on both Android and iOS devices. The choice of React Native aligned with the project's goal of delivering a high-quality, native-like user experience while maintaining a single codebase, a critical advantage for a solo developer working under time and resource constraints.

The user interface was designed in Figma during the prototyping phase, ensuring a cohesive, pet-friendly aesthetic characterized by warm earthy tones (soft browns, creams, and oranges), rounded card layouts, gradient accents, and intuitive navigation. This design language was consistently implemented across all screens to promote familiarity, reduce cognitive load, and create an emotionally engaging experience for pet owners.

6.2.1 Authentication Module

This module handled user onboarding and secure access to the PawHub application. It included the Welcome, Login, Register, and Forgot Password screens, providing a seamless and secure entry point for new and returning users. The onboarding process beginned with a swipeable welcome screen

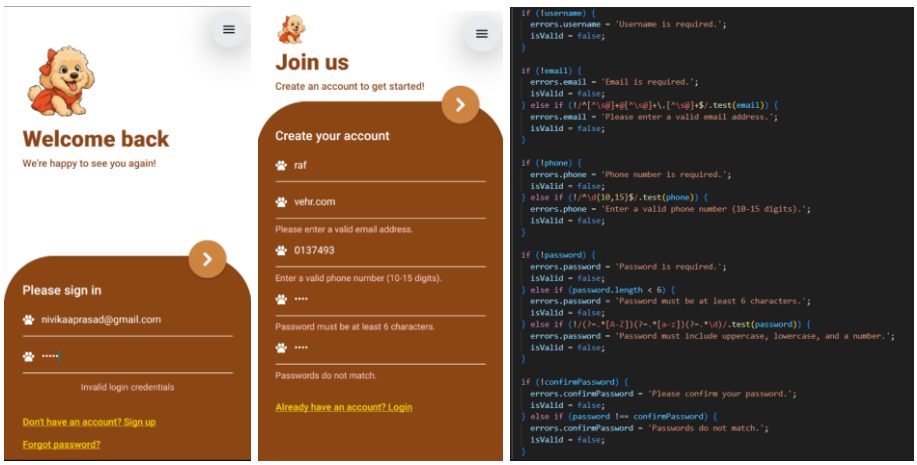
featuring animated transitions that introduce key app features, allowing users to easily navigate to the login or registration interface.



(a) (b) (c)

Figure 6.1(a)(b)(c): Authentication Screen Required Input Validations

The Login, Register, and Forgot Password screens implemented comprehensive input validation to ensure that all required fields were properly filled in before submission to ensure data integrity and security.



(a) (b) (c)

Figure 6.2(a)(b)(c): Authentication Screen Input Validations

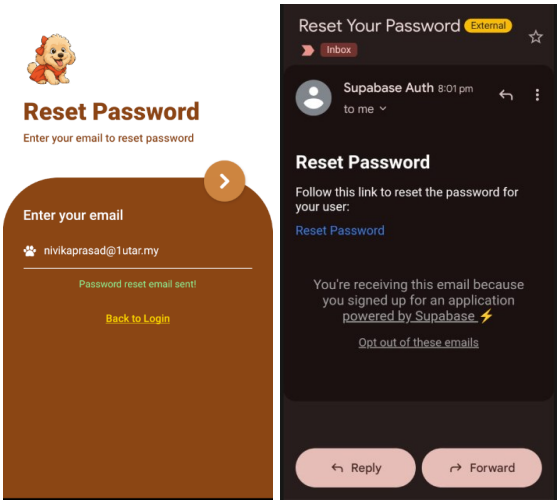
The input validation on the Login, Register, and Forgot Password screens ensured that essential fields like username, email, and phone number were correctly filled before submission. Emails must follow a valid format,

phone numbers must be 10–15 digits, and passwords required a minimum of six characters with uppercase, lowercase, and numeric characters for stronger security. A confirm password checked and prevented mismatches, while clear error messages guided users to fix mistakes. This approach maintained data integrity, enhanced account security, and improved the overall user experience. All user inputs were sanitized and validated on the client side to prevent injection attacks and ensure secure data transmission.

```
// Save token to storage
await AsyncStorage.setItem('accessToken', session.access_token);
```

Figure 6.3: JWT token for session management

Upon successful registration or login, the system leveraged Supabase Auth for secure authentication, generating a JWT token that was stored in AsyncStorage for session persistence.



(a) (b)

Figure 6.4(a)(b): Forgot Password Account Recovery Email

For account recovery, the Forgot Password functionality allowed users to enter their email and receive a secure reset link via email, powered by Supabase's built-in password recovery system. This module established a secure foundation for the entire app, ensuring that only authenticated users

could access personalized features while maintaining a user-friendly onboarding experience.

6.2.2 AI Chatbot Module

This module served as the interactive core of the PawHub application, allowing users to consult a virtual assistant driven by AI in real time for advice on pet care. The interface was designed to mimic a familiar messaging app, ensuring intuitive navigation and a seamless user experience.

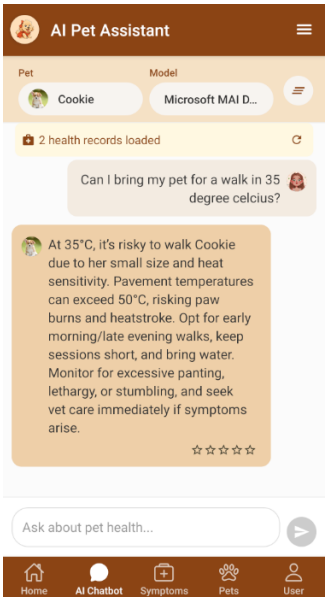


Figure 6.5: AI Chatbot Interface

The screen features a clean, scrollable chat window where messages were displayed in bubbles, AI replies to the left are soft brown with a pet-themed logo, while user messages were displayed towards the right in a light beige tone. A typing indicator with animated dots appeared when the AI was generating a response, providing visual feedback and improving perceived responsiveness.

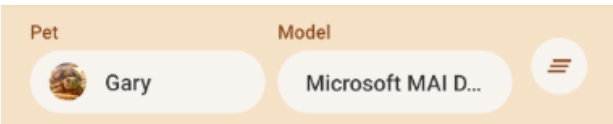
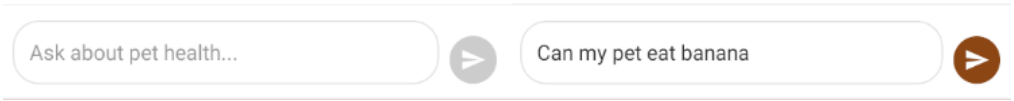


Figure 6.6: Selector buttons to select pet and AI model

At the top of the screen, two selector buttons allowed users to choose a pet and select an AI model (Microsoft MAI DS R1, Mistral-7B-Instruct, or Horizon Alpha). This personalization ensured that AI responses were tailored to the selected pet's profile, including breed, age, weight, and medical history. Tapping the pet selector opened a modal listing all pets with their photos, names, breeds, and calculated ages. The model selector displayed available AI models along with their context window sizes (e.g., 32K, 8K, 256K tokens), empowering users to choose based on performance, speed, or depth of response.



(a) (b)

Figure 6.7: AI multi-line text field enabled and disabled

Users input queries into a multi-line text field at the bottom, with a send button that disabled during loading to prevent duplicate submissions. If the AI failed to respond due to connectivity or API issues, a friendly fallback message was shown, advising users to check their internet connection and consult a veterinarian if needed.

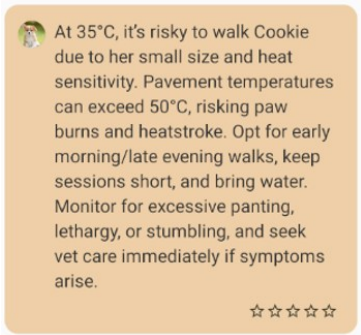


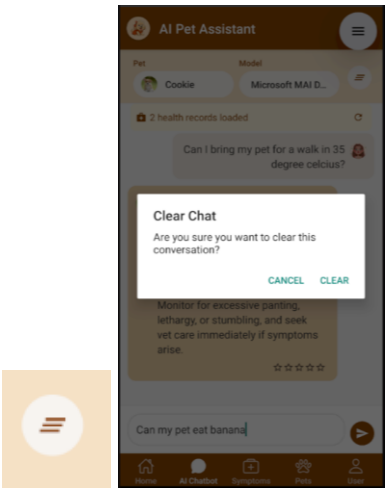
Figure 6.8: AI message rating system

A key feature was the message rating system, each AI response could be rated with 1–5 stars, allowing users to provide feedback directly within the chat. This data was sent to the backend and stored in the chat_history table for future model improvement.

```
// === 💬 Load Chat History ===
const loadChatHistory = useCallback(async () => {
  try {
    const data = await apiCall(`/chat/history?session_id=${sessionId.current}`);
    const formatted = data.map((msg) => ({
      id: msg.id,
      text: msg.message,
      isUser: msg.is_user,
      sessionId: sessionId.current,
      rating: msg.rating,
      created_at: msg.created_at,
    }));
    setMessages(formatted);
  } catch (error) {
    console.error('Failed to load chat history:', error);
    setMessages([]);
  }
}, []);
```

Figure 6.9: AI Chat History session ID

The chat maintained session continuity by grouping messages under a unique session ID. When a user switched pets, the conversation resets automatically to ensure context accuracy.



(a) (b)

Figure 6.10(a)(b): Clear Chat button and trigger

The “Clear Chat” button at the top allowed user to clear their existing chat. When the button was pressed, it triggered a confirmation alert to prevent accidental deletion of the chat.

Figure 6.11: Health Records Loaded Badge

Additionally, a health records badge appeared when records were loaded, informing users that the AI had access to relevant medical history for

more accurate advice of the selected pet. An error banner displayed non-intrusive alerts if data failed to load.

```
const token = await AsyncStorage.getItem('accessToken');
if (!token) return;
```

Figure 6.12: Session Management Token in AI Chatbot screen

All interactions were secured using JWT authentication. The access token, stored in AsyncStorage, was included in every API request to endpoints such as /chat/message, /chat/models, and /chat/history. This ensured that only authenticated users could access AI features and their data remained private and secure. This module exemplified a robust integration of AI, personalization, and user-centered design, delivering intelligent, context-aware pet care support in a secure, engaging, and reliable format.

6.2.3 Symptom Diagnosis Module

This module enabled users to perform AI-powered health assessments for their pets based on described symptoms. It provided a structured, user-friendly interface that guided pet owners through the process of submitting symptoms and receiving actionable insights in a clear, organized format.

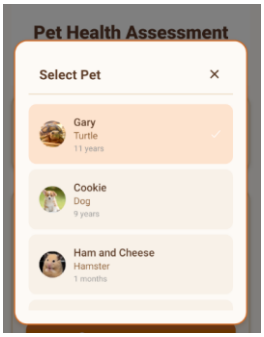
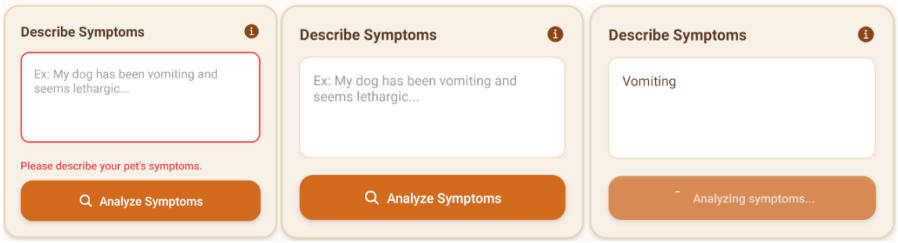


Figure 6.13: Symptom Diagnosis Screen Pet Selection Modal

The screen beginned with a pet selector button at the top, allowing users to choose which pet they are assessing. Tapping this opened a modal listing all registered pets with their photos, names, breeds, and calculated ages, ensuring the AI received accurate context for personalized results.



(a) (b) (c)

Figure 6.14(a)(b)(c): Input Validation and Loading Indicator for Severity Assessment

Below, a multi-line text input field allowed users to freely describe their pet's symptoms and input was validated. If the text input was empty, a red error validation appeared at the below encouraging users to input symptoms. A prominent "Analyze Symptoms" button triggered the diagnosis process, disabled during loading to prevent duplicate submissions. While the system processed the request, a loading indicator with "Analysing symptoms…" was displayed. Once completed, the AI-generated diagnosis was presented in a structured card layout with clearly labeled sections as show in the figure below.

Severity Assessment :

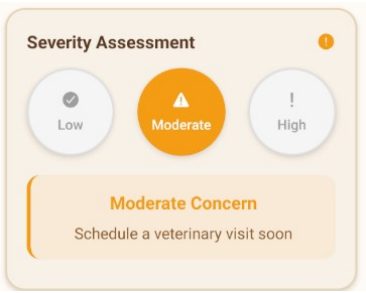


Figure 6.15: Severity Assessment Results

A color-coded severity indicator appeared at the top of the result card such as Green for Low Risk, Yellow for Moderate Concern, Red for High Priority. This visual system helped users quickly understand the urgency of the situation and decide whether to monitor at home or seek immediate veterinary care.

Diagnosis: The identified condition or possible illness.

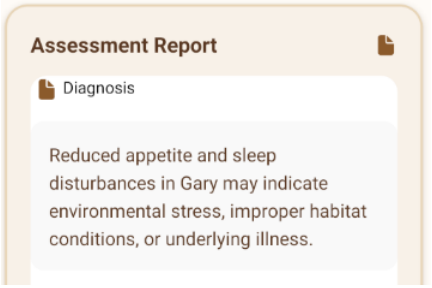


Figure 6.16: Symptom Diagnosis Result Diagnosis

Recommendation: Immediate actions to take.

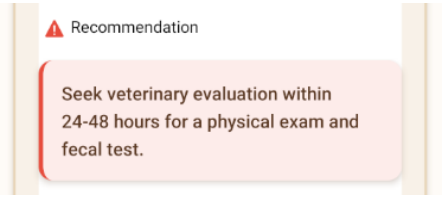


Figure 6.17: Symptom Diagnosis Result Recommendation

Possible Causes: Potential underlying reasons.

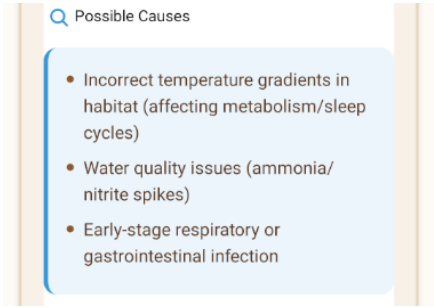


Figure 6.18: Symptom Diagnosis Result Possible Causes

Additional Notes: Contextual advice or observations.

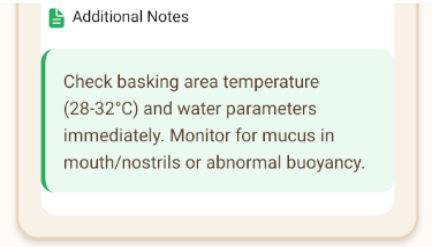


Figure 6.19: Symptom Diagnosis Result Additional Notes

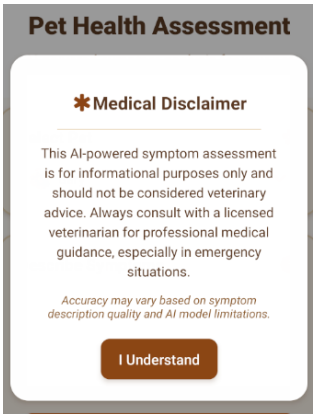
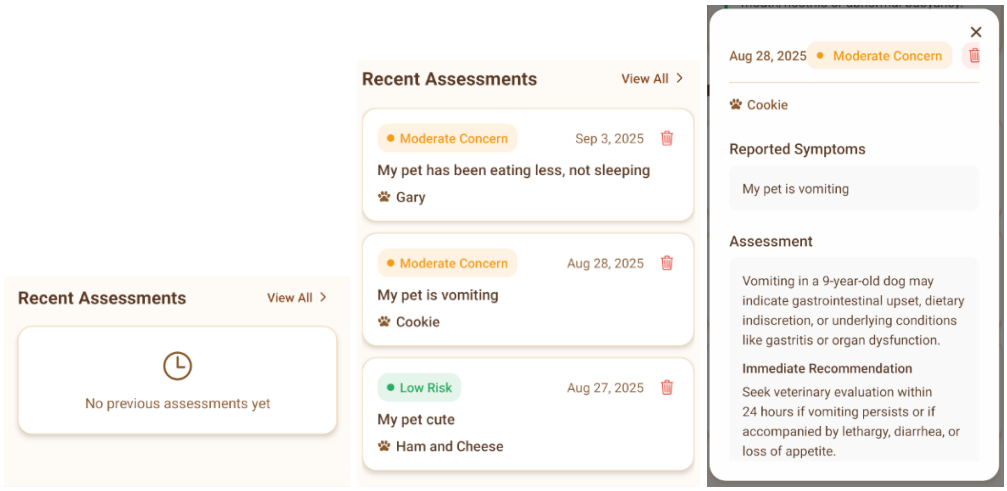


Figure 6.20: Symptom Diagnosis Disclaimer

This visual indicator helped users quickly assess urgency and decide whether to monitor at home or seek veterinary care. The severity level was determined both by the AI model and client-side logic that scanned for keywords in the response. A non-intrusive disclaimer button was there to ensure that users don't follow the AI diagnosis blindly if their pet symptoms were too serious.



(a) (b) (c)

Figure 6.21(a)(b)(c): View Past Assessments in Symptom History Modal

A "Symptom History" button opened a scrollable modal showing all past assessments, including the date, symptoms, associated pet, and severity level. Each entry could be tapped to view full details or deleted with confirmation. If no history existed, an empty state message was shown. This

feature allowed users to track changes over time and review previous AI suggestions. All data was securely transmitted to the Node.js backend via the /symptom/diagnose endpoint, where the AI prompt is enriched with pet details and processed using OpenRouter. The result was saved in the symptom_history table in Supabase for future reference.

```
console.error('Diagnosis error:', error);
setDiagnosisData({
  diagnosis: `Error: ${error.message.includes('unavailable')
    ? 'The AI service is temporarily down. Please try again later.'
    : 'Unable to get diagnosis. Please try again.'}`
});
```

Figure 6.22: Error Handling - AI Service Unavailable

In case of network issues or AI service downtime, a friendly error message was displayed. This maintained a positive user experience while clearly communicating the issue.

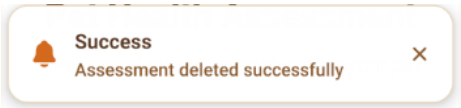


Figure 6.23: NotificationToast component for success message

The module used a reusable NotificationToast component to provide feedback on success message, enhancing usability. It also included real-time validation, session-based state management, and responsive design for consistent performance across devices. This module exemplified the integration of artificial intelligence, personalization, and user-centered design, delivering timely, structured, and visually intuitive pet health insights while emphasizing responsible use and the importance of professional veterinary care.

6.2.4 Pet Management Module

This module enabled users to fully manage their pets' information and health records through an intuitive, organized interface. It was a core component of PawHub, allowing pet owners to maintain comprehensive digital profiles and

medical histories for each pet. The module supported all CRUD (Create, Read, Update, Delete) actions for both pet profiles and health records, ensuring flexibility and data control.

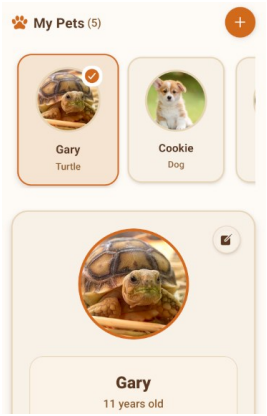
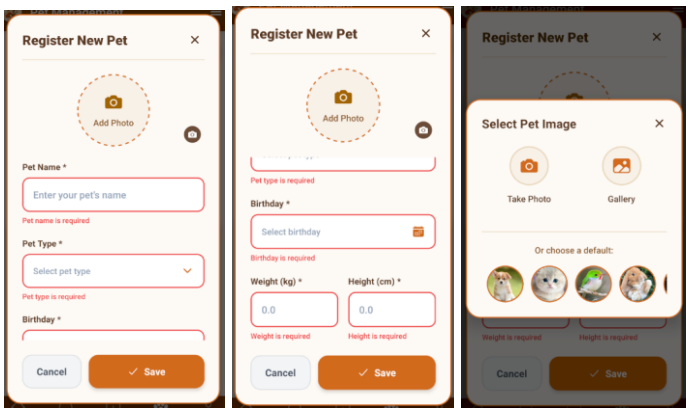


Figure 6.24: List of Registered Pets in Pet Management Screen

The main screen displayed a scrollable list of all registered pets in card format at the top. Each card showed the pet's photo, name, breed, age, and weight, providing a quick visual overview. Tapping a pet card open its detailed profile and associated health records.



(a) (b) (c)

Figure 6.25(a)(b)(c): Add New Pet Form validation and Image Upload Option

Users could add a new pet by tapping the "+" button, which opened a modal form titled "Register New Pet". Every user input had input validation. The app also included eight default pet avatars (dog, cat, bird, rabbit, hamster,

turtle, snake) to ensure every pet has a visual identity even without a custom photo. Image upload was handled via react-native-image-crop-picker, supporting both camera and gallery access.

(a) (b)

Figure 6.26(a)(b): Edit Pet Profile, Update Existing Pet Type Information

Existing pet profiles could be edited by tapping an “Edit” button, which repopulates the same form with current data. All fields were editable, and changes were validated before saved securely to the backend via the /pets endpoint. The module used a reusable CustomAlert component for all confirmation dialogs and a NotificationToast to provide feedback on successful operations.

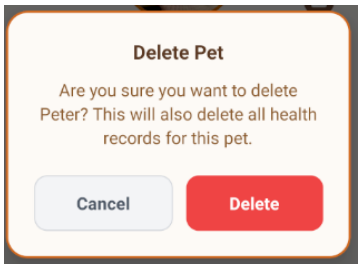


Figure 6.27: Delete Pet Confirmation Alert

Deleting a pet triggered a confirmation alert that warns users. This prevented accidental data loss and emphasized the cascading effect of deletion.

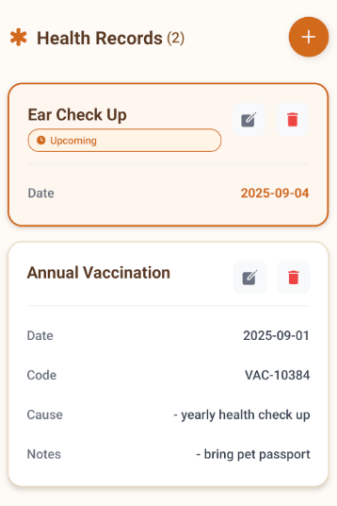
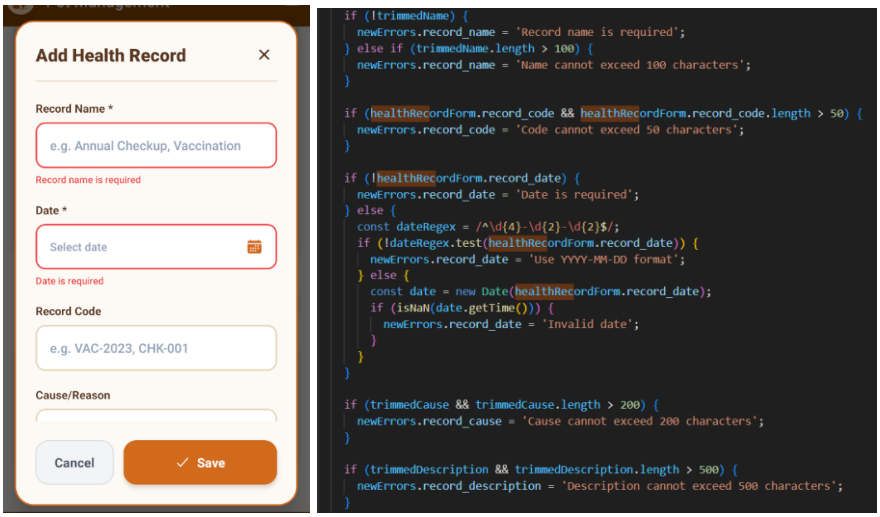


Figure 6.28: Health Records List as Chronological Medical History

Each pet had a dedicated Health Records section where users could view, add, edit, or delete medical entries such as vaccinations, check-ups, treatments, or deworming. Records were displayed in chronological order with key details visible at a glance such as event name, date, cause, and description.

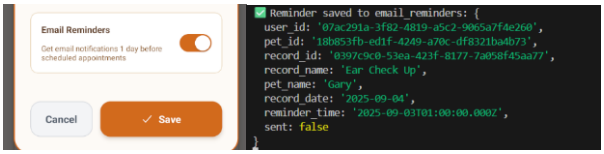


(a) (b)

Figure 6.29(a)(b): Add Health Record Input Validations

To add a record, users filled out a form that captured essential health information. Input validation ensured record name and date were required, Date picker restricted future dates only for scheduled events, Optional fields

(code, cause, description) could be left blank The notification toggle allowed users to enable email reminders for future-dated events.



(a) (b)

Figure 6.30(a)(b): Future Date with Email Reminder Enabled

If the record date was in the future and the notification toggle was on, the system automatically created an entry in the email_reminders table. The backend checked daily for upcoming reminders and sends an email via Resend one day before the event.

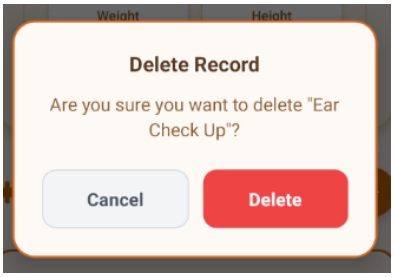


Figure 6.31: Delete Health Record Confirmation

Deleting a health record also used a confirmation alert to prevent accidental removal. This applied to both past and future records, ensuring users were aware of their actions.

```
const token = await AsyncStorage.getItem('accessToken');
```

Figure 6.32: Pet Management Screen session

Data was fetched from and synced with the Node.js backend using secure JWT-authenticated API calls to /pets and /health-records. This module exemplified a robust, user-centered approach to pet health management, combining data integrity, usability, and automation to help pet owners stay organized and proactive in their pet's care.

6.2.5 Profile Management Module

This module allowed users to manage their personal account settings, including profile information, avatar, password, and account security. It offered users a safe and intuitive way to customize their identities within the PawHub app while protecting the privacy and integrity of their data.

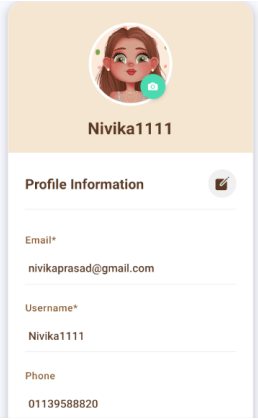


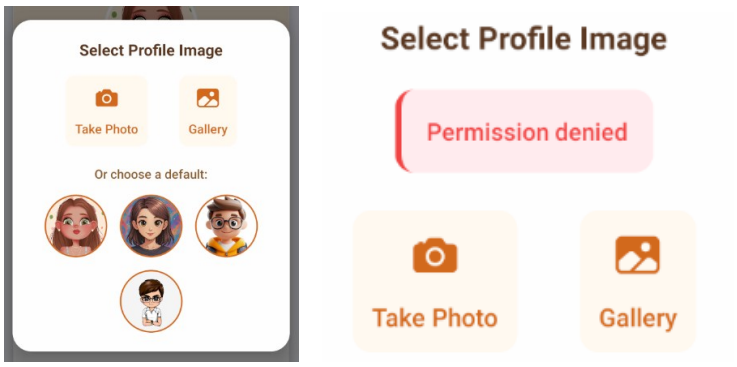
Figure 6.33: Avatar and User Information in Profile Management Screen

The screen was structured as a card-based layout with two main sections which is the Avatar Section and Profile Information.

```
const response = await makeAuthenticatedRequest(`${API_URL}/profile/avatar`, {
  method: 'POST',
  headers: {
    'Content-Type': 'multipart/form-data',
  },
  body: formData,
});
```

Figure 6.34: Selected image is uploaded via multipart/form-data

At the top, users could update their profile picture by tapping the avatar, which opens a modal with multiple options such as take a photo using the camera, select an image from the device gallery, or choose from four predefined default avatars hosted in Supabase Storage. The selected image was uploaded via multipart/form-data to the /profile/avatar endpoint and stored securely in Supabase, with the public URL saved in the profiles table.



(a) (b)

Figure 6.35(a)(b): Camera, Gallery, and Default Avatars in Image Picker

The avatar upload modal included proper Android permission handling (camera and storage), error feedback, and a loading indicator during upload. If the user already had a custom avatar, it was displayed using the uri source in React Native's Image component. If permission was not given to access the gallery or camera, permission denied error was shown.

(a) (b)

Figure 6.36(a)(b): Edit Profile Information with Input Validation

Below the avatar, users could view and edit their email, username, phone number, and password. When in edit mode, the static text fields transformed into input fields with real-time validation. Validation errors were displayed below each field with red text, ensuring users could correct mistakes before submission.

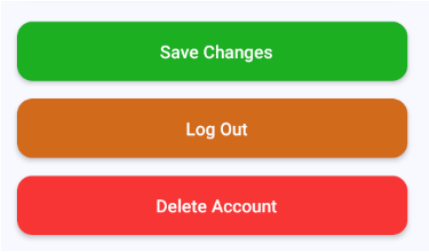


Figure 6.37: Save Changes, Log Out, and Delete Account Buttons

The module included three action buttons such as Save Changes which sends updated profile data to the /profile endpoint via a secure PUT request. A success toast appeared upon completion. Log out which triggered supabase.auth.signOut() to end the session and redirected to the login screen. Delete account which showed a confirmation alert warning that all data will be permanently deleted. If confirmed, a DELETE request was sent to /profile, the account was soft-deleted, and the user was signed out.

```
// Helper function to make authenticated API calls
const makeAuthenticatedRequest = async (url, options = {}) => {
  let token = await getAuthToken();

  if (!token) {
    const { data: { session }, error } = await supabase.auth.refreshSession();
    if (error || !session) {
      throw new Error('Please log in again');
    }
    token = session.access_token;
  }

  const response = await fetch(url, {
    ...options,
    headers: {
      ...options.headers,
      Authorization: `Bearer ${token}`,
      ...(options.body instanceof FormData ? {} : { 'Content-Type': 'application/json' }),
    }
  });
```

Figure 6.38: Session Management Token in Profile Management

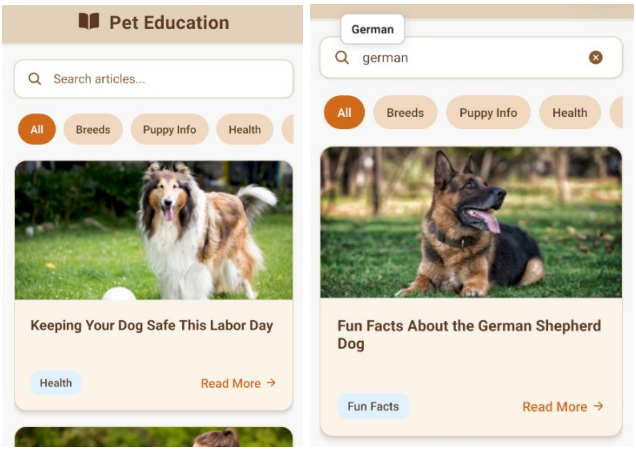
All API interactions were protected with JWT authentication. The app used a custom makeAuthenticatedRequest helper that automatically retrieved the access token from AsyncStorage or refreshed the session if expired. This ensured secure, seamless communication with the backend.

Figure 6.39: Success Toast on Profile Update

A success toast (“Profile updated successfully!”) appeared after a successful save, reinforcing positive user feedback. This module exemplifiee a secure, well-structured approach to user account management, combining real-time validation, secure image handling, and safe data deletion to deliver a reliable and trustworthy experience.

6.2.6 Education Module

This module provided users with access to reliable, up-to-date pet care information sourced from trusted authorities such as the American Kennel Club (AKC). It served as a knowledge hub within the PawHub application, empowering pet owners with expert-backed articles on topics including dog breeds, puppy care, health, training, nutrition, travel, dog sports, and fun facts. The interface’s simplicity, ease of use, and intuitive design guarantee that users found the right information easily depending on their needs or interests.



(a) (b)

Figure 6.40(a)(b): Pet Education Screens Header and Search Bar

The screen featured a well-organized layout beginning with a prominent header. Below, a search bar allowed users to type keywords and filter articles in real time by title or category. The search input included a clear button to reset the query, enhancing usability and interactivity.

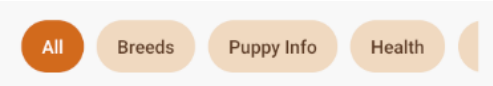
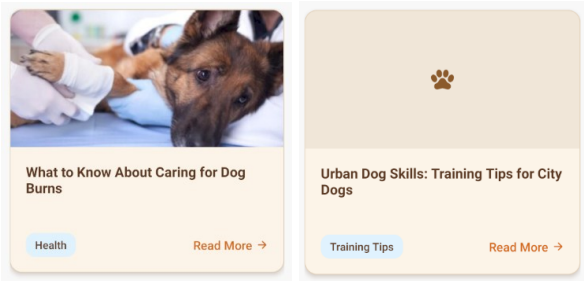


Figure 6.41: Category Tabs

A horizontal scrollable tab bar displayed all available categories: All, Breeds, Puppy Info, Health, Training, Nutrition, Travel, Sports, and Fun Facts. Each tab was styled as a pill-shaped button, with the active category highlighted in a warm orange color for clear visual feedback. Tapping a category filtered the article list to show only those matching the selected topic, enabling focused browsing.



(a) (b)

Figure 6.42(a)(b): Article Card Layout

Articles were displayed in a vertical scrollable list using card-based design. Each card included a thumbnail image at the top, followed by the article title, and a footer with the category badge and a “Read More” button with an arrow icon. If an article lacked an image, a placeholder paw icon was shown to maintain visual consistency.

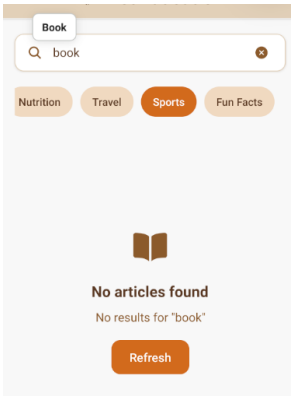
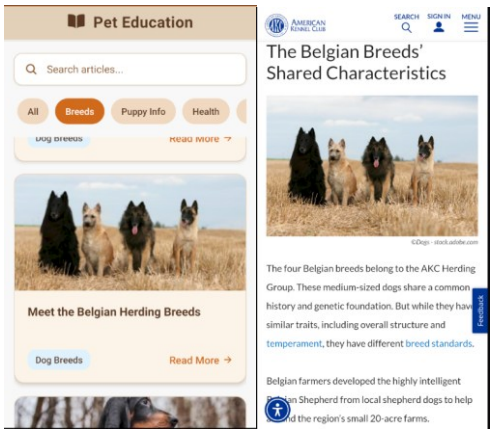


Figure 6.43: No Articles Found

When no articles matched the search or selected category, an empty state was displayed with a book icon, a descriptive message and a “Refresh” button to retry loading data. This improved user experience by providing feedback and recovery options.



(a) (b)

Figure 6.44(a)(b): Opening Article in External Browser

Tapping on an article card or the “Read More” button opened the original webpage in the device’s default browser using Linking.openURL(). This ensured users receive the most current and accurate information directly from the source, while also respecting content licensing and avoiding duplication.

```
const fetchArticles = async () => {
  setLoading(true);
  try {
    const response = await fetch(`${API_URL}/articles`);

    if (!response.ok) {
      throw new Error('Failed to fetch articles');
    }

    const data = await response.json();
    setArticles(data);
    setFilteredArticles(data);
  } catch (err) {
      console.error('Fetch articles error:', err);
      Alert.alert('Error', 'Could not load articles. Please try again.');
      setArticles([]);
      setFilteredArticles([]);
  } finally {
      setLoading(false);
  }
};
```

Figure 6.45: Fetching Articles from Backend

On initial load, a loading indicator appeared with the message "Loading articles…" to inform users that content is being fetched from the backend. The data was retrieved via a secure GET request to the /articles endpoint, which pulls from the articles table in Supabase populated weekly by a web scraper running on a application terminal. By connecting AI-powered features with trusted resources, it reinforced the app's role as a comprehensive pet wellness platform.

6.2.7 Feedback Module

This module allowed users to share their experience and provide valuable feedback on the PawHub application, helping to guide future improvements and ensure the app meets user expectations. It featured a clean, user-friendly interface that encouraged honest and constructive input through a structured feedback form and an accessible history view.

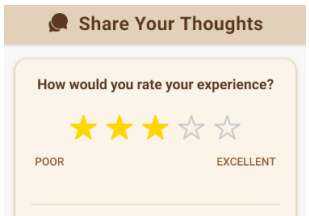


Figure 6.46: Header and Rating Section in Feedback Screen

The screen opened with a welcoming header. Below, users were prompted to rate their overall experience using a 5-star system displayed

prominently in the center of a card. Each star was interactive, allowing users to tap and select a rating from 1 (Poor) to 5 (Excellent).

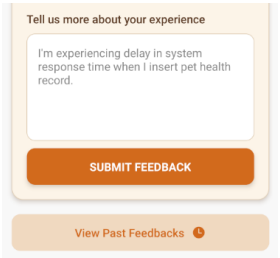
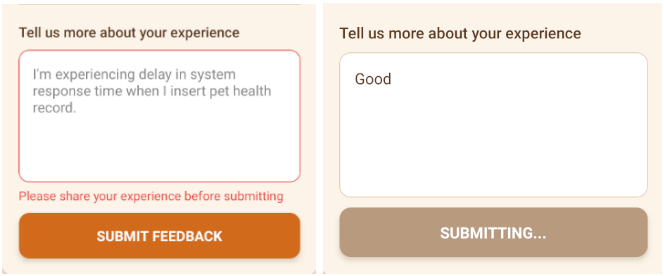


Figure 6.47: Feedback Input Field with Placeholder

Below the rating, users were asked to "Tell us more about your experience" in a multi-line text input field. The field included a descriptive placeholder to guide users in providing detailed feedback. The input supported text wrapping and dynamic expansion up to a reasonable height, ensuring users could express themselves fully.



(a) (b)

Figure 6.48(a)(b): Submit Feedback Button and Error Handling

A "Submit Feedback" button was displayed at the bottom of the card. If the user attempted to submit without entering any text, a validation error appeared with the input field border turning red to highlight the issue. During submission, the button changed to "SUBMITTING…" and disabled further interaction to prevent duplicate entries.

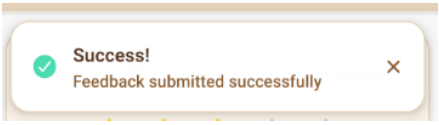


Figure 6.49: Success Notification Toast on Submission

Upon successful submission, a toast notification slided in from the top with a checkmark icon. The toast automatically disappeared after 3 seconds, providing subtle but clear confirmation without interrupting the user flow.

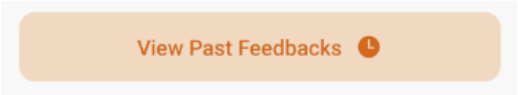


Figure 6.50: View Past Feedbacks Button

A dedicated “View Past Feedbacks” button was placed below the form, styled with a soft background and an icon of a clock. Tapping it opened a full-screen modal that retrieved and displayed all previously submitted feedback entries.

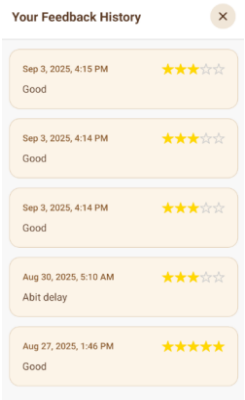


Figure 6.51: List of Submissions in Past Feedbacks Modal

The modal showed a scrollable list of past feedback, each entry including the submission date and time, the star rating, and the feedback text. Entries were listed in reverse chronological order, with the most recent at the top. If no feedback has been submitted, an empty state was shown with an icon and the message: “No feedback submitted yet”.

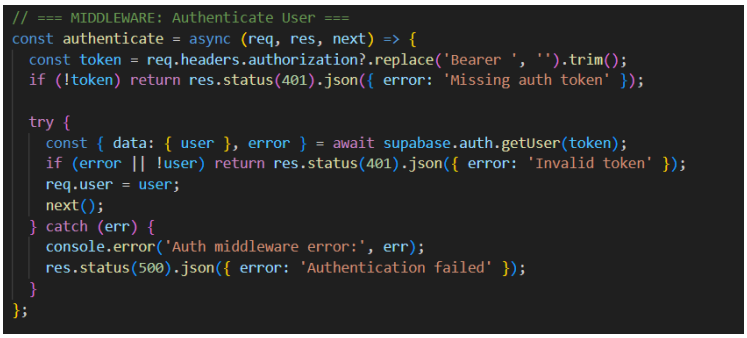
```
const fetchPastFeedbacks = async () => {
  setLoadingPast(true);
  try {
    const token = await AsyncStorage.getItem('accessToken');
    if (!token) throw new Error('No auth token');

    const response = await fetch(`${API_URL}/feedback`, {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    });

    if (!response.ok) throw new Error('Failed to fetch feedback');

    const data = await response.json();
    setPastFeedbacks(data);
  } catch (error) {
    console.error('Failed to fetch past feedbacks:', error);
    // Optionally show alert
  } finally {
    setLoadingPast(false);
  }
};
```

Figure 6.52: Loading Indicator in Feedback History Modal

When opening the history, a loading indicator appeared if data is being fetched from the backend via the /feedback endpoint. The request is authenticated using a JWT token retrieved from AsyncStorage, ensuring only the current user's data is accessed. All feedback was securely sent to the backend. This data can later be analyzed to improve app functionality, fix bugs, and enhance the user experience. This module exemplified a thoughtful approach to create a reliable feedback mechanism that empowered users to contribute to the app's continuous improvement.

6.3 Backend Implementation

The backend of PawHub was developed using Node.js with Express.js, forming a secure and scalable server that serves as the central control layer between the React Native frontend and external services such as Supabase, OpenRouter, and Resend. Hosted locally at http://10.0.2.2:3000 for Android emulator testing, the server was designed to handle authentication, data processing, API routing, and automated tasks with robust error handling and logging. The architecture followed a modular structure, separating concerns into routes, controllers, and middleware to ensure maintainability and clean code organization.



```
// === MIDDLEWARE: Authenticate User ===
const authenticate = async (req, res, next) => {
  const token = req.headers.authorization?.replace('Bearer ', '').trim();
  if (!token) return res.status(401).json({ error: 'Missing auth token' });

  try {
    const { data: { user }, error } = await supabase.auth.getUser(token);
    if (error || !user) return res.status(401).json({ error: 'Invalid token' });
    req.user = user;
    next();
  } catch (err) {
    console.error('Auth middleware error:', err);
    res.status(500).json({ error: 'Authentication failed' });
  }
};
```

Figure 6.53: Authentication middleware

All API endpoints were protected by a custom JWT authentication middleware that validated the Authorization: Bearer <token> header for every incoming request. This middleware retrieved the token from the request header, verified it against Supabase Auth, and attached the user ID to the request object if valid. Access was blocked if the token was absent, invalid, or expired, guaranteeing that only users who have been authenticated can use protected routes. Strict data privacy was maintained by this technique, which also stopped unwanted access to private data including chat history, pet profiles, and health records.

```
// === RATE LIMITING ===
const limiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 200,
  standardHeaders: true,
  legacyHeaders: false,
});

const authLimiter = rateLimit({
  windowMs: 15 * 60 * 1000,
  max: 20,
  message: { error: 'Too many attempts. Try again later.' },
  standardHeaders: true,
  legacyHeaders: false,
});

const chatLimiter = rateLimit({
  windowMs: 60 * 1000, // 1 minute
  max: 20, // 20 requests per minute for chat
  message: { error: 'Too many chat requests. Please slow down.' },
  standardHeaders: true,
  legacyHeaders: false,
});
```

Figure 6.54: Rate limiting

Express-rate-limit is used to establish rate limitation in order to improve security and avoid abuse. Different limits were applied based on the endpoint such as 20 attempts per 15 minutes for authentication routes to prevent brute-force attacks, 200 requests per 15 minutes for general usage to ensure fair system access, and 20 messages per minute on chat-related endpoints to avoid spam. These limits were enforced globally and per IP address, providing an additional layer of protection against denial-of-service threats.

The backend exposed a comprehensive RESTful API that supported all core functionalities of the application. Authentication routes such as /auth/login, /auth/register, and /profile handle user sign-in, registration, and profile retrieval. Pet management was facilitated through /pets, which allowed users to create, read, update, and delete pet profiles. Health records were managed via /health-records, supporting CRUD operations and enabling email reminders for future-dated events. AI-powered features were accessible through /chat/message, /chat/models, and /symptom/diagnose, where user inputs were enriched with pet context and sent to OpenRouter for intelligent responses. Additional endpoints like /feedback, /articles, and /home support feedback submission, educational content retrieval, and consolidated dashboard data, respectively.

```
// === SUPABASE CLIENT (anon) - for RLS-protected tables
const supabase = createClient(
  process.env.SUPABASE_URL,
  process.env.SUPABASE_ANON_KEY
);

// === SUPABASE ADMIN CLIENT (service_role) - for admin/auth operations
const supabaseAdmin = createClient(
  process.env.SUPABASE_URL,
  process.env.SUPABASE_SERVICE_ROLE_KEY
);
```

Figure 6.55: Supabase Client SDK

Data persistence was handled through Supabase, a PostgreSQL-based backend-as-a-service platform. The backend interacted with Supabase using the Supabase client SDK, ensuring secure and efficient database operations. Key tables include profiles, pets, health_records, symptom_history, chat_history, email_reminders, articles, and feedback, each designed with proper relationships and constraints. Row Level Security (RLS) policies are enforced on all tables to ensure that users can only access their own data, preventing cross-user data exposure. Foreign key constraints maintain referential integrity across related entities, preserving data consistency.

```
    // Build system prompt
    const buildSystemPrompt = () => {
      if (!selectedPet) {
        return `
You are PawHub, a friendly pet care assistant.
Guidelines:
1. Provide helpful advice for common pet concerns
2. Ask for pet details if needed for more specific advice
3. For emergencies, recommend immediate vet consultation
4. For general questions, provide concise answers
5. Give tips on pet care and health
6. Use metric measurements only
        `.trim();
      }

      const healthSummary = healthRecords.length > 0
        ? `Recent Health History (${healthRecords.length} records):\n${healthRecords.map(r =>
            `- ${r.record_name} (${r.record_date}): ${r.record_description || 'No details'} ${r.record_cause ? `| Cause: ${r.record_cause}` : ''}`
          ).join('\n')}`
        : "No health history recorded";

      return `
You are PawHub, an expert veterinary assistant for ${selectedPet.pet_name}.
Pet Profile:
- Name: ${selectedPet.pet_name}
- Breed: ${selectedPet.pet_breed || 'Unknown'}
- Age: ${calculatePetAge(selectedPet.pet_birthday)}
- Weight: ${selectedPet.pet_weight ? `${selectedPet.pet_weight} kg` : 'Unknown'}

${healthSummary}

Guidelines:
1. Provide concise, professional advice (2-3 sentences)
2. Consider breed-specific health concerns and the pet's health history
3. For emergencies, recommend immediate vet consultation
4. Use metric measurements only
5. Always ask for symptoms duration and severity for health concerns
6. Reference specific health records when relevant to the question
7. Use a friendly, supportive tone
8. If unsure, suggest consulting a vet
9. Take into account the pet's recent health records when giving advice
10. Only reference past conversations if directly relevant to the current question
11. If the user asks a new, unrelated question, focus only on the current query
      `.trim();
    };

    const systemPrompt = buildSystemPrompt();
```

Figure 6.56: Context-Rich AI Prompt with pet information

For AI-powered features, the backend integrated with OpenRouter.ai, allowing access to advanced language models such as Microsoft MAI DS R1, Mistral-7B-Instruct, and Horizon Alpha. When a user submits a query or symptom description, the server fetched the selected pet's details and health records from Supabase, constructed a context-rich prompt, and sent it to OpenRouter with appropriate headers including Authorization, HTTP-Referer, and X-Title. The AI response was parsed into a structured format with sections such as Diagnosis, Severity, Recommendation, Possible Causes, and Additional Notes, then saved to the symptom_history or chat_history table for future reference.

```
// ✅ Fallback logic
const modelOrder = preferredModel && OPENROUTER_MODELS.some(m => m.id === preferredModel)
  ? [preferredModel, ...OPENROUTER_MODELS.map(m => m.id).filter(m => m !== preferredModel)]
  : OPENROUTER_MODELS.map(m => m.id);

let aiResponse = null;
let usedModel = null;
```

Figure 6.57: Fallback logic to alternative models

A Model Control & Prioritization (MCP) strategy ensured reliability by automatically falling back to alternative models if the primary one fails, minimizing downtime and maintaining service continuity.

```
// Check every minute for due email reminders
function startEmailChecker() {
  setInterval(async () => {
    const now = new Date();
    console.log('🚨 Email checker running at:', now.toISOString());

    try {
      // Get all unsent reminders that are due (reminder_time <= now)
      const { data, error } = await supabase
        .from('email_reminders')
        .select('*')
        .is('sent', false)
        .lte('reminder_time', now.toISOString());

      if (error) throw error;

      console.log(`📩 Found ${data.length} pending reminders`);

      for (const reminder of data) {
        console.log('🎯 Processing reminder:', reminder);
        await sendEmailReminder(reminder);
      }
    } catch (err) {
      console.error('💥 Email check interval error:', err);
    }
  }, 60000); // Check every minute
}
```

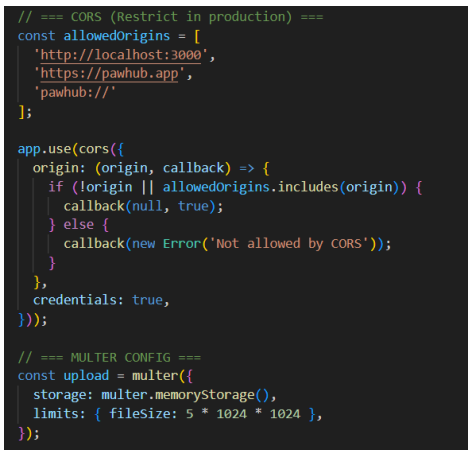
Figure 6.58: Email Reminder checker task runs every 60 seconds

An essential component of the backend is the automated email reminder system, powered by Resend. A background task runs every 60 seconds via setInterval, querying the email_reminders table for records where the reminder_time is in the near future and the sent flag is still false. For each matching record, an email is sent to the user with personalized content such as the pet's name, event type, and scheduled date. After successful delivery, the sent field is updated to true, preventing duplicate emails. This proactive notification system helped users stay on top of preventive care, such as vaccinations and check-ups, without manual tracking.

```
async function scrapeAndStore() {
  try {
    for (const cat of categories) {
      console.log(`\n=== Scraping: ${cat.name} ===`);
      const { data: html } = await axiosInstance.get(cat.url);
      const $ = cheerio.load(html);

      let articles = [];

      // AKC specific parsing logic
      if (cat.url.includes('akc.org')) {
      $('.content-card').each((_, el) => {
        const title = $(el).find('.content-card__title').text().trim();
        const summary =
          $(el).find('.content-card__body p').text().trim() ||
          $(el).find('.content-card__dek').text().trim() ||
          '';
        const link = $(el).find('.content-card__title').attr('href');

        let image =
          $(el).find('img').attr('data-src') ||
          $(el).find('img').attr('src') ||
          '';

        if (image && image.startsWith('//')) {
          image = 'https:' + image;
        }

        const article = { title, summary, link, image, category: cat.name };
        // console.log('Parsed AKC article:', article);
        if (title && link) articles.push(article);
      });
```

Figure 6.59: Scraper runs and fetches articles

Educational content displayed in the app is sourced from the American Kennel Club (AKC) through a terminal runs weekly web scraper, the scraper runs and fetched articles from AKC's website across categories including Dog Breeds, Health, Training, Nutrition, Travel, Dog Sports, and Fun Facts. Using Axios and Cheerio, it extracted the title, link, image URL, and category, then upserted the data into the articles table in Supabase. This ensured that the Education screen remained up-to-date with fresh, expert-backed content.



```
// === CORS (Restrict in production) ===
const allowedOrigins = [
  'http://localhost:3000',
  'https://pawhub.app',
  'pawhub://'
];

app.use(cors({
  origin: (origin, callback) => {
    if (!origin || allowedOrigins.includes(origin)) {
      callback(null, true);
    } else {
      callback(new Error('Not allowed by CORS'));
    }
  },
  credentials: true,
}));

// === MULTER CONFIG ===
const upload = multer({
  storage: multer.memoryStorage(),
  limits: { fileSize: 5 * 1024 * 1024 },
});
```

Figure 6.60: CORS and Multer Configuration

Security is a top priority throughout the backend implementation. The server used Helmet.js to set secure HTTP headers, CORS was configured to allow only trusted origins, and all input were validated both on the client and server side to prevent injection attacks. Sensitive API keys such as OPENROUTER_API_KEY and RESEND_API_KEY were stored in environment variables and never exposed in the codebase. File uploads for profile pictures were handled securely using Multer, with images stored in Supabase Storage and public URLs saved in the profiles table. Passwords were managed entirely by Supabase Auth and were never accessible to the application, ensuring compliance with best practices in user data protection.

This backend design demonstrated a robust, intelligent, and secure architecture that seamlessly integrated frontend, database, AI, and automation systems. By combining modern tools, modular design, and strong security practices, PawHub delivered a reliable and scalable pet care solution that enhanced user experience while maintaining data integrity and privacy.

6.4 Database Integration

The PawHub application was set up using Supabase, a powerful backend-as-a-service platform built on PostgreSQL, to manage all data storage and retrieval efficiently. The setup process began with creating a new project on the Supabase dashboard, where the database was initialized and connected to both the React Native frontend and Node.js backend. Tables were created either through the Supabase web interface or by executing raw SQL scripts in the SQL Editor, ensuring consistency and accuracy in the schema design. This centralized setup allowed seamless integration of authentication, file storage, real-time updates, and Row Level Security (RLS), providing a robust foundation for the entire application.

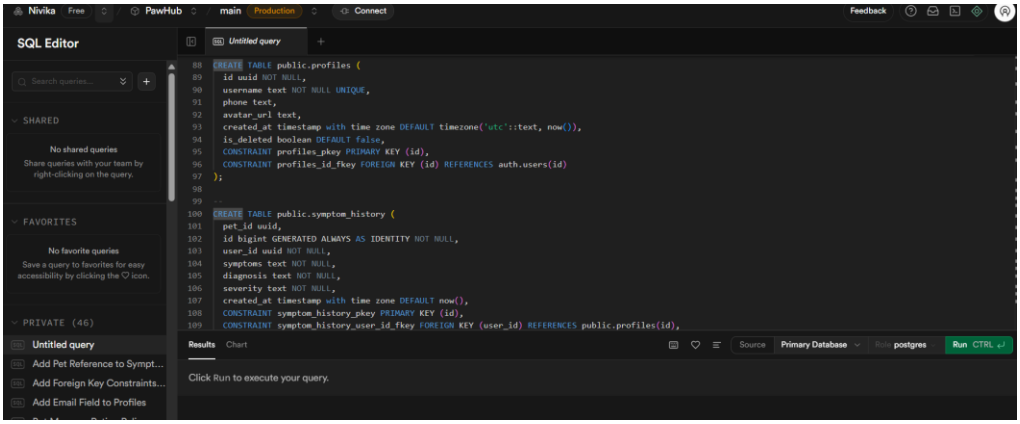


Figure 6.61: Creation of database table using PostgreSQL in supabase

PostgreSQL served as the core relational database engine for PawHub, offering reliability, scalability, and advanced querying capabilities. It stored structured data such as user profiles, pet information, health records, AI chat logs, symptom history, feedback, and educational articles. The use of PostgreSQL ensured strong data integrity through constraints, transactions, and indexing, while also supporting JSON fields and full-text search for future enhancements. Its compatibility with Supabase enabled smooth interaction with the frontend and backend, making it an ideal choice for a data-intensive mobile application like PawHub.

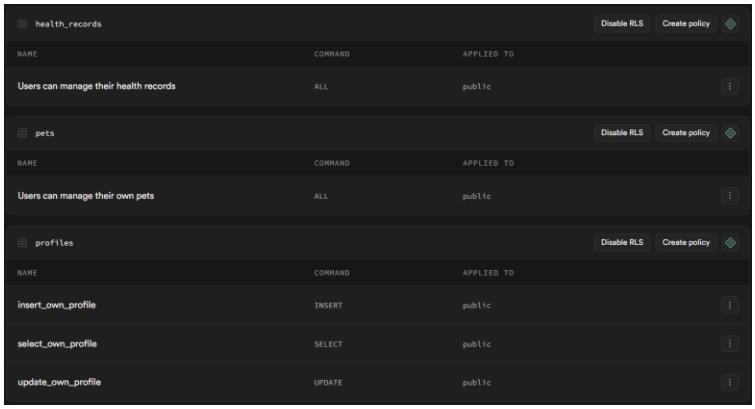


Figure 6.62: RLS Policies in Supabase

Row Level Security (RLS) policies were implemented across all relevant tables to enforce strict data privacy and access control. These policies ensured that users can only view, edit, or delete their own data, preventing unauthorized access to other users' information. For example, a user can only access pet profiles, health records, chat history, and symptom diagnoses linked to their unique user ID. RLS was enabled on tables such as profiles, pets, health_records, chat_history, symptom_history, and feedback, with policies defined using PostgreSQL's native syntax. This security model was essential for maintaining user trust and complying with data protection standards.

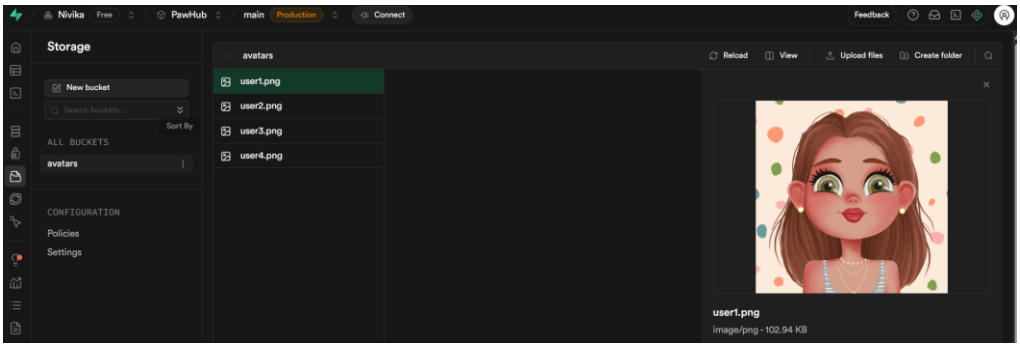


Figure 6.63: Avatar Bucket in Supabase Storage

To manage profile pictures for users, a dedicated storage bucket named "avatars" was created within Supabase Storage. This bucket securely hosted all uploaded images, with public URLs generated for display in the app. Default avatars for users were pre-uploaded to this bucket, ensuring visual consistency even when no custom image was selected. The backend handles

image uploads via Multer middleware, processed the file, and stored it in the bucket with appropriate metadata. Access to the bucket is restricted using policies, allowing only authenticated users to upload or modify their own images.

```
const { data, error } = await supabase.auth.signUp({
  email,
  password,
  options: {
    data: { username, phone } // Not stored in auth metadata unless used in RLS
  }
});
```

```
const { data, error } = await supabase.auth.signInWithPassword({ email, password });
```

(a) (b)

Figure 6.64(a)(b): JWT-based token generation

Supabase Auth managed authentication and offers session management, safe email/password login, and token creation based on JWT. Upon registration or login, Supabase creates a user entry in the auth.users table and triggers the creation of a corresponding record in the profiles table. This integration simplified user management and ensured secure access to protected routes. The system used the supabase.auth.signInWithPassword() and supabase.auth.getUser() methods to manage sessions, while the backend verified JWT tokens in API requests to authenticate users before granting access to sensitive data.

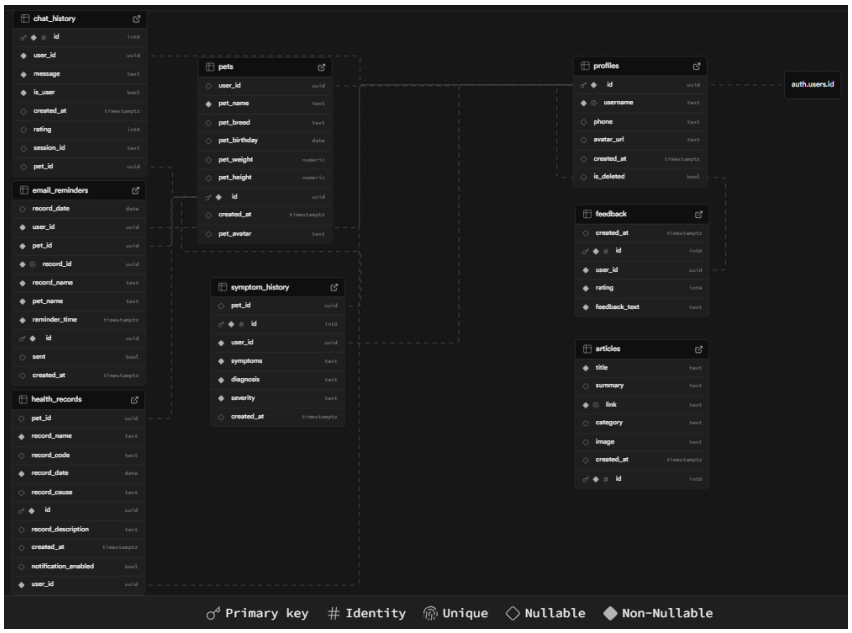


Figure 6.65: Database Schema in Supabase

The database schema is carefully designed to reflect the relationships between entities in the PawHub ecosystem. Key tables include profiles, pets, health_records, symptom_history, chat_history, email_reminders, articles, and feedback. Each table is structured with appropriate data types, primary keys, and constraints to ensure consistency. For instance, the pets table includes fields such as pet_name, pet_breed, pet_birthday, pet_weight, pet_height, and pet_avatar, all linked to the owner via the user_id foreign key. This well-organized schema supported the app's functionality while allowing room for future expansion.

Foreign key constraints were used throughout the database to maintain referential integrity and enforce logical relationships between tables. For example, the pets table references the profiles table via the user_id field, ensuring that every pet belongs to a valid user. Similarly, health_records and symptom_history tables reference the pets table through pet_id, linking medical events and AI diagnoses to specific animals. These constraints prevent orphaned records and ensure that data remains consistent and accurate across the system.

(a) (b)

Figure 6.66(a)(b): Environment Variable Import and Validations in Backend

Environment variables were securely managed using a .env file in both the frontend (PawHub/.env) and backend (PawHub/backend/.env) directories. Sensitive credentials were stored in these files and loaded at runtime using the dotenv package. This approach keeps API keys and configuration settings out of the source code, reducing the risk of exposure.

The backend validated the presence of these variables at startup, ensuring the server cannot run without proper configuration.

6.5 AI Features Implementation

The AI features in PawHub were designed to provide pet owners with intelligent, real-time support through two core modules, the AI Chatbot and the Symptom Diagnosis tool. These features leverage advanced language models via OpenRouter.ai, enabling the app to deliver context-aware, personalized responses that enhance user experience while maintaining reliability through model fallback logic. The integration of AI into pet care allowed users to receive immediate guidance on health, behavior, nutrition, and training, bridging the gap between everyday concerns and professional veterinary care.

6.5.1 AI Chatbot

The AI Chatbot served as the interactive hub for general pet care assistance, allowing users to ask questions and receive instant, conversational responses.

```
// === OPENROUTER CONFIG ===
const OPENROUTER_API_KEY = process.env.OPENROUTER_API_KEY;

const OPENROUTER_MODELS = [
  { id: 'microsoft/mai-ds-r1:free', name: 'Microsoft MAI DS R1', context: 32000 },
  { id: 'mistralai/mistral-7b-instruct', name: 'Mistral-7B-Instruct', context: 8000 },
  { id: 'openrouter/horizon-alpha', name: 'Horizon Alpha', context: 256000 }
];
```

Figure 6.67: OpenRouter Configuration

Built on a multi-model architecture, the chatbot supported three AI models: Microsoft MAI DS R1 (primary), Mistral-7B-Instruct, and Horizon Alpha, giving users the flexibility to choose based on response style, depth, or performance. This model selection were implemented through a dropdown interface in the app, enabling dynamic routing of queries to the selected model via OpenRouter's API.

```
    // Build system prompt
    const buildSystemPrompt = () => {
      if (!selectedPet) {
        return `
You are PawHub, a friendly pet care assistant.
Guidelines:
1. Provide helpful advice for common pet concerns
2. Ask for pet details if needed for more specific advice
3. For emergencies, recommend immediate vet consultation
4. For general questions, provide concise answers
5. Give tips on pet care and health
6. Use metric measurements only
        `.trim();
      }

      const healthSummary = healthRecords.length > 0
        ? `Recent Health History (${healthRecords.length} records):\n${healthRecords.map(r =>
            `- ${r.record_name} (${r.record_date}): ${r.record_description || 'No details'} ${r.record_cause ? `| Cause: ${r.record_cause}` : ''}`
          ).join('\n')}`
        : "No health history recorded";

      return `
You are PawHub, an expert veterinary assistant for ${selectedPet.pet_name}.
Pet Profile:
- Name: ${selectedPet.pet_name}
- Breed: ${selectedPet.pet_breed || 'Unknown'}
- Age: ${calculatePetAge(selectedPet.pet_birthday)}
- Weight: ${selectedPet.pet_weight ? `${selectedPet.pet_weight} kg` : 'Unknown'}

${healthSummary}

Guidelines:
1. Provide concise, professional advice (2-3 sentences)
2. Consider breed-specific health concerns and the pet's health history
3. For emergencies, recommend immediate vet consultation
4. Use metric measurements only
5. Always ask for symptoms duration and severity for health concerns
6. Reference specific health records when relevant to the question
7. Use a friendly, supportive tone
8. If unsure, suggest consulting a vet
9. Take into account the pet's recent health records when giving advice
10. Only reference past conversations if directly relevant to the current question
11. If the user asks a new, unrelated question, focus only on the current query
      `.trim();
    };

    const systemPrompt = buildSystemPrompt();
```

Figure 6.68: Context Rich AI prompt for AI Chatbot

When a user submitted a message, the backend constructs a context-rich prompt that included the selected pet's profile such as breed, age, weight, and medical history to ensure responses were tailored and relevant. The request was sent with proper authentication headers (Authorization, HTTP-Referer, X-Title) to OpenRouter, and the AI response was parsed and displayed in a familiar messaging interface. A typing indicator provided visual feedback during response generation, enhancing perceived responsiveness.

```
// ✅ Fallback logic
const modelOrder = preferredModel && OPENROUTER_MODELS.some(m => m.id === preferredModel)
  ? [preferredModel, ...OPENROUTER_MODELS.map(m => m.id).filter(m => m !== preferredModel)]
  : OPENROUTER_MODELS.map(m => m.id);

let aiResponse = null;
let usedModel = null;

for (const model of modelOrder) {
  try {
    console.log(`🔁 Trying model: ${model}`);
```

Figure 6.69: Fallback logic for AI Chatbot

To ensure continuous availability, the system implemented Model Control & Prioritization (MCP) logic. If the primary model failed or timed out, the backend automatically retries the request with an alternative model,

minimizing downtime and maintaining service continuity. Users could rate each AI message with 1–5 stars, and this feedback was securely stored in the chat_history table in Supabase for future analysis and model improvement.

```
// Save user message immediately
const { data: userMessage, error: userMsgError } = await supabase
  .from('chat_history')
  .insert([{
    user_id: userId,
    message: message.trim(),
    is_user: true,
    session_id
  }])
  .select()
  .single();

if (userMsgError) throw userMsgError;
```

Figure 6.70: Save chat history immediately with a unique session_id

The chat maintained session continuity using a unique session_id, allowing users to resume conversations with preserved context. When a different pet was selected, the chat resets to ensure accurate, pet-specific advice. All interactions were protected with JWT authentication, and the frontend used secure storage via AsyncStorage to manage user sessions. This module exemplified a robust integration of AI, personalization, and user-centered design, delivering intelligent, context-aware pet care support in a secure and engaging format.

6.5.2 Symptom Diagnosis

The Symptom Diagnosis module enabled users to perform AI-powered health assessments by submitting descriptions of their pet's symptoms. This function was intended to assist pet owners in determining if the situation required immediate veterinarian care and in assessing the severity of the problem. When a user entered symptoms and selects a pet, the backend retrieved the pet's full profile and constructed a detailed prompt enriched with breed-specific tendencies, age, weight, and existing health records.

```
    // STRICT PROMPT: Enforces full structure
    const systemPrompt = `
You are PawHub, an expert veterinary assistant for ${selectedPet.pet_name}.
Pet Profile:
- Name: ${selectedPet.pet_name}
- Breed: ${selectedPet.pet_breed || 'Unknown'}
- Age: ${calculatePetAge(selectedPet.pet_birthday)}
- Weight: ${selectedPet.pet_weight ? `${selectedPet.pet_weight} kg` : 'Unknown'}
${healthSummary}

### INSTRUCTIONS ###
Based on the pet profile and the symptoms provided by the user, provide a detailed diagnosis.
You MUST respond in a STRICTLY STRUCTURED FORMAT as shown below.
Respond in **exactly this format**. Never deviate:
Diagnosis: <clear diagnosis in 1-2 sentences>
Severity: <low|moderate|high>
Recommendation: <immediate action user should take, starting with "Please" or "Seek">
Possible causes:
• <cause 1>
• <cause 2>
<optional additional bullet>

Additional notes: <any other relevant info, max 2 sentences>

Do not use markdown. Do not add explanations. Follow this structure exactly.
    `.trim();
```

Figure 6.71: Symptom Diagnosis AI Structured Context Prompt

This structured prompt was sent to OpenRouter, where an AI model analyzed the input and returned a response formatted into key sections: Diagnosis, Severity, Recommendation, Possible Causes, and Additional Notes. The backend parsed this response using string matching and regular expressions to extract each component, ensuring a clean, user-friendly display. A severity level classified as Low Risk, Moderate Concern, or High Priority is determined based on keywords in the AI's output and displayed with a color-coded badge (green, yellow, red) to help users quickly assess urgency.

```
// Save the symptom history
const { data, error } = await supabase
  .from('symptom_history')
  .insert([{
    user_id: userId,
    pet_id: pet_id, // saving pet_id
    symptoms: symptoms.trim(),
    diagnosis: diagnosis.trim(),
    severity
  }])
  .select()
  .single();

if (error) {
  console.error('Supabase error saving symptom history:', error);
  return res.status(500).json({ error: 'Failed to save symptom history' });
}
```

Figure 6.72: Diagnosis Result is Saved in the symptom_history Table

The diagnosis result was saved in the symptom_history table for future reference, allowing users to track changes over time. A dedicated Symptom History button opens a scrollable modal where past assessments can

be viewed or deleted. If no history exists, an empty state was shown with a descriptive message.

```
// === OPENROUTER CONFIG ===
// Define fallback priority order
const MODEL_FALLBACK_ORDER = [
  'microsoft/mai-ds-r1:free',       // Primary (free, fast)
  'mistralai/mistral-7b-instruct',  // Fallback 1
  'openrouter/horizon-alpha'        // Fallback 2 (long context)
];

const OPENROUTER_MODELS2 = MODEL_FALLBACK_ORDER.map(id => {
  const names = {
    'microsoft/mai-ds-r1:free': 'Microsoft MAI DS R1',
    'mistralai/mistral-7b-instruct': 'Mistral-7B-Instruct',
    'openrouter/horizon-alpha': 'Horizon Alpha'
  };
  const contexts = {
    'microsoft/mai-ds-r1:free': 32000,
    'mistralai/mistral-7b-instruct': 8000,
    'openrouter/horizon-alpha': 256000
  };
  return { id, name: names[id], context: contexts[id] };
});
```

Figure 6.73: AI Fallback Logic for Symptom Diagnosis Screen

To ensure continuous availability, the system implemented Model Control & Prioritization (MCP) logic. If the primary model failed or timed out, the backend automatically retried the request with an alternative model, minimizing downtime and maintaining service continuity.

```
// 🚫 All models failed
if (!parsedResponse.diagnosis) {
  return res.status(500).json({
    error: 'AI service is temporarily unavailable. Please try again later.',
    fallback: true
  });
}
```

Figure 6.74: AI Error Message If All Models Fail

In cases where the AI service was temporarily unavailable, a friendly fallback message was displayed: “The AI service is temporarily down. Please try again later.” This graceful degradation ensured a positive user experience even during technical issues. Additionally, a non-intrusive disclaimer was shown with every result: “This AI-powered symptom assessment was for informational purposes only and should not be considered veterinary advice.” This reinforced that the app was a supportive tool, not a replacement for professional care.

The module included real-time validation, session-based state management, and responsive design for consistent performance across devices. By combining artificial intelligence, personalization, and responsible design, the Symptom Diagnosis feature delivered timely, structured, and visually intuitive pet health insights while emphasizing the importance of professional veterinary consultation.

6.6 Automated Email Reminder System

To assist pet owners in remembering their pets' vaccination schedules, the PawHub app featured an efficient and dependable automated email reminder system. This feature is particularly useful for tracking upcoming health events such as vaccinations, deworming, check-ups, and other medical appointments. By reducing the likelihood of missed appointments, the system supported long-term pet wellness and reinforced responsible pet ownership.

```
reminderTime.setDate(reminderTime.getDate() - 1); // 1 day before
reminderTime.setHours(9, 0, 0, 0); // Set to 9 AM for better user experience
```

Figure 6.75: System Calculates the Reminder

When a user added a new health record with a future date and enabled the "Enable Email Reminder" toggle, the backend automatically processed this information and scheduled a notification. The system calculate the reminder time by subtracting 24 hours from the event date and sets the time to 9:00 AM for optimal user engagement. This ensured that the email was delivered one day before the scheduled event, giving the owner sufficient time to prepare.

```
const { data: pet, error: petError } = await supabase
  .from('pets')
  .select('pet_name')
  .eq('id', data.pet_id)
  .single();

if (petError) {
  console.error('Failed to fetch pet for reminder:', petError);
} else {
  const reminderData = {
    user_id: req.user.id,
    pet_id: data.pet_id,
    record_id: data.id,
    record_name: data.record_name,
    pet_name: pet.pet_name,
    record_date: data.record_date,
    reminder_time: reminderTime.toISOString(),
    sent: false
  };

  const { error: upsertError } = await supabase
    .from('email_reminders')
    .upsert(reminderData, {
      onConflict: 'record_id,user_id' // ✅ Matches unique constraint
    });
```

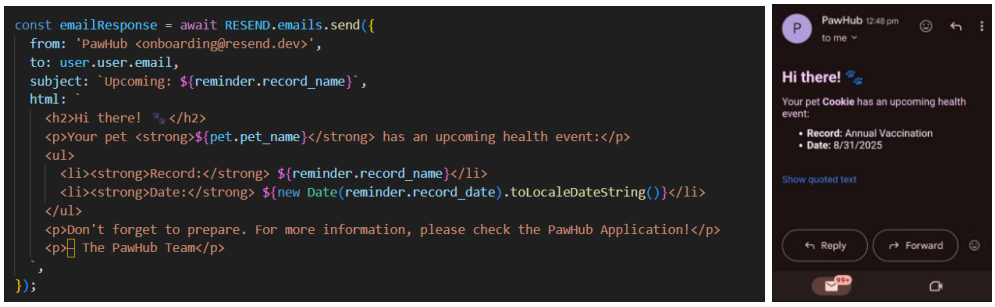
Figure 6.76: Scheduling Logic is implemented

The scheduling logic was implemented within the /health-records POST endpoint. After a new record is successfully saved to the health_records table in Supabase, the backend checks if reminders are enabled and the date is in the future. If so, it retrieved the associated pet's name and constructed a reminder object containing the user ID, pet ID, record ID, event name, pet name, event date, and calculated reminder time. This data is then upserted into the email_reminders table, with a unique constraint on record_id and user_id to prevent duplicate entries.

```
// Check every minute for due email reminders
function startEmailChecker() {
  setInterval(async () => {
    const now = new Date();
    console.log('⏰ Email checker running at:', now.toISOString());

    try {
      // Get all unsent reminders that are due (reminder_time <= now)
      const { data, error } = await supabase
        .from('email_reminders')
        .select('*')
        .is('sent', false)
        .lte('reminder_time', now.toISOString());

      if (error) throw error;

      console.log(`📬 Found ${data.length} pending reminders`);

      for (const reminder of data) {
        console.log('📧 Processing reminder:', reminder);
        await sendEmailReminder(reminder);
      }
    } catch (err) {
      console.error('🚨 Email check interval error:', err);
    }
  }, 60000); // Check every minute
}
```

Figure 6.77: Email Reminder Checker Running Every 60 Seconds

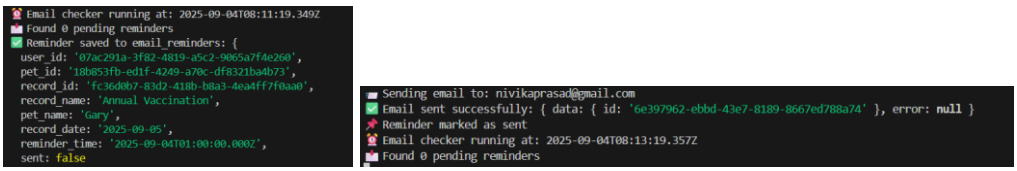
To ensure timely delivery, a background task runs every 60 seconds via setInterval in the Node.js server. This email checker queries the email_reminders table for all records where the reminder_time has passed and the sent flag is still false. For each matching record, the system retrieved the user's email address using Supabase Auth's admin API, then used Resend to send a professionally formatted email with a clear subject line and HTML body.



(a) (b)

Figure 6.78(a)(b): Example Email Sent via Resend

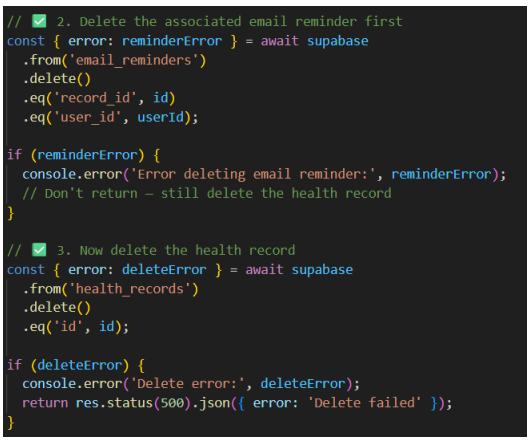
The email content was personalized and included the pet's name, the type of health event, and the scheduled date.



(a) (b)

Figure 6.79(a)(b): Reminder Marked as Sent in Database

Once the email was successfully sent, the sent field in the email_reminders table was updated to true, preventing the same reminder from being sent again. Detailed logging was implemented throughout the process, allowing developers to monitor successful deliveries and troubleshoot any issues.



```
// ✅ 2. Delete the associated email reminder first
const { error: reminderError } = await supabase
  .from('email_reminders')
  .delete()
  .eq('record_id', id)
  .eq('user_id', userId);

if (reminderError) {
  console.error('Error deleting email reminder:', reminderError);
  // Don't return — still delete the health record
}

// ✅ 3. Now delete the health record
const { error: deleteError } = await supabase
  .from('health_records')
  .delete()
  .eq('id', id);

if (deleteError) {
  console.error('Delete error:', deleteError);
  return res.status(500).json({ error: 'Delete failed' });
}
```

Figure 6.80: Reminders deleted when health record deleted

The system also includes safeguards to maintain data integrity. When a health record is deleted, the associated reminder is automatically removed from the email_reminders table to avoid orphaned entries. Additionally, a daily cleanup function runs to identify and delete any reminders that no longer have a corresponding health record, ensuring the database remains consistent.

```
// Get user email
const { data: user, error: userError } = await supabaseAdmin.auth.admin.getUserById(reminder.user_id);
```

Figure 6.81: Email Address Accessed via Supabase Admin API

Security and privacy were prioritized throughout the implementation. The user's email address was only accessed via Supabase Admin API with proper authentication, and all API keys (including RESEND_API_KEY) were stored in environment variables using .env files. The entire process was asynchronous and non-blocking, ensuring that the main application flow was not affected by email operations.

This automated email reminder system exemplified a seamless integration of backend logic, database management, and third-party services. By combining real-time data processing, scheduled checks, and personalized communication, PawHub delivers a valuable, user-centric feature that promotes proactive pet care and enhances the overall user experience.

6.7 Web Scraping for Educational Content

The Education module in PawHub was powered by a dynamic web scraping system that ensured users have access to fresh, reliable, and expert-backed pet care information. The application integrated a weekly scraping process that fetched articles from trusted sources such as the American Kennel Club (AKC), ensuring that the knowledge base remains current and relevant. This approach not only reduces maintenance overhead but also enhanced the app's value by delivering up-to-date guidance on topics like dog breeds, puppy care, health, training, nutrition, travel, and fun facts.

```
// 2) Axios instance with a browser User-Agent
const axiosInstance = axios.create({
  headers: {
    'User-Agent':
      'Mozilla/5.0 (Windows NT 10.0; Win64; x64) ' +
      'AppleWebKit/537.36 (KHTML, like Gecko) ' +
      'Chrome/91.0.4472.124 Safari/537.36'
  }
});
```

Figure 6.82: Axios for HTTP requests

The scraping process was implemented using Node.js with the libraries Axios for HTTP requests and Cheerio for parsing HTML content. A dedicated script, scraper.js, was responsible for visiting predefined AKC article pages, extracting key information, and structuring it into a standardized format. For each article, the scraper collected the title, category, image URL, and original link.

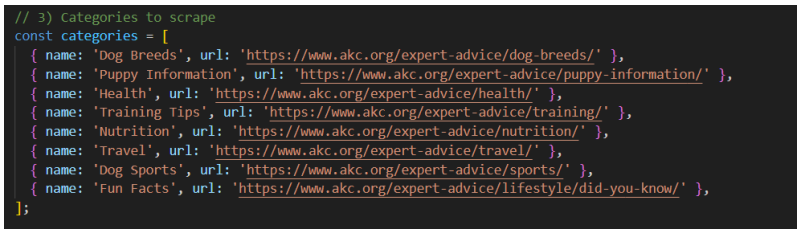
```
if (articles.length) {
  const { data, error } = await supabase
.from('articles')
.upsert(articles, { onConflict: ['link'] })
.select(); // forces data to return
```

Figure 6.83: Scraper performs a link-based upsert operation

To ensure content integrity and avoid duplicates, the scraper performed a link-based upsert operation when inserting data into the articles table in Supabase. If an article with the same link already exists, it is updated, otherwise, a new record is created. This mechanism guaranteed that only

unique articles were stored and that existing entries were refreshed if updated on the source website.

The scraping task run weekly, it was manually updated by the developer in the terminal. This timing was chosen to minimize server load and ensure fresh content was available at the start of the weekend, when users were more likely to engage with educational material. The GitHub Actions workflow could be implemented as future enhancement to trigger the scraper.js script in a Node.js environment, it can execute the scraping logic, and securely connects to Supabase using environment variables to insert the data.



```
// 3) Categories to scrape
const categories = [
  { name: 'Dog Breeds', url: 'https://www.akc.org/expert-advice/dog-breeds/' },
  { name: 'Puppy Information', url: 'https://www.akc.org/expert-advice/puppy-information/' },
  { name: 'Health', url: 'https://www.akc.org/expert-advice/health/' },
  { name: 'Training Tips', url: 'https://www.akc.org/expert-advice/training/' },
  { name: 'Nutrition', url: 'https://www.akc.org/expert-advice/nutrition/' },
  { name: 'Travel', url: 'https://www.akc.org/expert-advice/travel/' },
  { name: 'Dog Sports', url: 'https://www.akc.org/expert-advice/sports/' },
  { name: 'Fun Facts', url: 'https://www.akc.org/expert-advice/lifestyle/did-you-know/' },
];
```

Figure 6.84: Categorized based on the source URL

All scraped articles were categorized based on their source URL and content structure. Categories included Dog Breeds, Puppy Info, Health, Training, Nutrition, Travel, Sports, and Fun Facts, allowing users to easily navigate and filter content within the app. Each article card in the Education screen displayed the title, a thumbnail image, the category badge, and a “Read More” button that opened the original webpage in the device’s default browser. This design respects content ownership and licensing while providing seamless access to authoritative information.

```
// AKC specific parsing logic
if (cat.url.includes('akc.org')) {
$('.content-card').each((_, el) => {
  const title = $(el).find('.content-card__title').text().trim();
  const summary =
    $(el).find('.content-card__body p').text().trim() ||
    $(el).find('.content-card__dek').text().trim() ||
    '';
  const link = $(el).find('.content-card__title').attr('href');

  let image =
    $(el).find('img').attr('data-src') ||
    $(el).find('img').attr('src') ||
    '';

  if (image && image.startsWith('//')) {
    image = 'https:' + image;
  }

  const article = { title, summary, link, image, category: cat.name };
  // console.log('Parsed AKC article:', article);
  if (title && link) articles.push(article);
```

Figure 6.85: Script logs the error and Process other articles

Security and reliability were prioritized throughout the implementation. The scraper included error handling for network failures, rate limiting, and changes in page structure. If a source page was temporarily unavailable or the HTML format changes, the script logs the error and continues processing other articles, preventing a single failure from halting the entire operation. Additionally, all API keys and database credentials were stored in environment variables, ensuring sensitive data was never exposed in the codebase.

By combining web scraping, scheduled execution, and secure database integration, PawHub delivers a continuously updated educational experience that empowers pet owners with trustworthy, expert-backed insights without requiring manual updates or content creation.

6.8 Security Implementation

The PawHub application prioritized security across all layers of the system to protect user data, ensure privacy, and maintain the integrity of sensitive information such as pet health records and personal details. A multi-layered security approach has been implemented that spans the frontend, backend, database, and third-party integrations, ensuring that the application adhered to modern best practices in mobile and web security.

```
// === MIDDLEWARE: Authenticate User ===
const authenticate = async (req, res, next) => {
  const token = req.headers.authorization?.replace('Bearer ', '').trim();
  if (!token) return res.status(401).json({ error: 'Missing auth token' });

  try {
    const { data: { user }, error } = await supabase.auth.getUser(token);
    if (error || !user) return res.status(401).json({ error: 'Invalid token' });
    req.user = user;
    next();
  } catch (err) {
    console.error('Auth middleware error:', err);
    res.status(500).json({ error: 'Authentication failed' });
  }
};
```

Figure 6.86: Authentication Middleware and Supabase Auth

At the authentication level, PawHub leveraged Authentication middleware and Supabase Auth for secure user sign-up, login, and session management. This eliminated the need to handle raw passwords within the application logic, as all credential verification was performed securely by Supabase using industry-standard hashing and encryption.

```
// Save token to storage
await AsyncStorage.setItem('accessToken', session.access_token);
```

Figure 6.87: JWT (JSON Web Token) saved to storage

A JWT (JSON Web Token) was generated and saved in the device's AsyncStorage upon successful authentication, this token was then used for upcoming API queries. This token was validated on every request through a custom authenticate middleware on the backend, ensuring that only authorized users can access protected routes.

```
// === RATE LIMITING ===
const limiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 200,
  standardHeaders: true,
  legacyHeaders: false,
});

const authLimiter = rateLimit({
  windowMs: 15 * 60 * 1000,
  max: 20,
  message: { error: 'Too many attempts. Try again later.' },
  standardHeaders: true,
  legacyHeaders: false,
});

const chatLimiter = rateLimit({
  windowMs: 60 * 1000, // 1 minute
  max: 20, // 20 requests per minute for chat
  message: { error: 'Too many chat requests. Please slow down.' },
  standardHeaders: true,
  legacyHeaders: false,
});
```

Figure 6.88: Express Rate Limits

To prevent abuse and protect against brute-force attacks, rate limiting is enforced using express-rate-limit on key endpoints. The /auth/ routes were limited to 20 attempts per 15 minutes, preventing repeated login attempts. General API usage was capped at 200 requests per 15 minutes per IP, and chat-related endpoints were limited to 20 messages per minute to discourage spamming. These limits were applied globally and were transparently managed by the Express server, enhancing system resilience without impacting legitimate users.

```
// Fetch pets from backend
const fetchPets = async () => {
  try {
    setLoading(true);
    const token = await AsyncStorage.getItem('accessToken'); // Retrieve token from storage

    console.log('Fetched token:', token);
    if (!token) {
      console.log('No token found');
      return;
    }
    const response = await fetch(`${API_URL}/pets`, {
      headers: {
        Authorization: `Bearer ${token}`
      }
    });
```

Figure 6.89: HTTPS-style practices

All communication between the React Native frontend and Node.js backend was secured using HTTPS-style practiced via JWT-based authentication. Every API request included an Authorization: Bearer <token> header, which is validated before any data processing occurs. This ensured that even if an endpoint is exposed, it cannot be accessed without proper credentials. In order to mitigate typical online vulnerabilities like XSS and clickjacking, the backend also used Helmet.js to specify secure HTTP headers.

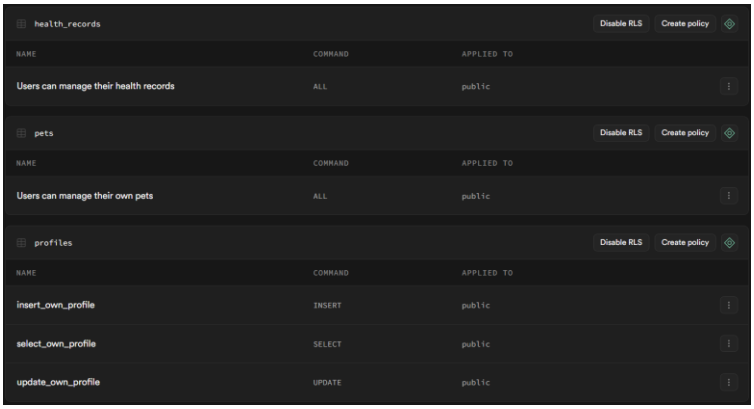


Figure 6.90: Row Level Security (RLS) policies in Supabase

The database layer was protected by Row Level Security (RLS) policies in Supabase, which enforced strict data isolation. Users can only access, modify, or delete records that belong to them. For example, a user cannot view another user's pet profiles, health records, chat history, or symptom diagnoses even if they know the record ID. These policies were defined directly in PostgreSQL and are enforced at the database level, providing a robust defense against unauthorized data access.

```
// === CORS (Restrict in production) ===
const allowedOrigins = [
  'http://localhost:3000',
  'https://pawhub.app',
  'pawhub://'
];

app.use(cors({
  origin: (origin, callback) => {
    if (!origin || allowedOrigins.includes(origin)) {
      callback(null, true);
    } else {
      callback(new Error('Not allowed by CORS'));
    }
  },
  credentials: true,
}));

// === MULTER CONFIG ===
const upload = multer({
  storage: multer.memoryStorage(),
  limits: { fileSize: 5 * 1024 * 1024 },
});
```

Figure 6.91: Multers and Cors Configuration

File uploads, such as profile picture changes, were handled securely using Multer middleware on the backend. When a user uploaded an image, it was temporarily stored, validated for type and size, and then uploaded to the Supabase Storage "avatars" bucket with appropriate access controls. Public URLs are generated for display in the app, but direct access to the storage is

restricted to authenticated users only. Default avatars were pre-uploaded and referenced via secure signed URLs, minimizing exposure.

```
import dotenv from 'dotenv';
dotenv.config();
```

(a)

```
// === ENV VALIDATION ===
if (!process.env.SUPABASE_URL) {
  throw new Error('Missing SUPABASE_URL in .env');
}
if (!process.env.SUPABASE_ANON_KEY) {
  throw new Error('Missing SUPABASE_ANON_KEY in .env');
}
if (!process.env.OPENROUTER_API_KEY) {
  throw new Error('Missing OPENROUTER_API_KEY in .env');
}
```

(b)

Figure 6.92(a)(b): Environment Variables

Sensitive configuration data, including API keys for OpenRouter, Resend, and Supabase service roles, were stored in .env files and loaded using the dotenv package. These files were excluded from version control via .gitignore, preventing accidental exposure of secrets. The backend validated the presence of all required environment variables at startup, ensuring the server cannot run without proper configuration.

```
// Email validation
if (!profile.email.trim()) {
  newErrors.email = 'Email is required';
} else if (!emailRegex.test(profile.email)) {
  newErrors.email = 'Please enter a valid email';
}

// Username validation
if (!profile.username.trim()) {
  newErrors.username = 'Username is required';
} else if (profile.username.length < 3) {
  newErrors.username = 'Username must be at least 3 characters';
} else if (profile.username.length > 30) {
  newErrors.username = 'Username cannot exceed 30 characters';
}

// Phone validation
if (profile.phone && !phoneRegex.test(profile.phone)) {
  newErrors.phone = 'Please enter a valid phone number (10-15 digits)';
}

// Password validation
if (profile.password) {
  if (profile.password.length < 8) {
    newErrors.password = 'Password must be at least 8 characters';
  } else if (!passwordRegex.test(profile.password)) {
    newErrors.password = 'Must include uppercase, lowercase, number, and special character';
  }

  // Confirm password validation
  if (profile.password !== profile.confirmPassword) {
    newErrors.confirmPassword = 'Passwords do not match';
  }
}
```

Figure 6.93: Input Validation

Besides, to stop injection attacks and corrupted data, input validation was done on both the frontend and backend. On the frontend, all user inputs were validated for registration, login, and profile updates. On the backend, all

incoming data was checked for type, format, and completeness before being processed or stored. For example, email fields were validated using regex patterns, phone numbers were checked for length, and password strength was enforced during profile updates.

```
// Get user email
const { data: user, error: userError } = await supabaseAdmin.auth.admin.getUserById(reminder.user_id);
```

Figure 6.94: Supabase Admin API

Additionally, the backend used the Supabase Admin API with a service role key to securely retrieve user emails for email reminders, ensuring that this sensitive operation is performed only in a trusted environment. The service role key was never exposed to the frontend and is strictly limited to server-side use.

This comprehensive security strategy ensured that PawHub remained a trustworthy and reliable platform for pet owners. By integrating secure authentication, rate limiting, encrypted storage, input validation, and environment isolation, the application protected user data at every level while delivering a seamless and intuitive experience.

6.9 Conclusion

The implementation of PawHub demonstrated a successful integration of modern mobile development, artificial intelligence, and secure backend architecture to deliver a comprehensive pet care solution. The system effectively combined user-centric design with intelligent features such as AI-powered symptom diagnosis, virtual pet assistance, and personalized health tracking to empower pet owners with timely, actionable insights. Each module has been carefully developed and tested, ensuring a seamless and intuitive user experience across all screens, from authentication and pet profile management to education and feedback submission.

The frontend, built with React Native, provided a responsive and visually appealing interface that maintained consistency across devices while offering smooth navigation and real-time interactions. The backend, powered by Node.js and Express.js, serves as a robust and secure gateway between the

client and external services, handling authentication, data processing, and API routing with efficiency and reliability. By leveraging Supabase as the primary database and authentication provider, the application ensured data integrity, scalability, and strong security through Row Level Security (RLS) policies that restrict user access to their own data.

Key advanced features such as the AI Chatbot and Symptom Diagnosis module showcased the effective use of OpenRouter to deliver context-aware responses personalized to each pet's profile and medical history. The Model Control & Prioritization (MCP) strategy ensured service continuity by automatically falling back to alternative AI models when needed, enhancing system reliability. Additionally, the automated email reminder system, powered by Resend, supports preventive care by notifying users of upcoming health events, while the weekly web scraper kept the Education module updated with fresh, expert-backed content from trusted sources like the American Kennel Club.

Security has been prioritized throughout the development process, with JWT-based authentication, rate limiting, input validation, secure file storage, and environment variable isolation working together to protect user data and prevent abuse. The application adhered to best practices in data handling and privacy, ensuring compliance with modern security standards.

Overall, the implementation of PawHub reflected a professional, well-structured approach that successfully bridges the gap between pet owners and virtual pet care assistance. This foundation not only met current functional requirements but also supported future enhancements such as push notifications, multi-language support, or integration with wearable pet devices. With its robust architecture, intelligent features, and user-focused design, PawHub stands as a scalable, real-world-ready mobile application that delivers practical value to pet owners while demonstrating technical excellence in full-stack development.

CHAPTER 7

TESTING

7.1 Introduction

The testing phase was a critical component in the development lifecycle of any software application, ensuring that the system functioned as intended, met user requirements, and maintained high standards of reliability, usability, and security. For the PawHub Application, testing strategy was implemented to validate all aspects of the system, from individual components to end-to-end user workflows. The test conducted includes unit testing, system usability testing, and user acceptance testing. The goal was to identify and resolve defects early and enhance user experience. Testing was conducted iteratively alongside development, following an Agile-inspired approach that allowed for continuous feedback and improvement. All tests were documented, and results were used to refine features, fix bugs, and optimize performance.

7.2 Test Execution

Test execution was carried out in a systematic manner across multiple environments to ensure compatibility, functionality, and performance consistency. The primary testing environment consisted of an Android Studio Emulator (Pixel 4 API 30) to validate real-world performance. The backend server was hosted locally at http://10.0.2.2:3000 to support emulator connectivity, while Supabase served as the cloud database and authentication provider. Each test cycle followed a predefined test plan that included test objectives, input data, expected outcomes, and pass/fail criteria. Functional testing was performed on all major modules: authentication, AI chatbot, symptom diagnosis, pet and health record management, profile editing, education, and feedback. Logs and error messages were monitored using console outputs and Supabase logs to trace issues. Test results were recorded in a structured format, and critical bugs were prioritized for immediate resolution. This disciplined execution ensured that PawHub met functional requirements and delivered a stable, user-friendly experience.

7.3 Unit Test

The unit testing phase focused on validating the core functionalities of the PawHub application to ensure reliable and error-free performance. Each module, including authentication, pet management, AI chatbot, symptom diagnosis, profile management, education, and feedback, was tested in isolation to verify correct behavior under various scenarios. The testing covered input validation, form handling, API integration, error states, and user interactions to confirm that individual components function as intended. This rigorous approach ensured the application's stability, usability, and readiness for further testing stages.

Table 7.1: Unit Test Case - Login

Test Case ID	TC-001	**Module Name**	Authentication Module	
Test Title	Login Screen			
Pre Condition	-			
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Valid credentials login	1. Enter registered email. 2. Enter correct password. 3. Tap "Submit".	User logs in successfully. JWT token stored in AsyncStorage. Navigate to Home screen.	Token stored. Navigation successful.	Pass
Empty email field	1. Leave email field empty. 2. Enter password. 3. Tap "Submit".	Error: "Email is required." Form submission blocked.	Error message appears. No API call made.	Pass
Empty password	1. Enter email. 2. Leave password	Error: "Password cannot be empty."	Error message	Pass

field	empty. 3. Tap “Submit”.	Form submission blocked.	appears. No API call made.	
Both fields empty	1. Leave both email and password empty. 2. Tap “Submit”.	Both errors appear: “Email is required” and “Password cannot be empty.”	Both error messages displayed.	Pass
Invalid email format	1. Enter “invalid-email” as email. 2. Enter valid password. 3. Tap “Submit”.	Backend returns error. Frontend shows “Invalid login credentials”.	Backend rejects. General error shown.	Pass
Correct email, wrong password	1. Enter correct email. 2. Enter incorrect password. 3. Tap “Submit”.	Backend returns error. App shows: “Invalid login credentials”.	Backend rejects. General error shown.	Pass
Unregistered email	1. Enter unregistered email. 2. Enter any password. 3. Tap “Submit”.	Backend returns error. App shows: “Invalid login credentials”.	General error appears. No navigation.	Pass
Loading state during login	1. Enter valid credentials. 2. Tap “Submit”. 3. Observe button.	Button shows Activity Indicator. Cannot be pressed again.	Spinner appears. Button disabled.	Pass
Network error (no internet)	1. Turn off Wi-Fi/data. 2. Enter credentials. 3. Tap “Submit”.	App shows: “Unable to connect. Please check your internet connection.”	Error message displayed.	Pass

		error.		
Backend server down	1. Stop Node.js server. 2. Attempt login.	Fetch fails. App shows: "Something went wrong. Please try again."	Error caught in catch block. Message shown.	Pass
Secure password input	1. Type in password field.	Characters are masked (****).	Password hidden during input.	Pass
Case-sensitive email handling	1. Enter "User@Example.com". 2. Submit with correct password.	Login succeeds. Email is handled case-insensitively by backend.	Login successful.	Pass
Forgot password link	1. Tap "Forgot password?" link.	Navigate to Forgot Password Screen.	Navigation occurs.	Pass
Sign up link	1. Tap "Don't have an account? Sign up".	Navigate to Register Screen.	Navigation occurs.	Pass
Rapid repeated login attempts	1. Submit invalid login 5 times quickly.	No client-side rate limiting, but backend blocks after 20 attempts (via express-rate-limit).	After 20 attempts, "Too many attempts, try again later" error received.	Pass
Input text clearing on re-entry	1. Enter email. 2. Navigate away. 3. Return to screen.	Fields are empty (unless auto-fill enabled).	Fields start empty.	Pass

Table 7.2: Unit Test Case - Register

Test Case ID	TC-002	**Module Name**	Authentication Module	
Test Title	Register Screen			
Pre Condition	-			
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Valid registration	1. Enter valid username, email, phone, password, and confirm password. 2. Tap “Submit”.	User is registered. Success alert appears. Redirect to Login screen.	Registration successful. Navigation occurs.	Pass
Duplicate Email	1. Enter existing email. 2. Fill other fields. 3. Tap “Register”.	Error: “An account with this email already exist” shown	Error shown, Registration blocked.	Pass
Empty username	1. Leave username empty. 2. Fill other fields. 3. Tap “Submit”.	Error: “Username is required.” Form submission blocked.	Error message displayed.	Pass
Invalid email format	1. Enter “invalid-email” as email. 2. Fill other fields. 3. Tap “Submit”.	Error: “Please enter a valid email address.”	Validation prevents submission.	Pass
Missing phone number	1. Leave phone field empty. 2. Fill other fields. 3. Tap “Submit”.	Error: “Phone number is required.”	Error message shown.	Pass
Invalid phone number	1. Enter “123” (less than 10 digits).	Error: “Enter a valid phone number (10–15	Validation enforced.	Pass

	2. Fill other fields. 3. Tap “Submit”.	digits).”		
Weak password (short)	1. Enter password: “pass1”. 2. Fill other fields. 3. Tap “Submit”.	Error: “Password must be at least 6 characters.”	Submission blocked.	Pass
Weak password (missing criteria)	1. Enter “password123” (no uppercase). 2. Fill other fields. 3. Tap “Submit”.	Error: “Password must include uppercase, lowercase, and a number.”	Validation triggered.	Pass
Password mismatch	1. Enter password: “Pass123”. 2. Confirm: “Pass12”. 3. Tap “Submit”.	Error: “Passwords do not match.”	Error message appears.	Pass
Network error (no internet)	1. Turn off Wi-Fi/data. 2. Attempt registration.	Error: “Unable to register. Please check your internet connection.”	General error message shown.	Pass
Backend server down	1. Stop Node.js server. 2. Attempt registration.	App shows: “Something went wrong. Please try again.”	Error caught and displayed.	Pass
Loading state during submission	1. Enter valid data. 2. Tap “Submit”. 3. Observe button.	Button shows Activity Indicator. Cannot be pressed again.	Spinner appears. Button disabled.	Pass
Navigate to Login screen	1. Tap “Already have an account? Login”.	Navigate to Login Screen.	Navigation occurs.	Pass

Table 7.3: Unit Test Case - Forgot

Test Case ID	TC-003	**Module Name**	Authentication Module	
Test Title	Forgot Screen			
Pre Condition	-			
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Valid email submission	1. Enter registered email. 2. Tap "Submit".	Success message: "Password reset link sent." Redirect to Login after 3 seconds.	Message shown. Navigation to Login occurs.	Pass
Empty email field	1. Leave email empty. 2. Tap "Submit".	Error: "Email is required." Form submission blocked.	Error message appears.	Pass
Invalid email format	1. Enter "invalid-email". 2. Tap "Submit".	Error: "Please enter a valid email address."	Validation prevents submission.	Pass
Unregistered email	1. Enter unregistered email. 2. Tap "Submit".	No error (security best practice). Generic success message shown.	App shows success to avoid exposing account status.	Pass
Loading state during submission	1. Enter valid email. 2. Tap "Submit". 3. Observe	Button shows Activity Indicator. Cannot be pressed again.	Spinner appears. Button disabled.	Pass

	button.			
Network error (no internet)	1. Turn off Wi-Fi/data. 2. Attempt submission.	Error: “Unable to send reset link. Please check your internet connection.”	General error message displayed.	Pass
Backend server down	1. Stop Node.js server. 2. Attempt reset.	Error: “Network error. Please try again.”	Error caught in catch block.	Pass
Navigate back to Login	1. Tap “Back to Login” link.	Navigate to Login Screen.	Navigation occurs.	Pass
Success message visibility	1. Submit valid email. 2. Observe feedback.	Green success message: “Password reset link sent to your email.”	Message displayed clearly.	Pass

Table 7.4: Unit Test Case - AI Chatbot

Test Case ID	TC-004	**Module Name**	AI Chatbot Module	
Test Title	AI Chatbot Screen			
Pre Condition	-			
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Send message with valid input	1. Select a pet. 2. Type “How do I train my puppy?” 3. Tap send.	AI responds with relevant advice. Message appears in chat.	Response received and displayed.	Pass
Empty	1. Leave input	No message sent. Input	Send button	Pass

message input	field empty. 2. Tap send.	field remains unchanged.	disabled. No action.	
Loading state during response	1. Send a message. 2. Wait for AI response.	Typing indicator (animated dots) appears. Send button disabled.	Indicator shown until response received.	Pass
Switch pet resets chat	1. Send a message. 2. Select a different pet.	Chat clears. New session starts.	Messages reset. Session ID regenerated.	Pass
Rate AI response (1–5 stars)	1. Receive AI response. 2. Tap stars to rate (e.g., 5).	Rating saved in chat_history table. Visual feedback shown.	Rating updated in Supabase.	Pass
Select different AI model	1. Tap model selector. 2. Choose "Mistral-7B-Instruct". 3. Send message.	Request sent to selected model. Response received.	Correct model used. Response generated.	Pass
Network error during message send	1. Turn off internet. 2. Send message.	Friendly fallback: "Check your connection. For urgent concerns, contact your vet."	Error message displayed.	Pass
Backend server down	1. Stop Node.js server. 2. Send message.	App shows fallback message. No crash.	Graceful error handling.	Pass
Clear chat confirmation	1. Tap "Clear Chat".	Chat history cleared. New session starts.	Messages removed.	Pass

	2. Confirm in alert.		Session reset.	
Health records badge visibility	1. Add health records for selected pet. 2. Open chat.	Badge shows: "X health records loaded".	Badge appears with correct count.	Pass
Model selector displays correct names	1. Tap model selector.	Modal shows: Microsoft MAI DS R1, Mistral-7B-Instruct, Horizon Alpha.	All models listed with context sizes.	Pass
User avatar displays correctly	1. User has custom profile picture. 2. Send message.	Avatar appears in user message bubble.	Image loaded from Supabase URL.	Pass
Bot avatar displays pet image	1. Pet has profile picture. 2. Receive AI response.	AI message shows pet's avatar.	Correct image displayed.	Pass

Table 7.5: Unit Test Case - Symptom Diagnosis

Test Case ID	TC-005	**Module Name**	Symptom Diagnosis Module	
Test Title	Symptom Diagnosis Screen			
Pre Condition	-			
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Submit symptoms	1. Select a pet. 2. Enter	AI returns structured	Response received and	Pass

with valid input	"vomiting, lethargic" in symptoms field. 3. Tap "Analyze Symptoms".	diagnosis with severity, recommendation, possible causes, and additional notes.	displayed correctly.	
Empty symptoms field	1. Leave symptoms field empty. 2. Tap "Analyze Symptoms".	Error: "Please describe your pet's symptoms." No API call made.	Validation triggered. Form blocked.	Pass
Loading state during analysis	1. Enter valid symptoms. 2. Tap "Analyze Symptoms". 3. Observe button.	Button shows loading indicator with "Analyzing symptoms…". Disabled during processing.	Spinner with words appears. Button disabled.	Pass
Severity badge displays correctly	1. Submit symptoms indicating emergency (e.g., "seizures"). 2. View result.	Red badge: "High Priority". Color-coded based on severity.	Correct color and label displayed.	Pass
View past assessments	1. Tap "Symptom History" button. 2. View list of past diagnoses.	Modal shows all previous assessments with date, pet, and severity.	History loaded from symptom_history table.	Pass

Delete past assessment	1. Open Symptom History. 2. Tap delete on an entry. 3. Confirm.	Entry removed from list and database.	Record deleted in Supabase.	Pass
Network error during diagnosis	1. Turn off internet. 2. Submit symptoms.	Friendly fallback: “Unable to get diagnosis. Please try again later.”	Error message displayed. Retry button shown.	Pass
Backend server down	1. Stop Node.js server. 2. Submit symptoms.	Friendly fallback: “Unable to get diagnosis. Please try again later.”	Error message displayed. Retry button shown.	Pass
AI response parsing (structured output)	1. Receive AI response. 2. View diagnosis card.	Response parsed into sections: Diagnosis, Recommendation, Possible Causes, Additional Notes.	Content correctly extracted and formatted.	Pass
Empty history state	1. No past assessments. 2. Tap “Symptom History”.	Modal shows: “No assessments yet.”	Empty state displayed correctly.	Pass
Disclaimer modal	1. Open Symptom Diagnosis screen and press info icon.	Modal appears: “This is for informational purposes only...”	Disclaimer shown.	Pass

Table 7.6: Unit Test Case - Pet Management

Test Case ID	TC-006	**Module Name**	Pet Management Module	
Test Title	Pet Management Screen			
Pre Condition	-			
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Add new pet with valid data	1. Tap "+" button. 2. Enter valid name, breed, birthday, weight. 3. Upload photo (optional). 4. Tap "Save".	Pet appears in list. Data saved in pets table. Success toast shown.	Pet added successfully.	Pass
Empty pet name validation	1. Tap "Add Pet". 2. Leave name empty. 3. Tap "Save".	Error: "Pet name is required." Form submission blocked.	Validation triggered.	Pass
Future birthday validation	1. Set birthday to next year. 2. Tap "Save".	Error: "Date cannot be in the future."	Future date submission blocked.	Pass
Invalid weight entry	1. Enter negative or non-numeric weight. 2. Tap "Save".	Error: "Weight must be a positive number."	Validation prevents save.	Pass
Invalid height entry (non-numeric)	1. Enter "xyz" as height. 2. Tap "Save".	Error: "Must be a valid number."	Prevents submission.	Pass
Edit existing pet profile	1. Tap "Edit" on a pet. 2. Change breed	Updated info reflected in list and database.	Changes saved in Supabase.	Pass

	or weight. 3. Tap “Save”.			
Delete a pet	1. Tap “Delete” on a pet. 2. Confirm in alert.	Pet removed from list. All associated health records deleted.	Record and related data removed.	Pass
Cancel delete action	1. Tap “Delete”. 2. Tap “Cancel” in alert.	No deletion occurs.	Pet remains in list.	Pass
Image upload from gallery	1. Tap image placeholder. 2. Choose “Gallery”. 3. Select image.	Image displayed in pet card. URL saved in database.	Photo uploaded to Supabase Storage.	Pass
Image upload from camera	1. Tap image placeholder. 2. Choose “Camera”. 3. Take photo.	Photo appears in pet profile.	Image saved and displayed.	Pass
Select default avatar	1. Tap image placeholder. 2. Choose a default pet icon (dog, cat, etc.).	Selected avatar displayed.	Default image URL saved.	Pass
Add health record with valid data	1. Select a pet. 2. Tap “Add Record”. 3. Enter valid name, date, cause. 4. Tap “Save”.	Record appears in list. Saved in health_records table.	Record created successfully.	Pass
Empty health record name	1. Leave record name empty. 2. Tap “Save”.	Error: “Record name is required.”	Form blocked.	Pass

Empty health record date	1. Leave record date empty. 2. Tap “Save”.	Error: “Date is required.”	Form blocked.	Pass
Future date with reminder enabled	1. Set date to next week. 2. Enable “Email Reminder”. 3. Save.	Entry created in email_reminders table.	Reminder scheduled successfully.	Pass
View health records list	1. Select a pet with records. 2. View list.	Records displayed in chronological order.	Correct sorting applied.	Pass
Edit existing health record	1. Tap “Edit” on a pet’s health record. 2. Change health record name. 3. Tap “Save”.	Updated info reflected in list and database.	Changes saved in Supabase.	Pass
Delete health record	1. Swipe or tap delete on a record. 2. Confirm.	Record removed from list and database.	Deletion confirmed in Supabase.	Pass
Network error during pet save	1. Turn off internet. 2. Try to save.	Error: “Something went wrong. Please try again.”	Friendly error message shown.	Pass
Backend server down	1. Stop Node.js server. 2. Attempt to save.	App shows error: “Could not add pet. Please try again”.	Graceful error handling.	Pass

Table 7.7: Unit Test Case - Profile Management

Test Case ID	TC-007	**Module Name**	Profile Management Module	
Test Title	Profile Management Screen			
Pre Condition	-			
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Load profile on screen open	1. Navigate to Profile screen. 2. Wait for data to load.	Email, username, phone, and avatar are displayed.	Data loaded from backend and shown.	Pass
Edit profile information	1. Tap “Edit”. 2. Change username. 3. Tap “Save Changes”.	Updated info saved in profiles table. Success toast shown.	Changes reflected in database.	Pass
Save with valid password change	1. Enter new strong password. 2. Confirm password. 3. Save.	Password updated in Supabase Auth. No error.	Login still works with new password.	Pass
Password mismatch validation	1. Enter new password. 2. Confirm with different text. 3. Tap “Save”.	Error: “Passwords do not match.” Form blocked.	Validation prevents submission.	Pass
Weak password validation	1. Enter “pass123” as password. 2. Save.	Error: “Must include uppercase, lowercase, number, and symbol.”	Submission blocked.	Pass
Empty email	1. Leave email	Error: “Email is	Form	Pass

field	empty. 2. Save.	required.”	validation triggered.	
Invalid email format	1. Enter “invalid-email”. 2. Save.	Error: “Please enter a valid email”.	Prevents invalid submission.	Pass
Short username validation	1. Enter “ab” as username. 2. Save.	Error: “Username must be at least 3 characters.”	Validation enforced.	Pass
Long username validation	1. Enter 31+ character username. 2. Save.	Error: “Username cannot exceed 30 characters.”	Input blocked.	Pass
Invalid phone number	1. Enter “123” as phone. 2. Save.	Error: “Enter a valid phone number (10–15 digits).”	Prevents save.	Pass
Upload profile photo from gallery	1. Tap avatar. 2. Choose “Gallery”. 3. Select image.	Image uploaded to Supabase Storage. Avatar updated.	New image appears in profile.	Pass
Take photo using camera	1. Tap avatar. 2. Choose “Camera”. 3. Take photo.	Photo captured, cropped, and uploaded.	Avatar updated with new photo.	Pass
Select default avatar	1. Tap avatar. 2. Choose a default user image.	Selected avatar displayed. URL saved in database.	Default image shown.	Pass
Cancel image picker	1. Tap avatar. 2. Tap outside modal.	Modal closes. No changes made.	Profile remains unchanged.	Pass
Log out	1. Tap “Log Out”.	Session cleared. Redirect to Login	User logged out	Pass

		screen.	successfully.	
Delete account confirmation	1. Tap "Delete Account". 2. Tap "Cancel".	No deletion. Back to profile.	Account remains active.	Pass
Delete account (confirm)	1. Tap "Delete Account". 2. Tap "Delete".	Account deleted. Redirect to Login.	Data removed from Supabase.	Pass
Loading state during save	1. Edit profile. 2. Tap "Save Changes". 3. Observe button.	Button shows loading spinner. Disabled during request.	Prevents duplicate submission.	Pass
Network error during save	1. Turn off internet. 2. Try to save changes.	Error: "Something went wrong. Please try again."	Friendly error message shown.	Pass
Backend server down	1. Stop Node.js server. 2. Attempt to save.	App shows error. No crash.	Graceful error handling.	Pass

Table 7.8: Unit Test Case - Education

Test Case ID	TC-008	**Module Name**	Education Module	
Test Title	Education Screen			
Pre Condition	-			
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Load articles on	1. Navigate to Education	Articles fetched from /articles endpoint and	List populated	Pass

screen open	screen. 2. Wait for data to load.	displayed.	with content.	
Search articles by title	1. Type “small dog” in search bar. 2. Observe results.	Only articles with matching title appear.	Filtered list shown.	Pass
Clear search query	1. Enter text. 2. Tap “X” button.	Search field clears. Full article list restored.	All articles reappear.	Pass
Select category filter (e.g., Health)	1. Tap “Health” tab. 2. Observe list.	Only articles under “Health” category are shown.	Filtering applied correctly.	Pass
Switch between categories	1. Tap “Nutrition”. 2. Then tap “Travel”.	Article list updates to match selected category.	Content changes dynamically.	Pass
View article details	1. Tap on an article card.	Opens external browser via Linking.openURL() using the article’s link.	Original webpage opens in browser.	Pass
Empty search results	1. Search for “xyz123” (no match).	Show: “No results for ‘xyz123’“.	Empty state displayed.	Pass
Default view (All category)	1. Open screen or select “All”.	Show all articles without filtering.	Full list displayed.	Pass
Image placeholder for missing image	1. Article has no image. 2. View card.	Display paw icon placeholder.	Placeholder shown correctly.	Pass

Read More button functionality	1. Tap “Read More” on a card.	Open article link in browser. Same as tapping the card.	Browser opens with correct URL.	Pass
Scrollable category tabs	1. Scroll horizontally through category tabs.	Tabs move smoothly. All categories accessible.	Horizontal scroll works.	Pass
Network error handling	1. Turn off internet. 2. Open Education screen.	Show error: “Could not load articles. Please try again.”	Alert shown. Empty state with retry.	Pass
Backend server down	1. Stop Node.js server. 2. Open screen.	App shows alert and empty state with refresh option.	Graceful error handling.	Pass

Table 7.9: Unit Test Case - Feedback

Test Case ID	TC-009	**Module Name**	Feedback Module	
Test Title	Feedback Screen			
Pre Condition	-			
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Submit feedback with valid input	1. Select 4 stars. 2. Enter “Great app for	Success toast: “Feedback submitted successfully”.	Feedback saved in feedback table with	Pass

	pet care". 3. Tap "Submit Feedback".		success toast.	
Empty feedback text validation	1. Leave text field empty. 2. Tap "Submit Feedback".	Error: "Please share your experience before submitting". Form blocked.	Validation prevents submission.	Pass
Rating selection (1–5 stars)	1. Tap 1 star. 2. Tap 5 stars.	Rating updates visually and in state.	Correct star count reflected.	Pass
Loading state during submission	1. Enter valid feedback. 2. Tap "Submit". 3. Observe button.	Button shows "SUBMITTING..." and disables.	Prevents duplicate submission.	Pass
View past feedbacks	1. Tap "View Past Feedbacks".	Modal opens showing all submitted feedback.	Data loaded from /feedback endpoint.	Pass
Past feedbacks loading state	1. Open history modal. 2. Wait for data.	Show: "Loading your feedback...".	Spinner appears during fetch.	Pass
Empty feedback history	1. No feedback submitted. 2. Open history modal.	Show: "No feedback submitted yet".	Empty state displayed.	Pass
Close history	1. Open modal.	Modal closes. Back to feedback form.	Navigation works	Pass

modal	2. Tap “X” or back button.		smoothly.	
Success notification visibility	1. Submit feedback. 2. Observe top of screen.	Toast slides in: “Success! Feedback submitted successfully”. Disappears after 3 seconds.	Notification shown and auto-dismissed.	Pass
Network error during submission	1. Turn off internet. 2. Try to submit.	Alert: “Could not submit feedback. Please try again.”	Error handled gracefully.	Pass
Backend server down	1. Stop Node.js server. 2. Attempt to submit.	App shows error alert. No crash.	Graceful error handling.	Pass

7.4 System Usability Scale (SUS) Test

System usability testing was carried out in order to assess the PawHub application's general usability, interface design, and simplicity of navigation. A group of ten pet owners, were invited to interact with the app in a controlled environment. Each participant was asked to complete the standardized System Usability Scale (SUS) questionnaire, a trustworthy and often used instrument for evaluating software system's perceived usability. The SUS consists of 10 Likert-scale statements designed to assess learnability, efficiency, and user confidence.

	The System Usability Scale Standard Version	Strongly Disagree					Strongly Agree
			1	2	3	4	5
1	I think that I would like to use this system frequently.		O	O	O	O	O
2	I found the system unnecessarily complex.		O	O	O	O	O
3	I thought the system was easy to use.		O	O	O	O	O
4	I think that I would need the support of a technical person to be able to use this system.		O	O	O	O	O
5	I found the various functions in this system were well integrated.		O	O	O	O	O
6	I thought there was too much inconsistency in this system.		O	O	O	O	O
7	I would imagine that most people would learn to use this system very quickly.		O	O	O	O	O
8	I found the system very awkward to use.		O	O	O	O	O
9	I felt very confident using the system.		O	O	O	O	O
10	I needed to learn a lot of things before I could get going with this system.		O	O	O	O	O

Figure 7.1: Standard SUS Test Questions (Item Benchmarks for the System Usability ScaleJUS, no date)

A globally recognized and proven technique for evaluating the general usability of software systems, the System Usability Scale (SUS) consists of 10 standardized questions, as shown in Figure 7.1. These questions were developed by John Brooke in 1986 and are designed to evaluate key aspects of user experience, including learnability, efficiency, ease of use, and user confidence (Item Benchmarks for the System Usability ScaleJUS, no date). The SUS uses a 5-point Likert scale for each statement, allowing for quantitative measurement of usability.

SUS Score	Grade	Adjective Rating
> 80.3	A	Excellent
68 – 80.3	B	Good
68	C	Okay
51 – 68	D	Poor
< 51	F	Awful

Figure 7.2: SUS Grading Table (Shei, 2023)

This benchmark table categorized SUS scores into descriptive adjective ratings, allowing for intuitive interpretation of the results. An average score of 68 is considered the industry average. Scores above 80.3 are classified as "Excellent", indicating a highly usable and satisfying user experience. The grading scale provides a clear framework for evaluating PawHub's performance against established usability standards.

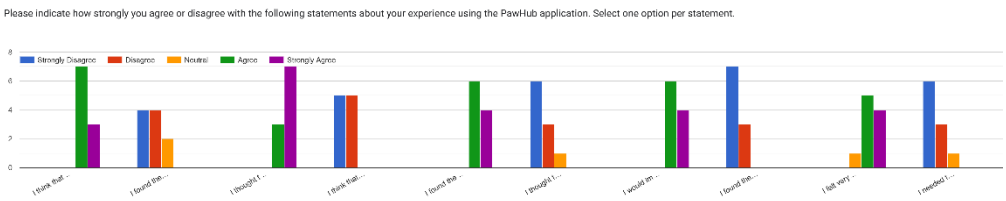


Figure 7.3: SUS Survey Response Chart

The visual representation of user responses highlights consistent agreement with positive usability indicators. Most participants strongly agreed with statements such as "I thought the app was easy to use" and "I felt very confident using the app." Even-numbered reverse-scored items (e.g., "I found the app unnecessarily complex") received low ratings, indicating users disagreed with negative usability claims. This pattern reflects a generally positive user experience across all tested features

Table 7.10: SUS Survey

Participants	**Scores for each Question**										**Total**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
1	4	1	5	1	4	1	5	1	5	2	92.50
2	5	1	5	1	5	1	5	1	5	1	100.00
3	4	2	4	2	4	2	4	1	3	3	72.50
4	4	2	5	2	5	2	5	1	5	1	90.00
5	4	3	4	2	4	3	4	2	4	2	65.00
6	5	1	5	1	5	1	4	1	4	1	95.00
7	4	3	4	2	4	2	4	2	4	2	72.50
8	4	2	5	1	4	1	4	2	5	1	87.50
9	5	2	5	1	4	1	4	1	4	1	90.00
10	4	2	3	2	4	2	3	2	3	2	67.50
Average Sus Score											83.25
Grade	A					**Adjective Rating**					Excellent

The individual SUS scores were calculated using the official formula: for odd-numbered items, (Score - 1) x 2.5, for even-numbered items, (5 - Score) x 2.5. The total for each participant was summed to produce a final score out of 100. As shown in the table, most participants scored between 87.5 and 100, indicating a highly positive perception of the app's usability.

The average SUS score of 83.25 places PawHub firmly in the "Excellent" category according to standard benchmarks. This result reflects a highly usable, intuitive, and efficient application. Users were able to complete tasks with confidence, found the interface easy to navigate, and reported minimal frustration during interaction.

Overall, the SUS test results confirm that PawHub delivers a user-centered, accessible, and highly satisfying experience. The app successfully balances advanced functionality with simplicity, making it appropriate for a variety of users, including those who are not as tech-savvy. These findings validate the effectiveness of the design decisions and iterative testing process employed during development.

7.5 User Acceptance Test

Five users participate in the User Acceptance Testing (UAT) portion of this project. Users were provided with realistic scenarios in order to evaluate the application's functionality, usability, and reliability under real-world conditions. The testing focused on core modules including authentication, pet management, health records, AI-powered features, education, feedback, and profile management. Each participant was asked to complete a series of tasks while providing feedback on their experience. The structured test template below outlines the UAT findings. Detailed observations and user comments were documented in Appendix C: User Acceptance Test Results.

Table 7.11: User Acceptance Testing Template for User

PawHub Application				
Test Module	**Test Case ID**	**Test Scenario**	**Status**	**Comment**
Register	UAT-001	Able to register a new account with valid username, email, phone, and password.	Pass	
Login	UAT-002	Able to log in with valid credentials.	Pass	
Forgot Password	UAT-003	Able to reset password using registered email.	Pass	
Pet Management	UAT-004	Able to add a new pet with photo, name, breed, birthday, weight, and height.	Pass	
	UAT-005	Able to edit existing pet profile.	Pass	
	UAT-006	Able to delete a pet and confirm associated records are removed.	Pass	
Health Records	UAT-007	Able to add a health record with future date and enable	Pass	

		email reminder.		
	UAT-008	Able to receive email reminder 24 hours before scheduled event.	Pass	
	UAT-009	Able to edit existing health records.	Pass	
	UAT-010	Able to delete a health record and confirm associated email reminders are removed.	Pass	
AI Chatbot	UAT-011	Able to send message to AI chatbot and receive relevant response.	Pass	
	UAT-012	Able to switch between AI models.	Pass	
	UAT-013	Able to switch between Pets.	Pass	
	UAT-014	Able to rate AI message with 1–5 stars.	Pass	
Symptom Diagnosis	UAT-015	Able to submit symptoms and receive AI-generated diagnosis with severity level.	Pass	
	UAT-016	Able to view and delete past symptom assessments.	Pass	
Education	UAT-017	Able to search articles and filter by category.	Pass	
	UAT-018	Able to tap article to open original article webpage.	Pass	
Feedback	UAT-019	Able to submit feedback with rating and comment.	Pass	
	UAT-020	Able to view past feedback submissions.	Pass	
Profile	UAT-021	Able to update username,	Pass	

Management		phone, and password.		
	UAT-022	Able to upload profile picture from gallery or camera.	Pass	
	UAT-023	Able to select default avatar.	Pass	
	UAT-024	Able to log out.	Pass	
	UAT-025	Able to delete account.	Pass	

All 25 test cases passed across 5 participants, confirming that PawHub is stable, user-friendly, and ready for deployment. The UAT results demonstrate strong alignment between the app's design and user needs, particularly in AI assistance, health tracking, and ease of use. Detailed qualitative feedback from users is included in Appendix C, supporting continuous improvement and future enhancements.

CHAPTER 8

CONCLUSION AND RECOMMENDATION

8.1 Conclusion

The PawHub application has been successfully developed as a comprehensive, AI-powered mobile application created to tackle the main issues pet owners encounter while trying to manage the health and wellbeing of their animals. The system integrates artificial intelligence, cloud-based data storage, and user-centered design to deliver a seamless and intelligent pet care experience.

By combining an AI chatbot, symptom diagnosis tool, digital health record management, and curated educational content, PawHub fills a critical gap in the current pet care technology landscape. Unlike existing applications that offer fragmented or isolated features, PawHub provides a unified platform where users can access real-time AI support, track medical history, receive timely health reminders, and obtain expert-backed information all within a single, intuitive interface.

The implementation leverages a React Native frontend, a secure Node.js backend, and Supabase as the unified database and authentication provider. Key features including the AI Chatbot, Symptom Diagnosis, Pet and Health Record Management, and Educational Content Delivery have been rigorously tested and validated through unit testing, system usability testing, and user acceptance testing, demonstrating high functionality, reliability, and user satisfaction.

The integration of AI capabilities via OpenRouter.ai enables context-aware interactions that are personalized to each pet's profile, significantly enhancing the app's value as a virtual pet care assistant. The automated email reminder system ensures timely notifications for vaccinations and check-ups, promoting responsible pet ownership. Additionally, the web scraping mechanism for educational content ensures that users receive up-to-date, expert-backed information from trusted sources such as the American Kennel Club (AKC).

Security measures such as JWT-based authentication, Row Level Security (RLS), input validation, and environment variable isolation ensure the protection of sensitive user and pet data. Overall, PawHub stands as a robust, scalable, and user-centered mobile application that effectively bridges the gap between pet owners and virtual veterinary support, providing a holistic, all-in-one solution for managing multiple pets and addressing their everyday health and care concerns.

8.2 Achievement of Objectives

All project objectives have been successfully achieved, demonstrating the effectiveness and completeness of the PawHub application.

The first objective, to develop an AI-powered chatbot with support for multiple AI models was fulfilled by integrating OpenRouter.ai, enabling users to interact with models such as Microsoft MAI DS R1, Mistral-7B-Instruct, and Horizon Alpha. This allows personalized, context-aware responses to pet care inquiries, enhancing the reliability and depth of assistance.

The second objective, to implement an AI-driven symptom diagnosis tool was accomplished through a structured analysis of user-reported symptoms, delivering AI-generated assessments with severity levels and recommendations, helping users determine the urgency of veterinary visits.

The third objective, to create a structured digital health record management system was realized with the implementation of Supabase powered CRUD operations for pet profiles and medical records, including vaccination logs, treatment history, and email reminders for upcoming events.

Finally, the fourth objective, to offer educational resources via web scraping was achieved by the collection of expert-backed articles from AKC, ensuring users have access to credible, regularly updated pet care knowledge. These achievements collectively confirm that PawHub meets its intended purpose as a holistic pet care assistant.

8.3 Limitation & Recommendations

Even though the objectives were effectively met throughout the project, several limitations were found within the application during the development and testing phases. These limitations are shown in the table below along with the recommendations for future revisions to optimize the performance of the application.

Table 8.1: Limitations and Recommendations

No.	Limitations	Recommendations
1	No offline capabilities, the app requires an active internet connection for core features such as AI chatbot, symptom diagnosis, and article loading. Users cannot access or update pet records when offline.	Implement offline data storage using SQLite or AsyncStorage to allow users to view pet profiles, add/edit health records, and draft messages while offline. Sync data when the connection is restored.
2	Web scraping is done manually, articles are not automatically updated as GitHub Actions were not implemented. This reduces the timeliness and scalability of the education module.	Integrate GitHub Actions or a cloud-based scheduler (e.g., Supabase Cron, AWS Lambda) to automate the web scraping process weekly for fresh, up-to-date content.
3	Limited source of educational content. Currently, articles are only scraped from the American Kennel Club (AKC). This restricts the diversity and coverage of pet care topics.	Expand the web scraper to include additional reputable sources such as ASPCA, PetMD, and AKC's international counterparts to provide broader, more diverse content.
4	No multi-language support. Because the app is presently only available in English, users who do not speak English may find it difficult to use.	Add multiple language support using an internationalization (i18n) library to include common languages such as Spanish, Mandarin, or Bahasa Malaysia,

		improving global accessibility.
5	No push notifications. The app relies solely on email reminders for upcoming health events, which may be overlooked or marked as spam.	Integrate Firebase Cloud Messaging (FCM) to deliver real-time push notifications for reminders, AI responses, and system updates, improving user engagement and reliability.

REFERENCES

Agile software development: everything you need to know (2024) www.nexapp.ca. Available at: https://www.nexapp.ca/en/blog/agile-software-development.

Appleby, R.B. and Basran, P.S. (2022) 'Artificial intelligence in veterinary medicine', Journal of the American Veterinary Medical Association, 260(8), pp. 1–6. Available at: https://doi.org/10.2460/javma.22.03.0093 .

asierr.dev (2024) 5 Supabase Features That Make It the Best Backend for Startups, Medium. Available at: https://medium.com/@asierr/5-supabase-features-that-make-it-the-best-backend-for-startups-b0c8340b7200 (Accessed: 17 April 2025).

Atlassian (2024) Waterfall Methodology for Project Management, Atlassian. Available at: https://www.atlassian.com/agile/project-management/waterfall-methodology.

Build Documentation | Firebase Documentation (no date) Firebase. Available at: https://firebase.google.com/docs/build.

Figma Design – Figma Learn - Help Center (2024) Figma.com. Available at: https://help.figma.com/hc/en-us/categories/360002042553.

Flutter documentation (no date) docs.flutter.dev. Available at: https://docs.flutter.dev/?_gl=1.

GitHub (2024) GitHub.com Help Documentation, docs.github.com. Available at: https://docs.github.com/en.

GitHub Logo Download - SVG - All Vector Logo (2016) AllVectorLogo. Available at: https://allvectorlogo.com/github-logo/ (Accessed: 22 April 2025).

Guo, D. et al. (2024) DeepSeek-Coder: When the Large Language Model Meets Programming -- The Rise of Code Intelligence, arXiv.org. Available at: https://doi.org/10.48550/arXiv.2401.14196.

Hill, P. (2024) Visual Studio Code 1.94 launched with big startup speed improvements, Neowin. Available at: https://www.neowin.net/news/visual-studio-code-194-launched-with-big-startup-speed-improvements/.

Interino, J. (2022) Figma for your Design Portfolio, Part 2: How to Create a Design Portfolio in Figma, Atomic Spin. Available at: https://spin.atomicobject.com/building-your-portfolio-figma/ (Accessed: 23 April 2025).

Introduction · React Native (no date) reactnative.dev. Available at: https://reactnative.dev/docs/getting-started.

Introduction · React Native (no date) reactnative.dev. Available at: https://reactnative.dev/docs/getting-started.

Item Benchmarks for the System Usability ScaleJUS (no date) uxpajournal.org. Available at: https://uxpajournal.org/item-benchmarks-system-usability-scale-sus/.

Jokar, M., Arman Abdous and Vahid Rahmanian (2024) 'AI chatbots in pet health care: Opportunities and challenges for owners', Veterinary medicine and science, 10(3). Available at: https://doi.org/10.1002/vms3.1464 .

Kaleel, S. B. and Harishankar, S. (2013) 'Applying Agile Methodology in Mobile Software Engineering: Android Application Development and its

Challenges'. Toronto Metropolitan University. doi: 10.32920/ryerson.14637270.v2.
Käpyaho, M. and Kauppinen, M. (2015) Agile requirements engineering with prototyping: A case study, IEEE Xplore. Available at: https://doi.org/10.1109/RE.2015.7320450.
Kissflow (2022) Rapid Application Development (RAD) | Definition, Steps & Full Guide, kissflow.com. Available at: https://kissflow.com/application-development/rad/rapid-application-development/.
Lai, N. et al. (2021) 'Pet owners' online information searches and the perceived effects on interactions and relationships with their veterinarians', Veterinary Evidence, 6(1). Available at: https://doi.org/10.18849/ve.v6i1.345 .
LLC, H.B. (2023) PetVet AI 24/7 Pet Health Care (1.0.8), [Mobile app]. Available at: https://play.google.com/store/apps/datasafety?id=ai.petvet.app&hl=en (Accessed: 20 March 2025)
Ltd, 11 Pets (2015) 11pets Pet care (6.003.003), [Mobile app]. Available at: https://play.google.com/store/apps/details?id=com.m11pets.elevenpets&hl=en (Accessed: 20 March 2025).
Lyssa AS (2024) PetVitality Pet Health Tracker (1.1.3), [Mobile app]. Available at: https://play.google.com/store/apps/details?id=com.lyssa.petvitality&hl=en (Accessed: 20 March 2025).
Managing Emails - Resend (2025) Resend.com. Resend. Available at: https://resend.com/docs/dashboard/emails/introduction (Accessed: 12 September 2025).
Mehra, A. (2025) 'Utilizing Machine Learning for Developing a Pet Health Monitoring System', Journal of Integrated Engineering Sciences (JIES), 1(1), pp. 37–43. Available at: https://journals.academicsp.com/index.php/jies/article/view/5 (Accessed: 20 March 2025).
Motion (2023) Understanding the Waterfall Methodology: A Sequential Approach to Project Management, www.usemotion.com. Available at: https://www.usemotion.com/blog/waterfall-methodology.
Najjar, A. (2023) How to install Emulator on Android Studio - Abdalqader Najjar - Medium, Medium. Available at: https://medium.com/@abdalqader27.najjar/how-to-install-emulator-on-android-studio-95eb101e604b (Accessed: 22 April 2025).
Niemiec, R. et al. (2024) 'Veterinary and pet owner perspectives on addressing access to veterinary care and workforce challenges', Frontiers in Veterinary Science, 11. Available at: https://doi.org/10.3389/fvets.2024.1419295 .
Node.js Development Services Company | Hire Node.js Developers (2016) Angularminds.com. Available at: https://www.angularminds.com/nodejs-development-company (Accessed: 12 September 2025).
Okoone (2025) Okoone.com. Available at: https://www.okoone.com/technologies/mobile/react-native/ (Accessed: 17 April 2025).
OpenAI (2023) GPT-4, Openai.com. Available at: https://openai.com/index/gpt-4-research/.
OpenJS Foundation (2017) Express - Node.js web application framework, Expressjs.com. Available at: https://expressjs.com/.

OpenRouter Logo PNG Vector (SVG) Free Download (2025) Seeklogo. Available at: https://seeklogo.com/vector-logo/611674/openrouter (Accessed: 27 April 2025).
PET, A.F. (2023) TTcare: Keep Your Pet Healthy (2.5.0), [Mobile app]. Available at: https://play.google.com/store/apps/details?id=com.ttcare.pet&hl=en (Accessed: 20 March 2025).
PLIT, A. (2021) Veterinary Medical Records and the Importance of Documentation, EquiManagement. Available at: https://equimanagement.com/business-development/legal/veterinary-medical-records-and-the-importance-of-documentation/ (Accessed: 19 March 2025).
Postman (2025) Postman.com. Available at: https://www.postman.com/devrel/openai/documentation/k25n3c8/openai-api (Accessed: 17 April 2025).
Principles - OpenRouter's Core Values (2025) OpenRouter Documentation. OpenRouter | Documentation. Available at: https://openrouter.ai/docs/overview/principles (Accessed: 27 April 2025).
Rapid Application Development (RAD) (no date) Cost Efficient IT. Available at: https://www.agilelonestar.com/knowledge-base/rapid-application-development.
Resend (2025) Resend, Resend.com. Available at: https://resend.com/emails.
Run apps on the Android Emulator (no date) Android Developers. Available at: https://developer.android.com/studio/run/emulator.
Sassafras Patterdale (2025) How to Protect Your Cats (and Backyard Chickens) From Bird Flu, WIRED. Available at: https://www.wired.com/story/pets-and-backyard-flocks-are-at-risk-from-bird-flu-heres-how-to-protect-them/ (Accessed: 19 March 2025).
Senter, A. (2024) 'Toxic': Dog chemo sparks outrageous debate, news. news.com.au — Australia's leading news site. Available at: https://www.news.com.au/lifestyle/home/pets/owners-desperate-move-after-pet-dogs-heartbreaking-diagnosis/news-story/53e26bcb46f8d01fa981faa0967d35cb (Accessed: 19 March 2025).
Setting up Firebase / Google Analytics (2025) Shopgate.com. Available at: https://support.shopgate.com/en/migrated/knowledge/firebase-for-mobile-apps (Accessed: 17 April 2025).
Shei, A. (2023) Foundit | A Platform for Founders to Shares and Grow Ideas — UI/UX Case Study, Medium. Available at: https://medium.com/@aryn.shei/foundit-a-platform-for-founders-to-shares-and-grow-ideas-ui-ux-case-study-bfcc4a7bd5ac (Accessed: 6 September 2025).
Supabase (no date) Supabase Docs, supabase.com. Available at: https://supabase.com/docs.
TechNode Feed (2025) DeepSeek-V3 ends promotional pricing, updates API service rates, TechNode. Available at: https://technode.com/2025/02/10/deepseek-v3-ends-promotional-pricing-updates-api-service-rates/ (Accessed: 17 April 2025).
vetrec (2024) Vetrec.io. Available at: https://www.vetrec.io/post/the-evolution-of-veterinary-records-from-paper-to-ai-medical-records (Accessed: 19 March 2025).
Visual Studio Code (2023) Documentation for Visual Studio Code, code.visualstudio.com. Available at: https://code.visualstudio.com/docs.

What is Flutter? Guide for Flutter App Development | Relia Software (no date) reliasoftware.com. Available at: https://reliasoftware.com/blog/what-is-flutter.
Your First API Call | DeepSeek API Docs (2025) Deepseek.com. Available at: https://api-docs.deepseek.com/.

APPENDICES

Appendix A: Fact Findings Survey

AI Driven Pet Care App for Virtual Assistance and Symptom Diagnosis

Dear Respondents,

I am a final-year undergraduate student from the Bachelor of Science (Honours) Software Engineering program at Lee Kong Chian Faculty of Engineering & Science, Universiti Tunku Abdul Rahman (UTAR). As part of my Final Year Project, I am developing a mobile application titled "**AI-Driven Pet Care App for Virtual Assistance and Symptom Diagnosis".**

This study aims to explore how Artificial Intelligence (AI) can assist pet owners in managing their pets health more effectively. This app is designed to provide AI-powered symptom diagnosis, a virtual pet care assistant, and digital health record management to help pet owners track and maintain their pets well-being. Through this survey, I seek to understand the challenges pet owners face and gather insights to improve the app's features, usability, and effectiveness.

Your participation is voluntary, and all responses will be kept strictly confidential for academic research purposes only. I would greatly appreciate your time in completing this short questionnaire. Your feedback will play a key role in developing *this* application into a smart and accessible pet care solution.

For any inquiries, feel free to contact me at nivikaprasad@1utar.my.

Thank you for your time and support! 🐾

* Indicates required question

1. Email *

2. Do you own a pet? *

Mark only one oval.

- ◯ Yes
- ◯ No

Section 1: General Information

3. 1. What is your age group? *

Mark only one oval.

- ◯ Under 18
- ◯ 18 - 24
- ◯ 25 - 34
- ◯ 35 - 44
- ◯ 45 +

4. 2. What type of pet(s) do you own? *

(Select all that apply)

Check all that apply.

- ☐ Dog
- ☐ Cat
- ☐ Rabbit
- ☐ Hamster
- ☐ Bird
- ☐ Fish
- ☐ Guinea-pig
- ☐ Other: _____

5. 3. How many pets do you currently have? *

Mark only one oval.

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ More than 3

6. 4. How long have you been a pet owner? *

Mark only one oval.

- ◯ Less than a year
- ◯ 1 – 3 years
- ◯ 4 – 6 years
- ◯ More than 6 years

Section 2: Current Pet Care Practices

7. 5. How do you currently manage your pet's health records? *

Mark only one oval.

- ◯ Physical records (e.g., physical booklets)
- ◯ Digital notes (e.g., phone/laptop)
- ◯ Vet-provided documents
- ◯ I do not track my pet's health records
- ◯ Other: _____

8. 6. How often do you visit a veterinarian for checkups or health concerns? *

Mark only one oval.

- ◯ Monthly
- ◯ Every 3-6 months
- ◯ Once a year
- ◯ Only when my pet is sick
- ◯ Rarely/Never

9. 7. Have you ever struggled to identify if your pet was sick? *

Mark only one oval.

- ◯ Yes, many times
- ◯ Sometimes
- ◯ No, I usually know when they are sick

10. 8. How do you usually search for information when your pet shows unusual symptoms? *

(Select all that apply)

Check all that apply.

- ☐ Google/Online searches
- ☐ Pet-related forums or social media groups
- ☐ Consulting a veterinarian directly
- ☐ Asking other pet owners
- ☐ I don't search, I go straight to the vet

11. 9. What challenges do you face in managing your pet's health? *

(Select all that apply)

Check all that apply.

- ☐ Forgetting vaccination dates
- ☐ Keeping track of vet appointments
- ☐ Understanding pet symptoms and when to take action
- ☐ Finding reliable pet care advice
- ☐ Managing multiple pet's health records
- ☐ Other: _____

12. 10. Would you find a pet health tracking system useful? *

Mark only one oval.

- Yes, it would help me stay organized
- Maybe, if it's easy to use
- No, I don't need one

Section 3: AI Chatbot & App Features

13. 11. Do you currently use AI-based tools (e.g., AI chatbots or virtual assistants) to ask questions about pet care? *

Mark only one oval.

- Yes, frequently.
- Sometimes, but I also rely on other sources.
- Rarely, but I'm interested in exploring.
- No, and I don't think they'd be useful.

14. 12. Do you think an AI-powered symptom diagnosis tool could be helpful in identifying potential health issues before visiting a vet? *

Mark only one oval.

- Yes, very helpful
- Maybe, but I would still consult a vet
- No, I don't trust AI for health diagnosis

15. 13. Would you use an AI chatbot to answer general pet care questions? *

Mark only one oval.

- ◯ Yes, if the information is accurate
- ◯ Maybe, for minor issues
- ◯ No, I prefer speaking to a professional

16. 14. What type of assistance would you expect from an AI chatbot? *

(Select all that apply)

Check all that apply.

- ☐ Nutrition & feeding advice
- ☐ Training & behavioral guidance
- ☐ Common illness symptoms & basic first aid
- ☐ Information on vaccinations & pet health care
- ☐ Emergency care guidance
- ☐ Other (please specify)

17. 15. How likely are you to trust an AI tool to diagnose pet symptoms? *

Mark only one oval.

	1	2	3	4	5	
Unli	◯	◯	◯	◯	◯	Very Likely

18. 16. Would an AI symptom checker help you decide if your pet needs a vet visit? *

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Maybe

19. 17. How quickly do you expect responses from the AI chatbot? *

Mark only one oval.

- ◯ Instantly (within seconds)
- ◯ Within a minute
- ◯ Speed is not important, accuracy matters more

Section 4: Pet Health Record Management Preferences

20. 18. What features would you like in a pet health record system? *

(Select all that apply)

Check all that apply.

- ☐ Digital vaccination and medical history tracking
- ☐ Reminders for vaccinations and vet appointments
- ☐ Symptom logging for better tracking
- ☐ Weight and diet monitoring
- ☐ Secure cloud storage
- ☐ Other: _____

21. 19. How important is it to have a centralized digital system to track your pet's health? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Very Important

22. 20. Would you be interested in receiving pet care tips, alerts, and reminders through the app? *

Mark only one oval.

- ◯ Yes, I would love regular tips and updates
- ◯ Maybe, but only for important notifications
- ◯ No, I don't need reminders

Section 5: App Usability and Feature Preferences

23. 21. What factors are most important in a pet care app? *

(Select up to 3)

Check all that apply.

- ☐ Accuracy of AI diagnosis
- ☐ Easy-to-use interface
- ☐ Reliable pet care advice
- ☐ Health record storage and reminders
- ☐ Fast response times
- ☐ Data privacy and security

24. 22. Would you prefer the app to provide AI-powered symptom diagnosis for common pet illnesses?

Mark only one oval.

- ◯ Yes, it would help in assessing my pet's health
- ◯ Maybe, but I would still confirm with a vet
- ◯ No, I don't trust AI for symptom diagnosis

25. 23. What additional features do you think a pet care app should have? *

26. 24. Any final feedback or suggestions for improving the app?

Appendix B: SUS Survey

Participant 1:

Name: Aishwarya

Please indicate how strongly you agree or disagree with the following statements about your experience using the PawHub application. Select one option per statement. *

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I think that I would like to use this system frequently	○	○	○	◉	○
I found the system unnecessarily complex.	◉	○	○	○	○
I thought the system was easy to use.	○	○	○	○	◉
I think that I would need the support of a technical person to be able to use this system.	◉	○	○	○	○
I found the various functions in this system were well integrated.	○	○	○	◉	○
I thought there was too much inconsistency in this system.	◉	○	○	○	○
I would imagine that most people would learn to use this system very quickly.	○	○	○	○	◉
I found the system very awkward to use.	◉	○	○	○	○
I felt very confident using the system	○	○	○	○	◉
I needed to learn a lot of things before I could get going with this system.	○	◉	○	○	○

Participant 2:

Name: Avnaesh Sonia Singh

Please indicate how strongly you agree or disagree with the following statements about your experience using the PawHub application. Select one option per statement. *

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I think that I would like to use this system frequently	○	○	○	○	◉
I found the system unnecessarily complex.	◉	○	○	○	○
I thought the system was easy to use.	○	○	○	○	◉
I think that I would need the support of a technical person to be able to use this system.	◉	○	○	○	○
I found the various functions in this system were well integrated.	○	○	○	○	◉
I thought there was too much inconsistency in this system.	◉	○	○	○	○
I would imagine that most people would learn to use this system very quickly.	○	○	○	○	◉
I found the system very awkward to use.	◉	○	○	○	○
I felt very confident using the system	○	○	○	○	◉
I needed to learn a lot of things before I could get going with this system.	◉	○	○	○	○

Participant 3:

Name: Felicia Lau Yee Siew

Please indicate how strongly you agree or disagree with the following statements about your experience using the PawHub application. Select one option per statement. *

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I think that I would like to use this system frequently	○	○	○	◉	○
I found the system unnecessarily complex.	○	◉	○	○	○
I thought the system was easy to use.	○	○	○	◉	○
I think that I would need the support of a technical person to be able to use this system.	○	◉	○	○	○
I found the various functions in this system were well integrated.	○	○	○	◉	○
I thought there was too much inconsistency in this system.	○	◉	○	○	○
I would imagine that most people would learn to use this system very quickly.	○	○	○	◉	○
I found the system very awkward to use.	◉	○	○	○	○
I felt very confident using the system	○	○	◉	○	○
I needed to learn a lot of things before I could get going with this system.	○	○	◉	○	○

Participant 4:

Name: Estin Ling Wing Yen

Please indicate how strongly you agree or disagree with the following statements about your experience using the PawHub application. Select one option per statement. *

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I think that I would like to use this system frequently	○	○	○	◉	○
I found the system unnecessarily complex.	○	◉	○	○	○
I thought the system was easy to use.	○	○	○	○	◉
I think that I would need the support of a technical person to be able to use this system.	○	◉	○	○	○
I found the various functions in this system were well integrated.	○	○	○	○	◉
I thought there was too much inconsistency in this system.	○	◉	○	○	○
I would imagine that most people would learn to use this system very quickly.	○	○	○	○	◉
I found the system very awkward to use.	◉	○	○	○	○
I felt very confident using the system	○	○	○	○	◉
I needed to learn a lot of things before I could get going with this system.	◉	○	○	○	○

Participant 5:

Name: Woo Khai Ren

Please indicate how strongly you agree or disagree with the following statements about your experience using the PawHub application. Select one option per statement. *

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I think that I would like to use this system frequently	○	○	○	◉	○
I found the system unnecessarily complex.	○	○	◉	○	○
I thought the system was easy to use.	○	○	○	◉	○
I think that I would need the support of a technical person to be able to use this system.	○	◉	○	○	○
I found the various functions in this system were well integrated.	○	○	○	◉	○
I thought there was too much inconsistency in this system.	○	○	◉	○	○
I would imagine that most people would learn to use this system very quickly.	○	○	○	◉	○
I found the system very awkward to use.	○	◉	○	○	○
I felt very confident using the system	○	○	○	◉	○
I needed to learn a lot of things before I could get going with this system.	○	◉	○	○	○

Participant 6:

Name: Babita

Please indicate how strongly you agree or disagree with the following statements about your experience using the PawHub application. Select one option per statement. *

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I think that I would like to use this system frequently	○	○	○	○	◉
I found the system unnecessarily complex.	◉	○	○	○	○
I thought the system was easy to use.	○	○	○	○	◉
I think that I would need the support of a technical person to be able to use this system.	◉	○	○	○	○
I found the various functions in this system were well integrated.	○	○	○	○	◉
I thought there was too much inconsistency in this system.	◉	○	○	○	○
I would imagine that most people would learn to use this system very quickly.	○	○	○	◉	○
I found the system very awkward to use.	◉	○	○	○	○
I felt very confident using the system	○	○	○	◉	○
I needed to learn a lot of things before I could get going with this system.	◉	○	○	○	○

Participant 7:

Name: Yap Ming Jun

Please indicate how strongly you agree or disagree with the following statements about your experience using the PawHub application. Select one option per statement. *

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I think that I would like to use this system frequently	○	○	○	◉	○
I found the system unnecessarily complex.	○	○	◉	○	○
I thought the system was easy to use.	○	○	○	◉	○
I think that I would need the support of a technical person to be able to use this system.	○	◉	○	○	○
I found the various functions in this system were well integrated.	○	○	○	◉	○
I thought there was too much inconsistency in this system.	○	◉	○	○	○
I would imagine that most people would learn to use this system very quickly.	○	○	○	◉	○
I found the system very awkward to use.	○	◉	○	○	○
I felt very confident using the system	○	○	○	◉	○
I needed to learn a lot of things before I could get going with this system.	○	◉	○	○	○

Participant 8:

Name: Yeap Huai Zhou

Please indicate how strongly you agree or disagree with the following statements about your experience using the PawHub application. Select one option per statement. *

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I think that I would like to use this system frequently	○	○	○	◉	○
I found the system unnecessarily complex.	○	◉	○	○	○
I thought the system was easy to use.	○	○	○	○	◉
I think that I would need the support of a technical person to be able to use this system.	◉	○	○	○	○
I found the various functions in this system were well integrated.	○	○	○	◉	○
I thought there was too much inconsistency in this system.	◉	○	○	○	○
I would imagine that most people would learn to use this system very quickly.	○	○	○	◉	○
I found the system very awkward to use.	○	◉	○	○	○
I felt very confident using the system	○	○	○	○	◉
I needed to learn a lot of things before I could get going with this system.	◉	○	○	○	○

Participant 9:

Name: Nikita Prasad

Please indicate how strongly you agree or disagree with the following statements about your experience using the PawHub application. Select one option per statement. *

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I think that I would like to use this system frequently	○	○	○	○	◉
I found the system unnecessarily complex.	○	◉	○	○	○
I thought the system was easy to use.	○	○	○	○	◉
I think that I would need the support of a technical person to be able to use this system.	◉	○	○	○	○
I found the various functions in this system were well integrated.	○	○	○	◉	○
I thought there was too much inconsistency in this system.	◉	○	○	○	○
I would imagine that most people would learn to use this system very quickly.	○	○	○	◉	○
I found the system very awkward to use.	◉	○	○	○	○
I felt very confident using the system	○	○	○	◉	○
I needed to learn a lot of things before I could get going with this system.	◉	○	○	○	○

Participant 10:

Name: Diksha Suri

Please indicate how strongly you agree or disagree with the following statements about your experience using the PawHub application. Select one option per statement. *

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I think that I would like to use this system frequently	○	○	○	◉	○
I found the system unnecessarily complex.	○	◉	○	○	○
I thought the system was easy to use.	○	○	◉	○	○
I think that I would need the support of a technical person to be able to use this system.	○	◉	○	○	○
I found the various functions in this system were well integrated.	○	○	○	◉	○
I thought there was too much inconsistency in this system.	○	◉	○	○	○
I would imagine that most people would learn to use this system very quickly.	○	○	◉	○	○
I found the system very awkward to use.	○	◉	○	○	○
I felt very confident using the system	○	○	◉	○	○
I needed to learn a lot of things before I could get going with this system.	○	◉	○	○	○

Appendix C: UAT Results

Participant 1:

Name: Charles Lee Ung Kiet

Test Starting Time: 9:45am

Test Ending Time: 10:15am

PawHub Application				
Test Module	**Test Case ID**	**Test Scenario**	**Status**	**Comment**
Register	UAT-001	Able to register a new account with valid username, email, phone, and password.	Pass	-
Login	UAT-002	Able to log in with valid credentials.	Pass	-
Forgot Password	UAT-003	Able to reset password using registered email.	Pass	-
Pet Management	UAT-004	Able to add a new pet with photo, name, breed, birthday, weight, and height.	Pass	Took me a second to find the add button
	UAT-005	Able to edit existing pet profile.	Pass	-
	UAT-006	Able to delete a pet and confirm associated records are removed.	Pass	-
Health Records	UAT-007	Able to add a health record with future date and enable email reminder.	Pass	-
	UAT-008	Able to receive email reminder 24 hours before scheduled event.	Pass	-
	UAT-009	Able to edit existing health records.	Pass	-

	UAT-010	Able to delete a health record and confirm associated email reminders are removed.	Pass	-
AI Chatbot	UAT-011	Able to send message to AI chatbot and receive relevant response.	Pass	-
	UAT-012	Able to switch between AI models.	Pass	-
	UAT-013	Able to switch between Pets.	Pass	-
	UAT-014	Able to rate AI message with 1–5 stars.	Pass	-
Symptom Diagnosis	UAT-015	Able to submit symptoms and receive AI-generated diagnosis with severity level.	Pass	-
	UAT-016	Able to view and delete past symptom assessments.	Pass	-
Education	UAT-017	Able to search articles and filter by category.	Pass	-
	UAT-018	Able to tap article to open original article webpage.	Pass	opened in browser. No crashes.
Feedback	UAT-019	Able to submit feedback with rating and comment.	Pass	-
	UAT-020	Able to view past feedback submissions.	Pass	-
Profile Management	UAT-021	Able to update username, phone, and password.	Pass	-
	UAT-022	Able to upload profile picture from gallery or camera.	Pass	-
	UAT-023	Able to select default avatar.	Pass	-

	UAT-024	Able to log out.	Pass	-
	UAT-025	Able to delete account.	Pass	-

Participant 2:

Name: Tee Junn Jeh

Test Starting Time: 1:05pm

Test Ending Time: 1:35pm

PawHub Application				
Test Module	**Test Case ID**	**Test Scenario**	**Status**	**Comment**
Register	UAT-001	Able to register a new account with valid username, email, phone, and password.	Pass	-
Login	UAT-002	Able to log in with valid credentials.	Pass	-
Forgot Password	UAT-003	Able to reset password using registered email.	Pass	-
Pet Management	UAT-004	Able to add a new pet with photo, name, breed, birthday, weight, and height.	Pass	-
	UAT-005	Able to edit existing pet profile.	Pass	-
	UAT-006	Able to delete a pet and confirm associated records are removed.	Pass	-
Health Records	UAT-007	Able to add a health record with future date and enable email reminder.	Pass	-
	UAT-008	Able to receive email	Pass	-

		reminder 24 hours before scheduled event.		
	UAT-009	Able to edit existing health records.	Pass	-
	UAT-010	Able to delete a health record and confirm associated email reminders are removed.	Pass	-
AI Chatbot	UAT-011	Able to send message to AI chatbot and receive relevant response.	Pass	-
	UAT-012	Able to switch between AI models.	Pass	-
	UAT-013	Able to switch between Pets.	Pass	-
	UAT-014	Able to rate AI message with 1–5 stars.	Pass	-
Symptom Diagnosis	UAT-015	Able to submit symptoms and receive AI-generated diagnosis with severity level.	Pass	My wifi disconnected, the results didn't show in 1st try.
	UAT-016	Able to view and delete past symptom assessments.	Pass	-
Education	UAT-017	Able to search articles and filter by category.	Pass	-
	UAT-018	Able to tap article to open original article webpage.	Pass	-
Feedback	UAT-019	Able to submit feedback with rating and comment.	Pass	-
	UAT-020	Able to view past feedback submissions.	Pass	-

Profile Management	UAT-021	Able to update username, phone, and password.	Pass	-
	UAT-022	Able to upload profile picture from gallery or camera.	Pass	-
	UAT-023	Able to select default avatar.	Pass	-
	UAT-024	Able to log out.	Pass	-
	UAT-025	Able to delete account.	Pass	-

Participant 3:

Name: Vanex

Test Starting Time: 11:00am

Test Ending Time: 11:30am

PawHub Application				
Test Module	**Test Case ID**	**Test Scenario**	**Status**	**Comment**
Register	UAT-001	Able to register a new account with valid username, email, phone, and password.	Pass	-
Login	UAT-002	Able to log in with valid credentials.	Pass	-
Forgot Password	UAT-003	Able to reset password using registered email.	Pass	-
Pet Management	UAT-004	Able to add a new pet with photo, name, breed, birthday, weight, and height.	Pass	-
	UAT-005	Able to edit existing pet profile.	Pass	-
	UAT-006	Able to delete a pet and confirm associated records	Pass	-

		are removed.		
Health Records	UAT-007	Able to add a health record with future date and enable email reminder.	Pass	-
	UAT-008	Able to receive email reminder 24 hours before scheduled event.	Pass	-
	UAT-009	Able to edit existing health records.	Pass	-
	UAT-010	Able to delete a health record and confirm associated email reminders are removed.	Pass	-
AI Chatbot	UAT-011	Able to send message to AI chatbot and receive relevant response.	Pass	-
	UAT-012	Able to switch between AI models.	Pass	-
	UAT-013	Able to switch between Pets.	Pass	-
	UAT-014	Able to rate AI message with 1–5 stars.	Pass	-
Symptom Diagnosis	UAT-015	Able to submit symptoms and receive AI-generated diagnosis with severity level.	Pass	-
	UAT-016	Able to view and delete past symptom assessments.	Pass	-
Education	UAT-017	Able to search articles and filter by category.	Pass	-
	UAT-018	Able to tap article to open original article webpage.	Pass	-
Feedback	UAT-019	Able to submit feedback with rating and comment.	Pass	-
	UAT-020	Able to view past feedback	Pass	-

		submissions.		
Profile Management	UAT-021	Able to update username, phone, and password.	Pass	-
	UAT-022	Able to upload profile picture from gallery or camera.	Pass	-
	UAT-023	Able to select default avatar.	Pass	-
	UAT-024	Able to log out.	Pass	-
	UAT-025	Able to delete account.	Pass	-

Participant 4:

Name: Alex Ting

Test Starting Time: 2:36pm

Test Ending Time: 3:06pm

PawHub Application				
Test Module	**Test Case ID**	**Test Scenario**	**Status**	**Comment**
Register	UAT-001	Able to register a new account with valid username, email, phone, and password.	Pass	-
Login	UAT-002	Able to log in with valid credentials.	Pass	-
Forgot Password	UAT-003	Able to reset password using registered email.	Pass	-
Pet Management	UAT-004	Able to add a new pet with photo, name, breed, birthday, weight, and height.	Pass	-
	UAT-005	Able to edit existing pet profile.	Pass	-
	UAT-006	Able to delete a pet and confirm associated records	Pass	-

		are removed.		
Health Records	UAT-007	Able to add a health record with future date and enable email reminder.	Pass	-
	UAT-008	Able to receive email reminder 24 hours before scheduled event.	Pass	Good, received the email.
	UAT-009	Able to edit existing health records.	Pass	-
	UAT-010	Able to delete a health record and confirm associated email reminders are removed.	Pass	-
AI Chatbot	UAT-011	Able to send message to AI chatbot and receive relevant response.	Pass	-
	UAT-012	Able to switch between AI models.	Pass	-
	UAT-013	Able to switch between Pets.	Pass	-
	UAT-014	Able to rate AI message with 1–5 stars.	Pass	-
Symptom Diagnosis	UAT-015	Able to submit symptoms and receive AI-generated diagnosis with severity level.	Pass	-
	UAT-016	Able to view and delete past symptom assessments.	Pass	-
Education	UAT-017	Able to search articles and filter by category.	Pass	-
	UAT-018	Able to tap article to open original article webpage.	Pass	-
Feedback	UAT-019	Able to submit feedback with rating and comment.	Pass	-
	UAT-020	Able to view past feedback	Pass	-

		submissions.		
Profile Management	UAT-021	Able to update username, phone, and password.	Pass	Updated username Form validated instantly.
	UAT-022	Able to upload profile picture from gallery or camera.	Pass	-
	UAT-023	Able to select default avatar.	Pass	-
	UAT-024	Able to log out.	Pass	-
	UAT-025	Able to delete account.	Pass	-

Participant 5:

Name: Benjamin

Test Starting Time: 3:10pm

Test Ending Time: 3:40pm

PawHub Application				
Test Module	**Test Case ID**	**Test Scenario**	**Status**	**Comment**
Register	UAT-001	Able to register a new account with valid username, email, phone, and password.	Pass	-
Login	UAT-002	Able to log in with valid credentials.	Pass	-
Forgot Password	UAT-003	Able to reset password using registered email.	Pass	-
Pet Management	UAT-004	Able to add a new pet with photo, name, breed, birthday, weight, and height.	Pass	-

	UAT-005	Able to edit existing pet profile.	Pass	-
	UAT-006	Able to delete a pet and confirm associated records are removed.	Pass	-
Health Records	UAT-007	Able to add a health record with future date and enable email reminder.	Pass	-
	UAT-008	Able to receive email reminder 24 hours before scheduled event.	Pass	-
	UAT-009	Able to edit existing health records.	Pass	-
	UAT-010	Able to delete a health record and confirm associated email reminders are removed.	Pass	-
AI Chatbot	UAT-011	Able to send message to AI chatbot and receive relevant response.	Pass	Impressive, the AI was able to answer based on my pet information.
	UAT-012	Able to switch between AI models.	Pass	-
	UAT-013	Able to switch between Pets.	Pass	-
	UAT-014	Able to rate AI message with 1–5 stars.	Pass	-
Symptom Diagnosis	UAT-015	Able to submit symptoms and receive AI-generated diagnosis with severity	Pass	-

		level.		
	UAT-016	Able to view and delete past symptom assessments.	Pass	-
Education	UAT-017	Able to search articles and filter by category.	Pass	-
	UAT-018	Able to tap article to open original article webpage.	Pass	-
Feedback	UAT-019	Able to submit feedback with rating and comment.	Pass	-
	UAT-020	Able to view past feedback submissions.	Pass	-
Profile Management	UAT-021	Able to update username, phone, and password.	Pass	-
	UAT-022	Able to upload profile picture from gallery or camera.	Pass	-
	UAT-023	Able to select default avatar.	Pass	-
	UAT-024	Able to log out.	Pass	-
	UAT-025	Able to delete account.	Pass	-