Hamdard University
Department of Computing
Final Year Project

Pet health care industry

Software Requirements Specifications

Submitted by
SUBHAN (571-2019)
HARIS BIN MOHSIN (1516-2020)
M.ASAD IBRAHIM(2602-2021)

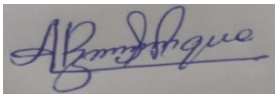
Supervisor(s)
SIR ABDUL RAZZAQUE

Spring 2020

Document Sign off Sheet

Document Information

Project Title	**Pet health Care industry**
Project Code	
Document Name	Software Requirements Specifications
Document Version	1.3
Document Identifier	FYP-041 FL-24 SRS
Document Status	Draft
Author(s)	SUBHAN (571-2019)
Approver(s)	SIR ABDUL RAZZAQUE
Issue Date	17-1-2025

Name	Role	Signature	Date
SUBHAN (571-2019)	Team Lead		
HARIS BIN MOHSIN (1516-2020)	Team Member 2		
M. ASAD IBRAHIM (2602 2021)	Team Member 3		
SIR ABDUL RAZZAQUE	Supervisor		

Revision History

Date	Version	Description	Author
15/Jan/2025	1.0	First Version	Subhan,haris , asaad

Definition of Terms, Acronyms, and Abbreviations

[This section should provide the definitions of all terms, acronyms, and abbreviations required to interpret the terms used in the document properly.]

Here is a more concise list of essential technical terms, acronyms, and abbreviations relevant to your app:

Definition of Terms, Acronyms, and Abbreviations

Term	Description
AI (Artificial Intelligence)	Technology that simulates human intelligence, used in the app for disease detection through image analysis.
API (Application Programming Interface)	A set of protocols that allow different software components to communicate, such as between the app and backend.
Kotlin	A programming language used to develop the Android app's frontend.
Node.js	A JavaScript runtime used for building the app's backend APIs.
Express	A web framework for Node.js that simplifies API development.
MySQL	A relational database system for storing app data like user profiles and pet information.
Firebase	A platform by Google providing services like authentication, real-time database, and push notifications.
Flask	A lightweight Python web framework used to build the AI model's backend service.
Push Notifications	Messages sent to users to alert them of important events, like meal reminders or new blog posts.
REST API (Representational State Transfer API)	A set of web services using HTTP for communication between the app and backend.
OAuth2	An authorization protocol that allows users to securely log in and share data with third-party apps.
2FA (Two-Factor Authentication)	A security measure requiring two forms of identification for logging into the app.
CRUD (Create, Read, Update, Delete)	The basic operations for managing data, such as adding, viewing, or modifying user and pet profiles.
JSON (JavaScript Object Notation)	A lightweight data format used for exchanging

Table of Contents

1. Introduction

1.1 Purpose of Document

The purpose of this document is to outline the requirements, design, and specifications for the Smart Paw app, a mobile application aimed at enhancing pet care through user-friendly features, including AI disease detection, meal reminders, and veterinary consultations.

1.2 Intended Audience

This document is intended for project stakeholders, including developers, designers, project managers, and potential investors, as well as end-users who will benefit from the Smart Paw app.

Problem Statement

The Smart Paw app addresses the challenge of managing pet health by providing a comprehensive platform for disease detection, meal scheduling, and access to veterinary services, ensuring pet owners have the necessary tools to care for their pets effectively.

1. Overall System Description

. Overall System Description

1.1 Project Background

The Smart Paw app was conceived to address the growing need for effective pet management solutions in an increasingly digital world. With pet ownership on the rise, there is a need for tools that help pet owners monitor their pets' health, manage feeding schedules, and access veterinary care efficiently.

1.2 Problem Statement

Many pet owners struggle to manage their pets' health and well-being due to a lack of accessible resources. The Smart Paw app aims to simplify this process by providing features for disease detection, meal reminders, and easy access to veterinary consultations.

1.3 Project Scope

The project scope includes the development of a mobile application that provides:

- User and pet profiles
- AI-driven disease detection through image uploads
- Meal scheduling and notification features
- Access to a directory of veterinary doctors
- A community section for health tips and blog sharing

1.4 Not In Scope

- The app will not include in-person veterinary consultations or emergency services.
- The development of a web-based version of the app is not included in this phase.
- Features unrelated to pet health management, such as social networking beyond the community blog, are excluded.

1.5 Project Objectives

- To develop a user-friendly mobile application that enhances pet health management.
- To implement an AI model for accurate disease detection from images.
- To ensure timely notifications for pet feeding schedules.
- To provide a comprehensive list of veterinary services accessible from within the app.
- To foster a community of pet owners through shared knowledge and experiences.

1.6 Stakeholders & Affected Groups

- **Primary Stakeholders**: Team
- **Secondary Stakeholders**: Developers, project managers, and investors.
- **Affected Groups**: Pets, as the primary beneficiaries of improved health management, and families of pet owners who benefit from enhanced care options.

1.7 Operating Environment

The Smart Paw app will operate on Android, requiring internet access for most functionalities. The backend services will be hosted on server(Firebase MySQL), ensuring real-time data synchronization and cloud storage.

1.8 System Constraints

- Limited by the capabilities of the AI model, which may not cover all possible pet diseases.
- Dependent on internet connectivity for key functionalities like image uploads and veterinary consultations.
- Performance may vary based on the device used for app installation and use.

1.9 Assumptions & Dependencies

- It is assumed that users will have access to smartphones with internet connectivity.
- The project depends on the successful training of the AI model with a sufficient dataset of pet diseases.
- Ongoing support from veterinary professionals is assumed for providing accurate information within the app.

2. External Interface Requirements

2.1 Hardware Interfaces

- **Mobile Devices**:
  - The Smart Paw app interfaces with Android smartphones and tablets.
  - **Logical Structure**: The app operates within the device's application layer, utilizing the operating system's APIs for interaction with device hardware (e.g., camera, notifications).
  - **Physical Addresses**: The app will access the device's camera and storage for image uploads and local data storage.
  - **Expected Behavior**: The app should seamlessly access the camera to capture images for disease detection, receive user input via touch, and display notifications for feeding schedules.

2.2 Software Interfaces

- **Firebase**:
  - **External Owner**: Google
  - **Interface Details**:
    - The app interfaces with backend via Rest API for user authentication, real-time database access, and cloud storage.
    - Utilizes backend via Rest API Authentication for user login and registration processes.

  TensorFlow/PyTorch:

  - **External Owner**: Open-source community
  - **Interface Details**:
    - The app interfaces with the chosen AI framework (TensorFlow or PyTorch) for processing images uploaded for disease detection.
    - Expected to leverage pre-trained models and provide APIs for inference requests.
- **Retrofit**:
  - **External Owner**: Square, Inc.
  - **Interface Details**:
    - Retrofit is used for making network calls to APIs, including any third-party services for additional functionalities or data retrieval.

2.3 Communications Interfaces

- **Local Area Network (LAN)**:
  - The app will communicate over Wi-Fi or mobile data networks to connect with Firebase and other online services.

    - Supports standard communication protocols (HTTP/HTTPS) for secure data transmission.
- **Push Notifications**:
    - The app will utilize Firebase Cloud Messaging (FCM) to send push notifications for meal reminders and updates.
- **Image Upload**:
    - The app will allow users to upload images directly from their devices to the Firebase storage using HTTP POST requests.

3. System Functions / Functional Requirements

3.1 System Functions

Ref #	Functions	Category	Attribute	Details & Boundary Constraints
R1.1	Allow users to register and log in	Evident	System Response Time	Login and registration should occur within 3 seconds.
R1.2	Enable users to create and manage pet profiles	Evident	Data Integrity	All pet data must be saved accurately with no loss.
R1.3	Provide AI disease detection by uploading an image	Evident	AI Model Response Time	Detection results should be returned within 5 seconds.
R1.4	Send meal reminders via push notifications	Evident	Notification Delivery Time	Notifications must be delivered within 1 minute of the scheduled time.
R1.5	Display a list of available veterinary contacts	Evident	System Availability	Veterinary information should be updated in real-time.
R1.6	Allow users to schedule feeding times for their pets	Evident	User Customization	Users can set multiple feeding times per pet.
R1.7	Provide access to community blog posts	Evident	Content Loading Time	Blog posts should load within 2 seconds.
R1.8	Enable users to contact veterinary professionals	Evident	System Responsiveness	Contact features should initiate within 3 seconds.
R1.9	Store user data securely in Database	Hidden	Data Security	All user data must be encrypted during storage and transmission.
R1.10	Perform background data syncing with Firebase	Hidden	System Performance	Background syncing should not interfere with app performance.
R1.12	Allow users to upload blog posts	Frill	Content Moderation	User-generated content must be reviewed for appropriateness before publishing.

System Attributes/Nonfunctional Requirements

Attribute	Details and Boundary Constraints	Category
Response Time	(Boundary constraint) All user actions, including registration and login, must complete within 5 seconds.	Mandatory
Concurrent User Load	A minimum of 50 users should be able to connect simultaneously without performance degradation.	Mandatory
Usability	The app should have an intuitive user interface, enabling new users to navigate basic features without external help.	Mandatory
Security	User data must be encrypted by end points. Must be using Https.	Mandatory
Compatibility	The app must support the latest two major versions of Android .	Mandatory
Performance	The app should maintain smooth operation without lag, even under maximum load conditions.	Mandatory
Battery Efficiency	The app should minimize battery consumption, aiming for a battery drain of less than 15% per hour of active use.	Optional
Aesthetics	The interface should use a modern design with appealing visuals and color schemes to enhance user engagement.	Optional

3.2.1 List of Actors

- **Pet Owner**: This user manages their profile, pets, and health-related activities.
- **Veterinary Professional**: This actor provides consultation and manages veterinary information.
- **AI Model**: The system that processes images to identify pet diseases.
- **Administrator**: Manages app content, user data, and oversees the overall system functionality.

3.2.2 List of Use Cases

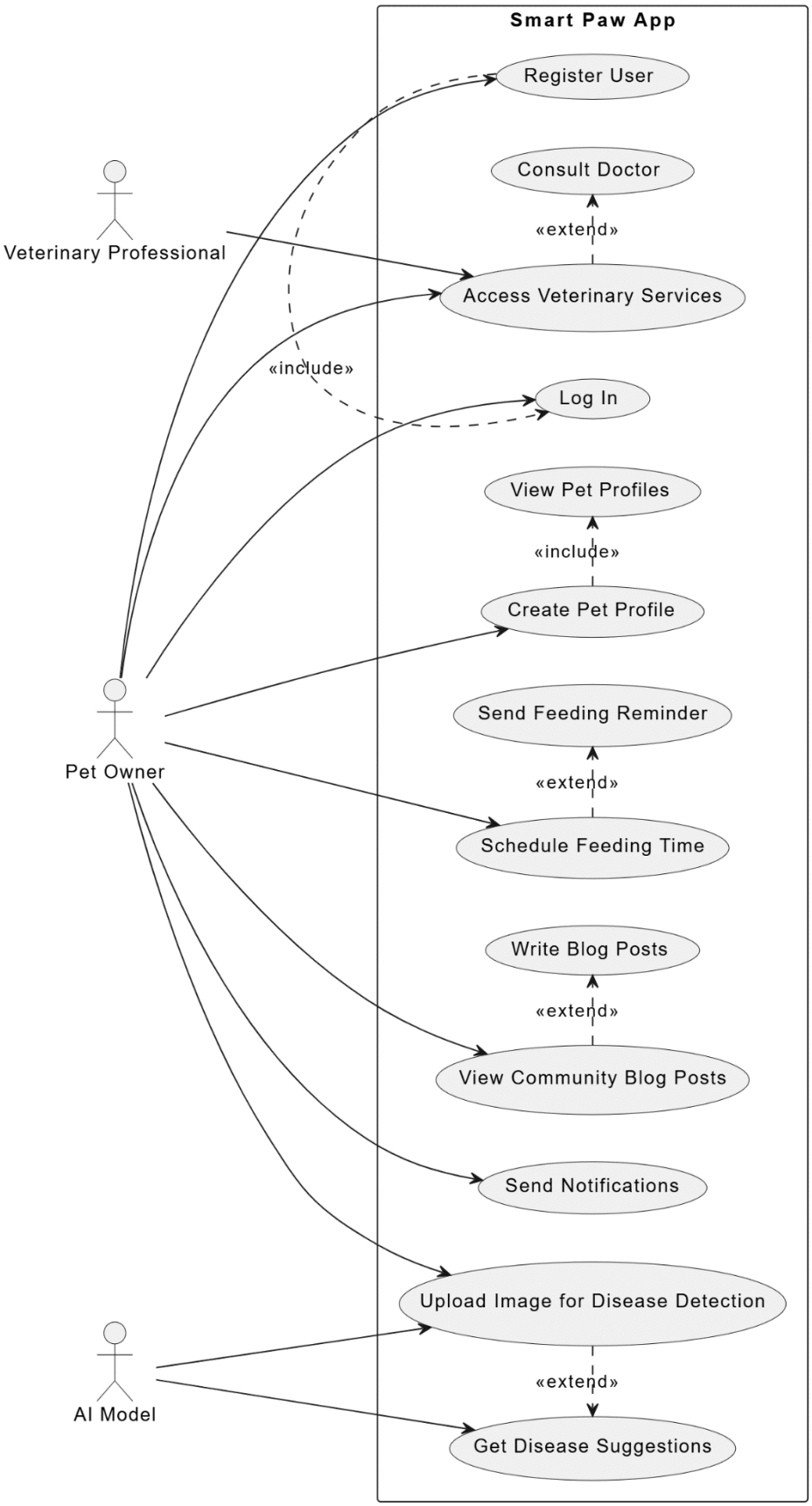
Use Case ID	Use Case Name	Description
UC1	Register User	Allows a new user to create an account with personal and pet information.
UC2	Log In	Authenticates existing users to access their profiles and app features.
UC3	Create Pet Profile	Enables users to add and manage profiles for up to three pets.

UC4	Upload Image for Disease Detection	Allows users to upload images of symptoms for AI analysis to identify potential diseases.
UC5	Schedule Feeding Time	Lets users set and modify feeding schedules for their pets, receiving reminders when due.
UC6	Access Veterinary Services	Provides a list of veterinary professionals available for consultation and contact.
UC7	View Community Blog Posts	Allows users to read and contribute to blog posts related to pet health and care.
UC8	Send Notifications	Sends reminders and alerts to users about feeding times and health tips.

3.2.3 Use Case Diagram

A use case diagram visually represents the interactions between actors and use cases.

1. **Actors**:
   - Place the **Pet Owner**, **Veterinary Professional**, **AI Model**, as stick figures or labeled icons on the left side of the diagram.
2. **Use Cases**:
   - Represent each use case as an oval shape, labeled with the use case name.
   - Arrange the use cases in the center of the diagram.
3. **Relationships**:
   - Draw lines connecting actors to their relevant use cases. For example, connect the **Pet Owner** to **Register User**, **Log In**, **Create Pet Profile**, etc.
   - Connect the **Veterinary Professional** to **Access Veterinary Services**.
   - Connect the **AI Model** to **Upload Image for Disease Detection**.
   - Connect the **User** to **View Community Blog Posts** and other relevant use cases.


Smart Paw App
Register User
Consult Doctor
«extend»
Veterinary Professional
Access Veterinary Services
«include»
Log In
View Pet Profiles
«include»
Create Pet Profile
Pet Owner
Send Feeding Reminder
«extend»
Schedule Feeding Time
Write Blog Posts
«extend»
View Community Blog Posts
Send Notifications
Upload Image for Disease Detection
AI Model
«extend»
Get Disease Suggestions

Description of Use Cases

Section : Register User

Actors: Pet Owner
Purpose: Allows a new user to create an account with personal and pet information.
Description: A user (Pet Owner) registers by providing personal information and, optionally, details about their pets to create a profile. This enables them to access and use all features of the app, including pet management, health tracking, and disease detection.
Cross References: Functions: R1.1
Use Cases: The user must complete the registration before they can log in and manage pets.

Pre-Conditions

- The user has not registered previously.
- The registration screen is accessible.

Successful Post-Conditions

- A new account is created, and the user is prompted to log in.
- The user can access features related to pet management, disease detection, and consultations.

Failure Post-Conditions

- No account is created, and the user remains unregistered.

Typical Course of Events

Actor Action	System Response
1. The Pet Owner navigates to the registration screen.	2. The system displays the registration form.
3. The Pet Owner enters required personal information (name, email, etc.).	4. The system validates the entered data.
5. The Pet Owner optionally adds pet information (name, breed, age, etc.).	6. The system validates the pet data if provided.
7. The Pet Owner submits the registration form.	8. The system creates the account and stores data.
9. The system prompts the user to log in.	10. The user proceeds to the login screen.

Alternative Course

- **Step 4:** Invalid data is entered (e.g., missing required fields).
  - **System Response:** The system highlights errors and prompts the user to correct them.

Section: Log In

Actors: Pet Owner
Purpose: Authenticates existing users to access their profiles and app features.
Description: A registered user logs into their account by entering their credentials (email and password) to access pet profiles, health tracking, and other app functionalities.
Cross References: Functions: R1.2
Use Cases: The user must complete the registration first before logging in.

Pre-Conditions

- The user has registered an account.
- The login screen is accessible.

Successful Post-Conditions

- The user is authenticated, and they are directed to the main dashboard.

Failure Post-Conditions

- The user remains unlogged in and cannot access the system features.

Typical Course of Events

Actor Action	System Response
1. The Pet Owner navigates to the login screen.	2. The system prompts the user to enter their credentials.
3. The Pet Owner enters their email and password.	4. The system validates the entered credentials.
5. The Pet Owner submits the login form.	6. The system logs the user in and redirects to the main dashboard.

Alternative Course

- **Step 4:** Invalid credentials (wrong email or password).

- **System Response:** The system displays an error message and prompts the user to re-enter their credentials.

Section : Create Pet Profile

Actors: Pet Owner
Purpose: Allows users to create and manage profiles for their pets.
Description: A user can add up to three pets, specifying details such as name, breed, age, medical history, etc. This helps track each pet's health and connect with relevant services (e.g., disease detection and veterinary consultation).
Cross References: Functions: R2.1
Use Cases: The Pet Owner must be logged in to create a pet profile.

Pre-Conditions

- The Pet Owner is logged into the system.
- The system allows the creation of a pet profile.

Successful Post-Conditions

- A pet profile is created, and the system stores the pet's data.

Failure Post-Conditions

- The pet profile is not created, and the system prompts the user to try again.

Typical Course of Events

Actor Action	System Response
1. The Pet Owner logs in and navigates to the pet profile page.	2. The system displays an option to create a new pet profile.
3. The Pet Owner enters pet details (name, breed, age, medical history).	4. The system validates the entered data.
5. The Pet Owner submits the pet information.	6. The system saves the pet profile and displays a confirmation message.
7. The Pet Owner can now manage or view the pet profile.	8. The system redirects the user to the pet profile page.

Alternative Course

- **Step 4:** Invalid data (e.g., incomplete or incorrect details) entered.

- **System Response:** The system prompts the user to correct the errors.

Section: Upload Image for Disease Detection

Actors: Pet Owner, AI Model
Purpose: Allows users to upload images of symptoms for AI analysis to identify potential diseases.
Description: Users can upload a photo of their pet's symptoms, such as skin conditions or other visible issues. The AI model processes the image to detect diseases and provides results.
Cross References: Functions: R3.1
Use Cases: The Pet Owner must be logged in and have a pet profile created.

Pre-Conditions

- The Pet Owner is logged into the system and has a valid pet profile.
- The system is ready to receive image uploads.

Successful Post-Conditions

- The AI model processes the image and returns the disease detection results.

Failure Post-Conditions

- The image upload fails, and the system does not return results.

Typical Course of Events

Actor Action	System Response
1. The Pet Owner navigates to the "AI Detection" page.	2. The system prompts the user to upload an image of the symptoms.
3. The Pet Owner uploads an image of their pet's symptoms.	4. The system validates the image format and quality.
5. The AI model processes the image and analyzes it.	6. The system returns disease detection results.
7. The Pet Owner views the detection results.	8. The system provides suggestions or advice for next steps.

Alternative Course

- **Step 4:** Invalid image format or poor quality.

- **System Response:** The system asks the Pet Owner to upload a valid image.

Section: Schedule Feeding Time

Actors: Pet Owner
Purpose: Lets users set and modify feeding schedules for their pets, receiving reminders when due.
Description: The Pet Owner sets specific times to feed their pets and the app sends notifications to remind them when it's time to feed their pets.
Cross References: Functions: R4.1
Use Cases: The Pet Owner must be logged in to set feeding times.

Pre-Conditions

- The Pet Owner is logged into the system.
- The system allows the user to set feeding times.

Successful Post-Conditions

- The feeding schedule is created, and notifications are set.

Failure Post-Conditions

- No feeding schedule is created, and the system prompts the user to try again.

Typical Course of Events

Actor Action	System Response
1. The Pet Owner navigates to the "Feeding Time" section.	2. The system prompts the user to input feeding time(s).
3. The Pet Owner sets the feeding time for their pet(s).	4. The system validates the input time(s) and stores the data.
5. The system confirms the feeding time is set.	6. The system sends notifications at the set times.

Alternative Course

- **Step 4:** Invalid feeding time entered (e.g., in the past).
  - **System Response:** The system prompts the user to enter a valid time.

Section : View Community Blog Posts

Actors: Pet Owner
Purpose: Allows users to read and contribute to blog posts related to pet health and care.
Description: Users can read and write community blog posts to share tips, experiences, and knowledge about pet care.
Cross References: Functions: R5.1
Use Cases: The Pet Owner must be logged in to access and write blog posts.

Pre-Conditions

- The Pet Owner is logged in.
- The system allows access to community blog posts.

Successful Post-Conditions

- The user can read blogs and contribute to them.

Failure Post-Conditions

- The user cannot access the blog posts, or no new posts are created.

Typical Course of Events

Actor Action	System Response
1. The Pet Owner navigates to the community blog section.	2. The system displays a list of available blogs.
3. The Pet Owner selects a blog post to read.	4. The system loads and displays the blog content.
5. The Pet Owner reads the blog and may comment or write a new post.	6. The system updates the blog content and comments.

Alternative Course

- **Step 3:** The blog post is unavailable or has been removed.
    - **System Response:** The system prompts the user with an error or redirects to a list of available blogs.

4. Non - Functional Requirements

To outline the Non-Functional Requirements for the "Smart Paw" app, we need to focus on aspects like performance, safety, security, reliability, usability, supportability, and user documentation. Below are the detailed descriptions for each category:

4. Non-Functional Requirements

4.1 Performance Requirements

- **App Launch Time:** The app should load and display the splash screen within 2 seconds, and the main dashboard should appear within 5 seconds after login.
- **AI Disease Detection:** The AI model should process and return results within 5-10 seconds after the user uploads an image of the pet's symptoms.
- **Navigation Speed:** All in-app navigation transitions (e.g., from home to profile, or to AI detection) should occur within 1 second to ensure smooth user experience.
- **Notifications:** Push notifications should be delivered within 5 seconds of a relevant event, such as meal time reminders or new blog posts.

4.2 Safety Requirements

- **Data Privacy:** All personal data, including user and pet information, should be stored securely, in compliance with GDPR and other relevant data protection regulations.
- **Image Upload Protection:** Uploaded pet images should undergo checks for malware or other harmful content.
- **Authentication:** The app should include two-factor authentication (2FA) for securing the user login and registration processes.
- **User Consent:** Explicit consent from users should be obtained for data collection, especially for sensitive information related to pets' health and diseases.

4.3 Security Requirements

- **Data Encryption:** Sensitive information, including user credentials and pet data, should be encrypted during transmission (SSL/TLS) and while stored in the database.
- **API Security:** All backend APIs (Node.js/Express and Flask) should implement proper authentication (OAuth2, API tokens) and authorization to prevent unauthorized access.

  Vulnerability Testing: The app should undergo regular penetration testing to identify and mitigate security vulnerabilities, especially with respect to user data and the AI model.

4.4 Reliability Requirements

- **App Availability:** The app should maintain 99.9% uptime, ensuring continuous availability for the users.
- **Error Handling:** The app should display user-friendly error messages for any failures (e.g., failed AI model prediction, connectivity issues) and should gracefully handle them without crashing.
- **Data Backup:** User data, including pet profiles and health records, should be backed up daily to prevent loss in case of system failure.
- **API Redundancy:** Ensure that the APIs are deployed with high availability, using load balancing and backup services in case of a server failure.

4.5 Usability Requirements

- **User Interface (UI):** The app should have an intuitive and easy-to-navigate interface with minimal user input required. It should follow Android design guidelines.
- **Accessibility:** The app should be fully accessible for users with disabilities, including screen reader support and text resizing.

4.6 Supportability Requirements

- **Bug Reporting:** Users should be able to report bugs or issues directly from the app through a simple, easy-to-use interface.
- **App Updates:** The app should provide a clear process for users to update to the latest version without data loss.
- **Error Logs:** Detailed error logs should be captured for internal use, allowing the development team to quickly identify and address issues.
- **Backend Monitoring:** Regular monitoring tools should be implemented for the backend APIs to identify any service degradation, slow responses, or potential outages.
- **Maintenance Windows:** Routine maintenance should be planned with minimal disruption to users, ideally outside of peak usage hours.

4.7 User Documentation

- **Getting Started Guide:** A clear, concise guide explaining the app's features, such as AI disease detection, profile management, and meal time notifications, should be available on the welcome screen.
- **Help Section:** A help section should be accessible from the main menu, with FAQs, troubleshooting tips, and a guide on how to use each feature, such as the disease detection model and doctor consultation.
- **AI Model Explanation:** Include a brief description of how the AI disease detection works, including what the model analyzes in the image and how users can interpret the results.

- **Push Notification Setup:** A section explaining how users can manage their notification preferences, including enabling and disabling push notifications for meal times, doctor consultations, etc.
- **Privacy Policy and Terms of Service:** Clear documentation should be provided regarding user data privacy, the type of data collected, how it is used, and user rights.

5. References

1. **Android Development Standards**
    - Android Developers Official Documentation:
      https://developer.android.com/docs
      Provides guidelines and best practices for Android application development, including UI/UX standards, coding conventions, and architectural patterns like MVVM.
    - Google Material Design Guidelines:
      https://material.io/design
      Offers comprehensive guidance on designing intuitive, user-friendly interfaces for Android apps.
2. **Diagramming and Design Tools**
    - Diagramming AI Tool:
      Used to create use case diagrams, class diagrams, ER diagrams, and other visual representations for the system.
      Example Tools: Lucidchart, Draw.io, or similar platforms.
    - UML and Software Modeling Standards:
      Fowler, Martin. *UML Distilled: A Brief Guide to the Standard Object Modeling Language.*
      Used for understanding and designing system structures.
3. **GUI Design and Prototyping**
    - Android Studio Design View:
      Provides a visual editor for creating and refining Android app interfaces.
    - References for GUI snapshots:
      Screenshots from the app interface, highlighting different functionalities like user registration, pet management, and doctor consultation.
4. **Research on Pet Care**
    - Problem Statement Research:
        - S. Yeates, "Advancing Veterinary Diagnostics Using Artificial Intelligence," *Journal of Animal Care Science*, 2023.
          Discusses how AI-powered systems can enhance diagnostics for pet health issues, particularly focusing on image-based disease detection.
        - A. Kumar and P. Singh, "Trends in Pet Care Management: A Data-Driven Approach," *International Journal of Veterinary Research*, 2022.
          Explores the role of technology in modern pet care, including mobile apps for health tracking and doctor consultations.
    - Design Considerations:
        - "Human-Centered Design for Pet Care Applications," *Interaction Design Journal*, 2021.
          Emphasizes the importance of user-friendly interfaces and robust backend systems for pet care apps

- **Tools and Technologies**
- Flask Framework Documentation:
  https://flask.palletsprojects.com
  Used for building the backend, including endpoints for pet disease detection.
- TensorFlow or PyTorch Documentation (for disease detection):
  Resources for integrating image-based disease detection using machine learning models.

Citation Example for Diagrams:

- All diagrams, including use case and class diagrams, were created using Diagramming AI Tool.