



SMART PAW

Final Year Project Report

Submitted by

SUBHAN (571-2019)

HARIS BIN MOHSIN (1516-2020)

M ASAAD IBRAHIM (2602-2021)

Supervisor

SIR ABDUL

RAZZAQUE

In partial fulfilment of the requirements for the degree of
Bachelor of Science in Computer Science
2024

Faculty of Engineering Sciences and Technology

Hamdard University, Main Campus, Karachi, Pakistan

Certificate of Approval



Faculty of Engineering Sciences and Technology

Hamdard Institute of Engineering and Technology
Hamdard University, Karachi, Pakistan

This project “SMART PAW” is presented by SUBHAN, HARIS BIN MOHSIN and M ASAAD IBRAHIM under the supervision of their project advisor and approved by the project examination committee, and acknowledged by the Hamdard Institute of Engineering and Technology, in the fulfillment of the requirements for the Bachelor degree of computer science.

Mr.Razaque
(Project Supervisor)

In-charge FYP-Committee

Mr. Waqar
(Project Co-Supervisor)

Chairman
(Department of Computing)

(Dean, FEST)

Authors' Declaration

We declare that this project report was carried out in accordance with the rules and regulations of Hamdard University. The work is original except where indicated by special references in the text and no part of the report has been submitted for any other degree. The report has not been presented to any other University for examination.

Dated:

Authors Signatures:

SUBHAN

HARIS BIN MOHSIN

M ASAAD IBRAHIM

Plagiarism Undertaking

We, SUBHAN , HARIS BIN MOHSIN and M ASAAD IBRAHIM, solemnly declare that the work presented in the Final Year Project Report titled SmartPaw has been carried out solely by ourselves with no significant help from any other person except few of those which are duly acknowledged. We confirm that no portion of our report has been plagiarized and any material used in the report from other sources is properly referenced.

Dated:

Authors Signatures:

SUBHAN

HARIS BIN MOHSIN

HARIS BIN MOHSIN

Acknowledgments

Praise be to Allah, Lord of Worlds.

Also would like to mention our Project Supervisor, who helped and supported us during that 1-year rollercoaster ride.

Document Information

Customer	
Project Title	SmartPaw
Document	Final Year Project Report
Document Version	1.0
Identifier	FYP-041/FL24
Status	Final
Author(s)	SUBHAN , HARIS BIN MOHSIN and M ASAAD IBRAHIM
Approver(s)	Mr Razaque
Issue Date	24-06-2024

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

Abstract

Keywords:

Pet care made smarter with AI. Pet owners often forget feeding times or miss early signs of illness. That's where Smart Paw steps in — combining tech with compassion. It helps owners stay on top of health, meals, and vet access. Pets stay healthy, owners stay happy. Everyone wins.

We saw the struggle and built “**Smart Paw**”, a digital buddy for pet care. It detects disease from images using AI, reminds users of feeding schedules, and connects them with vets nearby.

We use powerful tools like **image classification models** for early disease signs. Smart Paw analyzes pet photos and flags potential health issues. The app also gives meal alerts and lets users log health events.

The AI model is trained on labeled pet health images. It gets smarter with time, improving prediction accuracy.

Users can explore:

- Pet & owner profiles
- Disease scan via image upload
- Meal tracking and reminders
- Vet search by location
- A vibrant pet care community with tips and blogs

Mobile App Front End: Android Java / Kotlin / xml

Back End: FastAPI + Node.js

It's a full-featured app to simplify pet parenting with smart, responsive features — all in your pocket.

Table of Contents

SMARTPAW AI POWERED PET DISEASE DETECTION	1
Certificate of Approval	2
Authors' Declaration	3
Plagiarism Undertaking	2
Acknowledgments	3
Document Information.....	4
Abstract	5
Table of Contents	6
List of Figures.....	9
CHAPTER 1	10
INTRODUCTION	10
USE CASE:	12
CHAPTER 2	13
RELEVANT BACKGROUND & DEFINITIONS	13
CHAPTER 3	16
LITERATURE REVIEW & RELATED WORK	16
Comparative analysis chart	20
CHAPTER 4	21
PROJECT DISCUSSION	21
Chapter 5	27
IMPLEMENTATION	27
.8 Big picture	28

Chapter 5	36
EXPERIMENTAL EVALUATIONS & RESULTS	36
Models Training	36
CHAPTER 6	43
CONCLUSION AND DISCUSSION	43
A0. PROJECT REGISTRATION FORM	47
A1A. PROJECT PROPOSAL AND VISION DOCUMENT	48
Comparative analysis chart	60
A1B. COPY OF PROPOSAL EVALUATION COMMENTS BY JURY	62
A2. REQUIREMENT SPECIFICATIONS	63
A3. DESIGN SPECIFICATIONS	80
Database Design	92
Sequence Diagram 1: Data Upload and Preparation	96
A4. OTHER TECHNICAL DETAIL DOCUMENTS	104
1. Test Cases Document	104
TEST CASES	104
Test Case 1: Successful Data Upload	104
Test Case 2: Data Upload with Incorrect Format	104
• Description: Click on the "Train Model" button after uploading data.	105
Test Case 4: Model Training without Data Upload	105
• Description: Attempt to train the model without uploading any data.	105
Test Case 5: Forecasting Sales for Different Time Periods	105
• Description: Attempt to forecast sales without training the model.	105
Test Case 7: Handling Unexpected Input	106
a. UI/UX Detail Document	107

1. User Inputs:	107
2. Action Buttons:	107
3. Graphical Visualization	108
4. User Interface Design	108
b. CODING STANDARDS	114
2. Naming Conventions:	114
3. Comments:	115
4. Error Handling:	115
5. File Organization	115
6. Version Control	116
7. Performance:	116
8. Security:	116
9. Review and Testing:	116
c. USER DOCUMENT	117
d. Project Policy Document	119
A5. FLYER & POSTER DESIGN	123
A6. COPY OF EVALUATION COMMENTS	124
COPY OF EVALUATION COMMENTS BY SUPERVISOR FOR PROJECT – I MID SEMESTER EVALUATION	124
A7. COPY OF EVALUATION COMMENTS BY SUPERVISOR FOR PROJECT – II MID SEMESTER EVALUATION	125
A8. MEETINGS' MINUTES & Sign-Off Sheet	126
A9. FYP fortnightly sign-up sheet	135

List of Figures

Figure No	Description	Page No.
Figure 1	Use Case	12
Figure 2	Proposed System Architecture	27
Figure 3	Big picture	28
Figure 4	Software requirements Specification (SRS)	30
Figure 5	Gant Chart	32
Figure 6	Use Case Diagram	72
Figure 7	System Level Architecture	86
Figure 8	Software Architecture	89
Figure 9	Design Class Diagram	91
Figure 10	Sequence Diagram	93
Figure 11	State Transition Diagram	94
Figure 12	DFD Level 1	95
Figure 13	Sequence Diagram 1	96
Figure 14	Sequence Diagram 2	97
Figure 15	Sequence Diagram 3	99
Figure 16	State Diagram 1	101

In Scope:

- A user-friendly mobile app where pet owners can enter their pets' health information and receive graphical health insights and AI-driven disease predictions based on both uploaded images and textual symptom descriptions. The system administrator will manage user accounts, monitor app performance, and make data-driven improvements.
- The application will be a mobile-first solution, accessible on Android/iOS platforms, with one system administrator initially responsible for onboarding new users, including pet owners and veterinary professionals.
- Users will be able to search and retrieve health data by entering relevant filters or keywords such as pet breed, age, visible symptoms, or known conditions.
 - The system will allow users to upload pet images and/or describe symptoms via text. The app's AI models will analyze these inputs and return possible disease predictions across multiple categories such as dermatological, respiratory, or behavioral issues.
- Users will be able to view confidence scores for AI predictions and analyze trends via interactive visualizations and graphs.
- The application will also provide the ability to generate various types of reports, such as symptom logs, meal schedules, diagnosis history, and vet visit summaries.

Not in Scope:

- The project will focus only on disease detection using image-based and text-based AI models and will not include live veterinary consultations, treatment prescriptions, or emergency care functionalities.

CHAPTER 2

RELEVANT BACKGROUND & DEFINITIONS

2.1 Stakeholders

The stakeholders for the **Smart Paw** application include pet owners, veterinarians, the development team, and system administrators. Pet owners are the primary users, using the app to manage pet health and receive AI-based disease predictions. Veterinarians may review or validate predictions. The development team is responsible for building and maintaining the application. Additional stakeholders include animal welfare organizations, veterinary clinics, and pet health regulatory bodies.

2.2 Operating Environment

The Smart Paw app is an **Android-based mobile application** developed in **Java**. It runs on smartphones with Android 8.0 (Oreo) or higher. It requires a **stable internet connection** for core functionality, including real-time AI predictions, user authentication, and push notifications. The backend integrates cloud-based services for AI processing and database storage.

2.3 System Constraints

- **Software Constraints:** The system must validate user input (e.g., email), secure access with authentication, and protect against misuse such as spam, AI manipulation, or data breaches. Firebase Authentication and Firestore rules will help enforce security.
- **Hardware Constraints:** Users need a functioning Android device with a supported OS version, minimum 2 GB RAM, and an active internet connection.
- **Cultural Constraints:** The system is currently available only in the **English (UK)** language, requiring users to be proficient in reading English.
- **Legal Constraints:** All code, AI models, and system architecture are the intellectual property of the Smart Paw development team. Proper data use policies must be followed.
- **Environmental Constraints:** The AI model (hosted via a Python-based TensorFlow backend) may be deployed on a cloud server and accessed remotely through secure APIs.
- **User Constraints:** The system assumes an admin will register and authorize users (vets, content moderators). Regular users will register and log in using Firebase Authentication.

2.4 Assumptions & Dependencies

As a mobile application, it is assumed that:

- Users will have internet-enabled Android devices.
- The app uses **Firestore** for:
 - Authentication
 - Firestore database
 - Push notifications (via Firebase Cloud Messaging)
- AI disease detection uses **TensorFlow (Python)** models for:
 - **Image-based predictions** (pet skin, wounds, infections, etc.)
 - **Text-based predictions** (symptom descriptions or health logs)

2.5 Hardware Interfaces

- **Client Devices:** Android smartphones with at least 2 GB RAM, 1.8 GHz processor, and Android 8.0+.
- **Server Requirements:** Cloud-based server or virtual machine (VM) to run TensorFlow-based prediction services.
- **Network:** Stable Wi-Fi or mobile data required for real-time interaction between frontend and backend services.

2.6 Software Interfaces

- **Mobile App (Frontend):** Android (Java)
- **Backend AI Services:** Python (TensorFlow), exposed via REST API
- **Push Notifications:** Firebase Cloud Messaging (FCM)
- **Database:** Firestore
- **Authentication:** Firebase Authentication
- **Libraries & Tools:** TensorFlow (image + text models), Retrofit (API calls), Picasso/Glide (image handling), JSON/GSON for data exchange

All services communicate through HTTPS-secured APIs. Prediction results, health logs, images, and user metadata are exchanged between client and server in JSON format.

2.7 Interactions for Communications

- The app requires internet access via Wi-Fi or mobile data.
- It connects to Firebase services (Firestore, FCM, Authentication) and external TensorFlow-based APIs for disease predictions.
- Secure communication (SSL/TLS) is enforced for all network traffic.

Environmental Needs:

- The app functions on Android devices and can be accessed anywhere with internet connectivity.
- AI predictions depend on cloud access for processing image and text inputs.
- Minimum age to use the app is **15+**.
- Language supported: **English (UK)** only.

CHAPTER 3

LITERATURE REVIEW & RELATED WORK

Literature Review

The Smart Paw application is positioned at the intersection of veterinary care, artificial intelligence, and mobile technology. While AI has made substantial progress in human healthcare applications, its adaptation in the domain of pet health remains relatively limited. This literature review focuses on prior research related to AI-based health monitoring, disease prediction using deep learning models, and the use of image and text-based data in diagnostics — all of which align with Smart Paw’s objectives.

AI in Human and Veterinary Health:

The application of machine learning in healthcare has shown promising results in early disease detection, image classification, and diagnostic decision-making. Numerous studies have validated the effectiveness of CNNs for analyzing medical images and LSTM-based architectures for processing time-series or symptom-based textual inputs. However, veterinary medicine lacks comprehensive solutions that integrate both image and text data for intelligent disease prediction.

Deep Learning in Image-Based Diagnosis:

Convolutional Neural Networks (CNNs) have proven successful in medical imaging tasks such as skin lesion detection, X-ray analysis, and retinal scanning. The success of CNNs in human healthcare imaging sets a strong precedent for their use in veterinary imaging. Studies like Cheplygina et al. (2019) emphasize that the accuracy of disease classification can significantly improve when high-quality annotated datasets are used. Smart Paw leverages this by allowing pet owners to upload images of visible symptoms for real-time AI analysis.

Text-Based Symptom Interpretation:

Text-based symptom logging, combined with Natural Language Processing (NLP), has been effective in human self-diagnosis platforms like Babylon Health and Ada. LSTM models, due to their

sequence-handling capability, are commonly used for interpreting health-related text input. While these models are well-explored in human health systems, their adoption in pet care apps is rare. Smart Paw seeks to bridge this gap by enabling users to describe symptoms in plain language for AI-based health assessments.

Multimodal Approaches in Healthcare AI:

Few systems combine image and text-based inputs in a unified diagnostic model. Research shows that multimodal AI — which combines multiple data types — tends to outperform single-input models due to richer context and better feature extraction. This multimodal strategy is core to Smart Paw's approach, allowing it to analyze both user-submitted text and images for more accurate pet disease predictions.

4. Related Work and Gap Analysis

Unexplored Area:

There is limited research on AI-driven pet healthcare systems that integrate deep learning for real-time disease prediction based on user-generated data. Most existing studies and tools are focused on human medicine or commercial forecasting.

Underexplored Area:

Current pet care apps mainly offer features like vaccination reminders or appointment scheduling. Few systems provide intelligent diagnosis support using deep learning models, especially those trained on veterinary image data and symptom text.

Methodological Gap:

There is a lack of comparative studies analyzing multiple deep learning architectures (e.g., CNNs for images and LSTMs for text) in the context of veterinary disease prediction. Smart Paw proposes a solution that evaluates and combines both.

Contextual Gap:

Most studies are regionally constrained, focusing on specific breeds or conditions. Smart Paw aims to be globally accessible and inclusive of different species, breeds, and environments.

Temporal Gap:

Many existing models are trained on outdated or static datasets. Smart Paw incorporates real-time user input and dynamic learning, which allows it to adapt to new health trends or emerging diseases in the pet population.

Comparative analysis chart:

Comparative Analysis Chart:
Smart Paw vs. Existing Solutions

Feature/Criteria	Traditional Pet Care	Existing Pet Apps (Non-AI)	Smart Paw (Proposed)
Disease Detection	Manual observation by owner or vet	Symptom-based logs only	AI-based prediction (image + text)
Input Types	Owner descriptions only	Text input only	Image + Text (Multimodal)
AI Integration	None	None or very limited	CNN (image) + LSTM (text)
Real-time Prediction	No	No	Yes, instant output
User Target	Local / in-person only	General pet owners	Pet owners + Vets (globally)
Platform Type	Physical visits, paperwork	Mobile/web, static data	Mobile App (Android), Cloud-based
Data Personalization	No	Basic profile only	Personalized prediction & history
Push Notifications	No	Basic reminders	Custom alerts via Firebase
Accessibility	Depends on vet availability	App access with manual input	24/7 availability with AI
Language Support	Local spoken language	English, sometimes multi-lang	English (UK)

Conclusion:

Smart Paw introduces a **new standard** in pet care by integrating **deep learning models**, handling **multimodal data**, and enabling **real-time health monitoring**. It bridges the gap between traditional care and modern digital tools, providing scalable, intelligent, and accessible support to pet owners and veterinarians alike.

CHAPTER 4 – PROJECT DISCUSSION

4.1 Software Engineering Methodology

The Smart Paw application follows the **Evolutionary Prototyping** methodology, which enables the development team to iteratively improve the system based on ongoing feedback from stakeholders, pet owners, and veterinary consultants. This ensures that the final Android application, AI prediction system, and backend services evolve to meet functional and non-functional requirements effectively.

4.2 Project Methodology

1. Research-Based Prototyping

- **Purpose:** This stage focused on exploring the most effective AI architectures for pet disease detection using multimodal data—images and natural language symptom descriptions.
- **Approach:** Multiple models were experimented with, particularly:
 - **Efficient Net** for image-based classification
 - **DistilBERT** for symptom-based text interpretation
- **Outcome:** This phase helped assess the feasibility of combining transformer-based text classification with CNN-based image analysis to produce more accurate disease predictions.

2. Software-Based Prototyping

- **Purpose:** Build and refine a fully functional Android application that utilizes the trained AI models.
- **Frontend:** Developed in Java using XML layouts, integrating **Lottie animations** for a modern user experience.
- **Backend:** Developed using **NestJS**, exposing secure APIs for predictions and user interactions.
- **Notifications:** Implemented with **Firebase Cloud Messaging** for pet health alerts and appointment reminders.
- **AI Integration:** The EfficientNet and DistilBERT models were deployed via API endpoints and integrated into the mobile app to enable real-time prediction.

4.3 Phases of the Project

1. Project Initiation

- **Requirements Gathering:** In collaboration with vets and users, we defined key app features like image upload, symptom logging, AI prediction, and notifications.
- **Feasibility Study:** We evaluated cloud resource needs, on-device capabilities, and model hosting strategies.
- **Proposal:** The project scope, timeline, deliverables, and tech stack were finalized.

2. Planning

- **Milestone Planning:** Set deadlines for model training, frontend integration, backend APIs, and user testing.
- **Risk Management:** Considered issues like model bias, data privacy, and device compatibility.
- **Role Assignment:** Distributed tasks among mobile developers, AI engineers, and QA testers.

3. Design

- **System Architecture:** Designed a modular system using:
 - Android App (Java/XML)
 - Backend APIs (NestJS)
 - AI servers (Python models via Flask endpoints)
 - Firebase integration for authentication and push notifications
- **UI/UX:** Built user flows for pet profile management, image/symptom submission, and prediction results.
- **Model Design:**
 - EfficientNet: Fine-tuned on veterinary image datasets
 - DistilBERT: Trained on text descriptions of pet symptoms

4. Development

- **Frontend:** Developed Android app screens with XML, logic in Java, and visual animations using Lottie.

- **Backend:** Used NestJS to create REST APIs, handle user data, and serve prediction requests.

Model Training:

- CNN for images trained on labeled pet disease datasets
- Transformer for symptoms trained on vet-approved symptom text
- **Integration:** All modules were connected through secure API calls. AI models were hosted and served via backend.

5. Testing

- **Unit Testing:** Each component (model prediction, user input, push notification) was tested separately.
- **Integration Testing:** End-to-end testing was performed between app, backend, and AI models.
- **System Testing:** Full tests on different Android devices for layout, speed, and usability.
- **User Acceptance Testing (UAT):** Select users evaluated app effectiveness and provided feedback.

6. Deployment

- **Production Setup:** AI models hosted on server (or cloud instance); NestJS deployed on production backend.
- **Firebase Setup:** Authentication and messaging services were configured for live use.
- **App Launch:** The APK was signed and deployed for user access via Play Store or testing environments.

7. Maintenance & Support

- **Performance Monitoring:** Logs and analytics from backend and Firebase used to track issues.
- **Bug Fixes:** Post-launch issues are resolved with version updates.
- **Support:** FAQ and support channels are made available within the app.

8. Evaluation

- **Model Accuracy:** EfficientNet and DistilBERT were periodically retrained with new user data to improve prediction accuracy.
- **User Feedback:** User input was collected via surveys and in-app reviews.

- **Improvement Cycles:** The app was continuously updated based on real-world use cases and evolving pet health trends.

4.4 Software & Tools Used

Frontend (Android)

- Java (Android Studio)
- XML Layouts
- Lottie for animations
- Firebase Authentication
- Firebase Cloud Messaging (notifications)

Backend

- NestJS (Node.js framework)
- RESTful APIs
- Firebase Realtime Database & Firestore
- JWT-based authentication

AI / Deep Learning

- Python 3
- TensorFlow & Keras
- HuggingFace Transformers (DistilBERT)
- EfficientNet (for image classification)

Version Control

- Git
- GitHub

4.5 Hardware Used

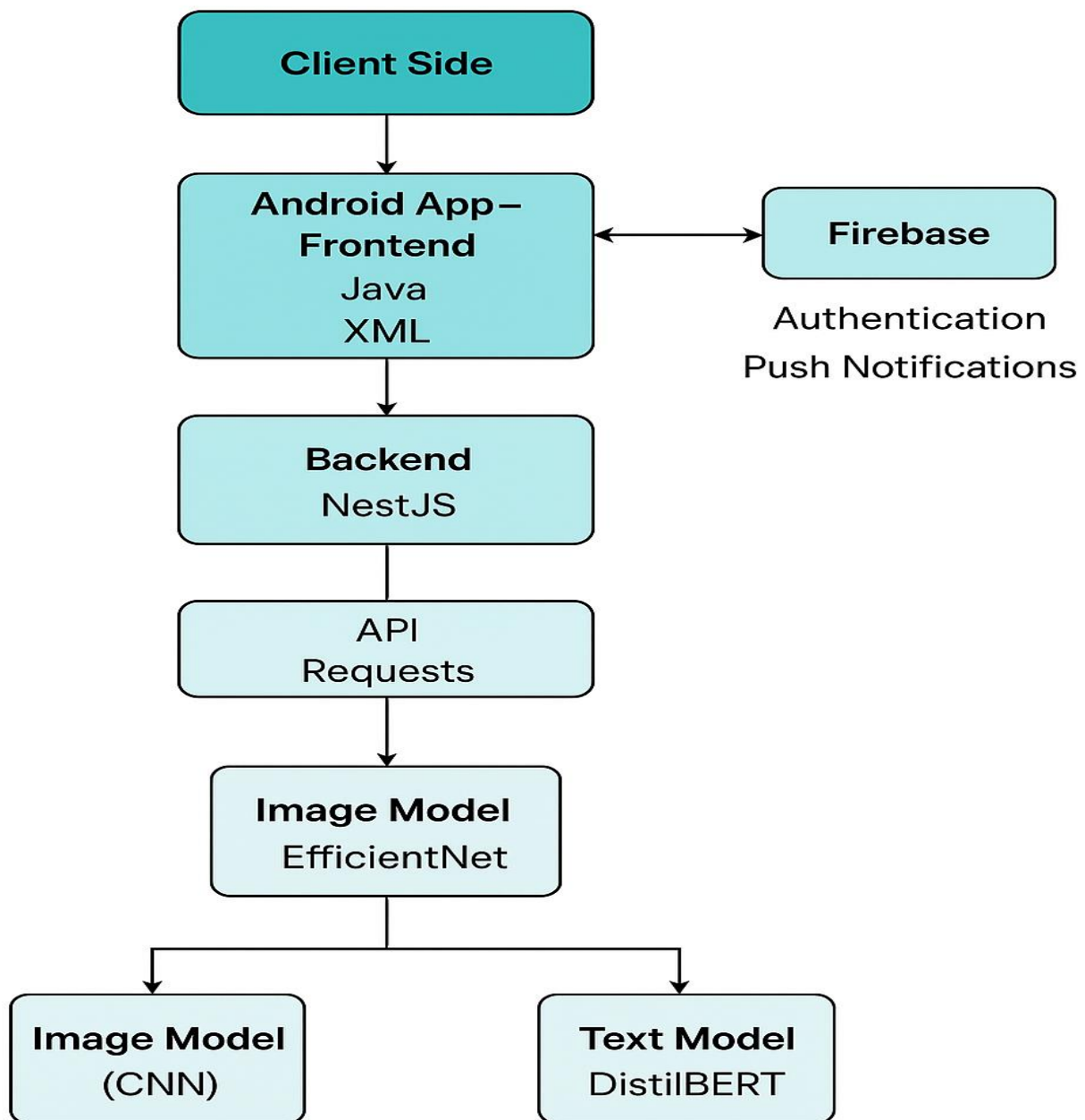
- Development: Laptop with at least 8 GB RAM, Intel i5/i7 CPU, and GPU-enabled environment for training AI models
- Mobile Testing: Android devices with OS version 8.0 and above
- Server Hosting: Cloud or VPS with Python, Node.js, and TensorFlow environment for AI model serving
- Internet: Reliable WiFi required for Firebase syncing, model API calls, and app updates

Chapter 5

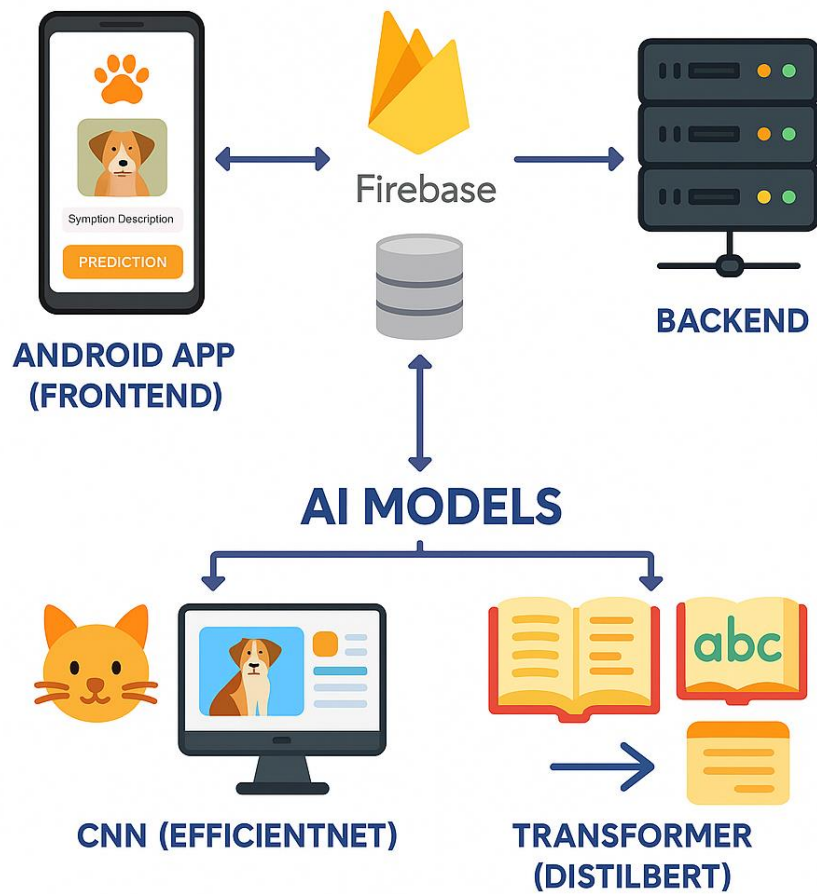
IMPLEMENTATION

4.1 Proposed System Architecture/Design

Proposed System Architecture for Smart Paw



.2 Big picture



Feature	Smart Paw	PetCoach	Pawprint	Pet First Aid
AI Disease Detection	✓ Yes (Image + Text)	✗ No	✗ No	✗ No
Real-Time Prediction	✓ Yes	✗ No	✗ No	✗ No
Push Notifications	✓ Firebase	✓ Yes	✓ Yes	✓ Yes
Vet Resources	✓ Directory	✓ Live Chat	✓ Records	✓ Emergency Info
Platform	Android (Java)	iOS/Android	iOS/Android	iOS/Android

4.2 Functional Requirements

Android app allows users to register pets and upload images & symptoms. Admin can register and authorize new users. AI predicts diseases using image (EfficientNet) and text (DistilBERT) inputs. Firebase sends push notifications for reminders and alerts. Predictions are shown in a simple visual format and stored in history. NestJS backend handles API requests, model integration, and user management.

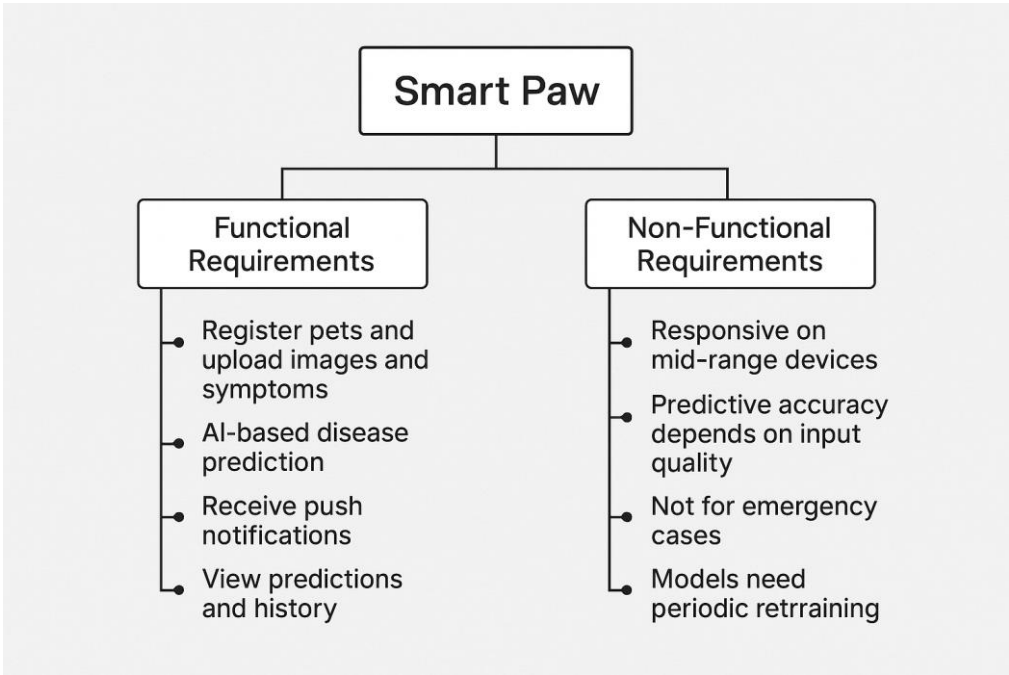
4.3 Non-Functional Requirements

Fast response expected on stable internet and mid-range Android devices. Prediction accuracy depends on input quality (clear images, relevant text). System cannot handle emergency cases or rare diseases with low data support. Models require retraining over time for improved accuracy.

4.4 Safety and Security Requirements

Data Privacy: User and pet data is securely stored and never shared publicly.
Authentication: Firebase Authentication restricts access to authorized users.
Data Transmission: HTTPS encryption secures all communications.
Access Control: Only authorized users and admins can perform actions; AI predictions are read-only and non-invasive.

)



4.2 Testing

1. Unit Testing:

The main emphasis is on testing the individuals' components of the system to check if they are working fine.

2. Integration Testing:

All integrated modules are tested to verify that combined functionality after integration is in a workable state.

3. System Testing:

This is the testing of the whole system as per the requirement. It involves overall requirement specification and all the combined parts of a system are tested.

4. Interface Testing:

The objective behind this testing is to validate the interface according to the business requirement. This testing comprises size of the buttons and input field present on the screen, alignment of all text, tables, and content in the tables. It also validates the menu of the application, after selecting different menu and menu items, it validates that the page does not fluctuate and the alignment remains same after hovering the mouse on the menu or sub-menu.

5. Performance Testing:

The objective of this testing is to ensure the system works under heavy traffic.

6. Compatibility Testing:

This testing will ensure that the application works on different browsers and operating systems and on different android phones. This type of testing also validates whether a web application runs on all versions of all browsers or not.

4.3 Purpose of Testing

This Test Plan documents and tracks the necessary information required in effectively validating and testing the functionalities of the EstateHUB Project. The main objective is to test whether the project satisfies its specified requirements. The testing of this estate management system will be performed by means of unit testing, performance testing, and system testing to ensure that the whole system is functional and working properly.

Test Case 1: User Registration and Login

Test Case ID	TC_01
Test Case Name	User Registration and Login
Objective	Verify successful user registration and login using Firebase
Preconditions	Internet is connected
Test Steps	1. Open app 2. Tap "Register" 3. Enter valid email & password 4. Tap "Sign Up" 5. Reopen app and login
Expected Result	User account is created and login is successful
Actual Result	User navigated to to dashboard
Pass/Fail	Success

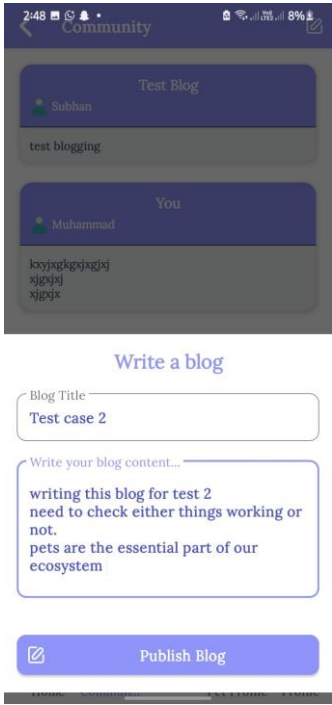
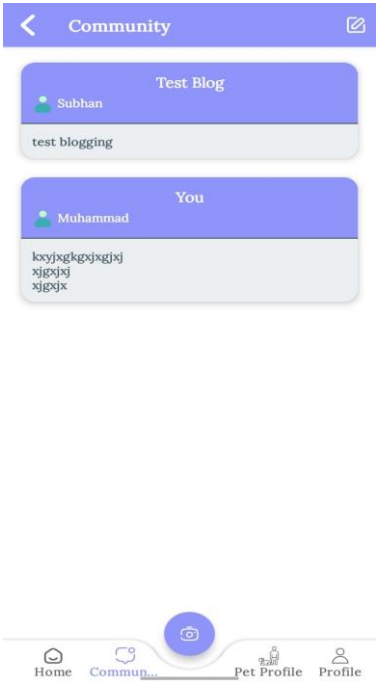
The first screenshot shows the registration form with the following fields: First Name (Subhan), Last Name (Ahmed), Email Address (subhan12@gmail.com), Date of Birth (02/07/2000), Address (Karachi zone east), Password (test123), and Confirm Password (test123). A 'Next' button is at the bottom.

The second screenshot shows the login screen with the following fields: Email Address (subhan123@gmail.com) and Password (test123). A 'Login' button is at the bottom. Below the button is a link: 'Don't have an account? Register'.

The third screenshot shows the app's dashboard. It features a user profile with a 'Hello Subhan' greeting and a 'How can we help?' message. Below this are five pet icons: Dog, Cat, Fish, Rabbit, and Parrot. A 'Popular Blogs' section is visible, followed by a 'View All' link. At the bottom, there are buttons for 'HEALTH TIP', 'FOOD TIME', and 'CONSULTATION'. The bottom navigation bar includes icons for Home, Commu..., Pet Prof..., and Profile.

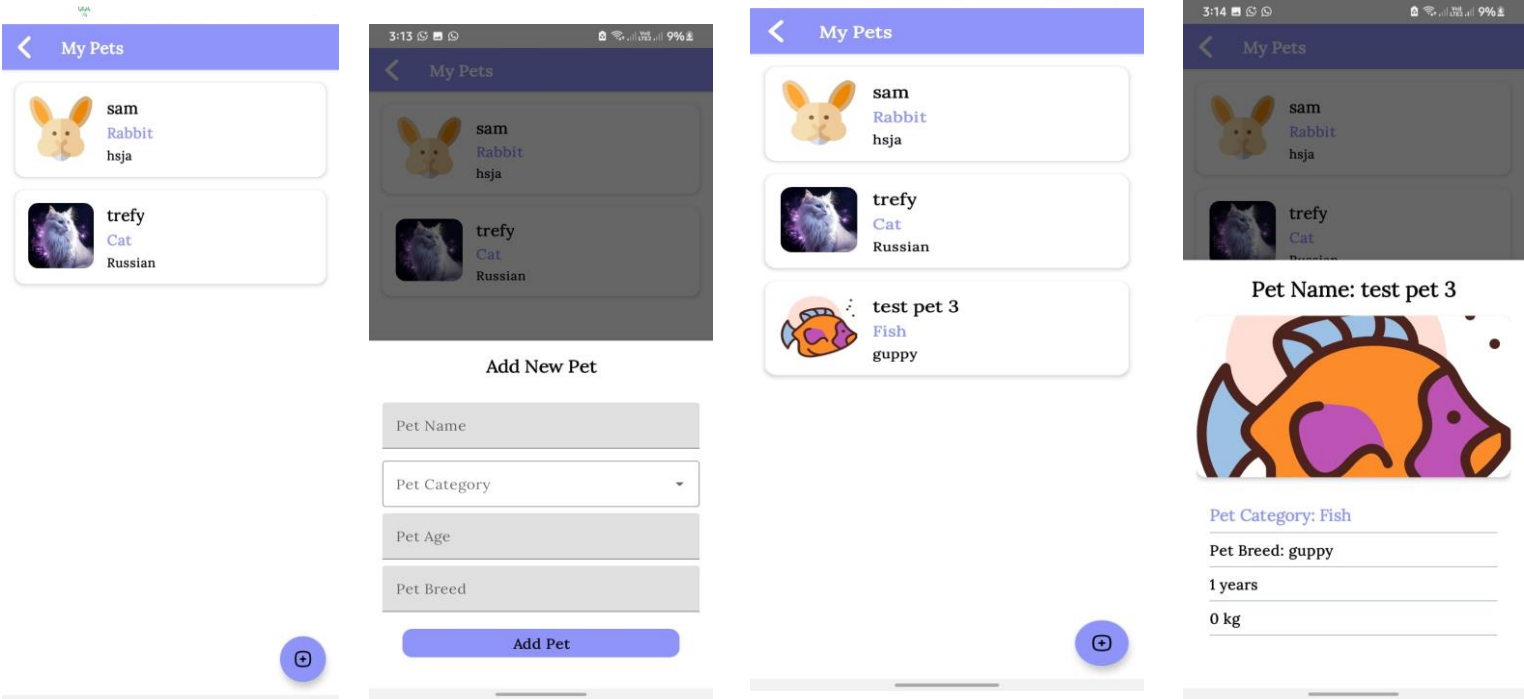
Test Case 2: Upload Blog in Community

Test Case ID	TC_02
Test Case Name	Upload Blog in Community
Objective	Verify image and text upload functionality
Preconditions	User is logged in
Test Steps	1. Tap "Community icon" 2. Add blog title and body 3. Publish blog.
Expected Result	Check your published blog in community fragment
Actual Result	[To be filled]
Pass/Fail	[To be filled]



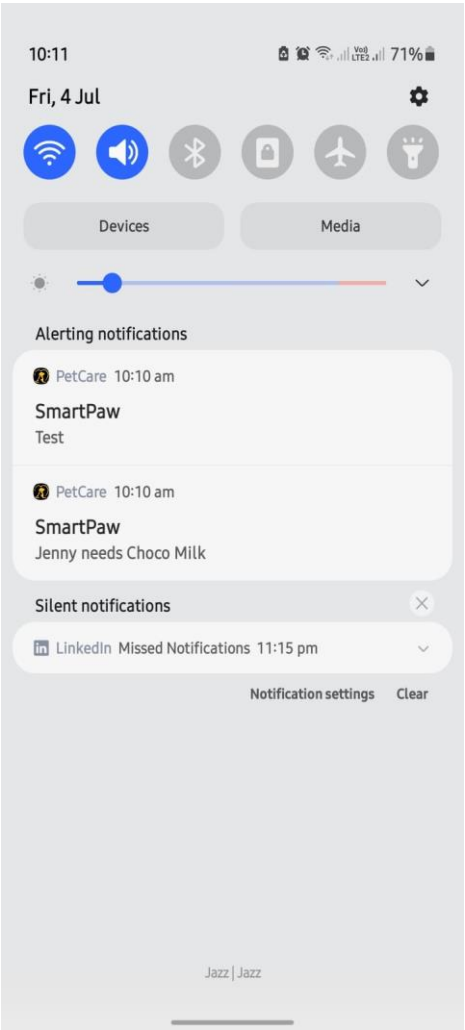
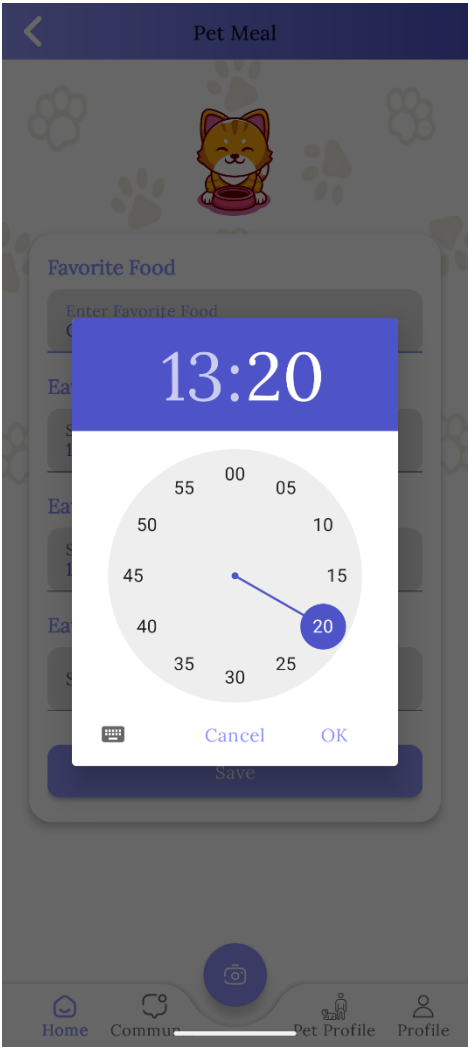
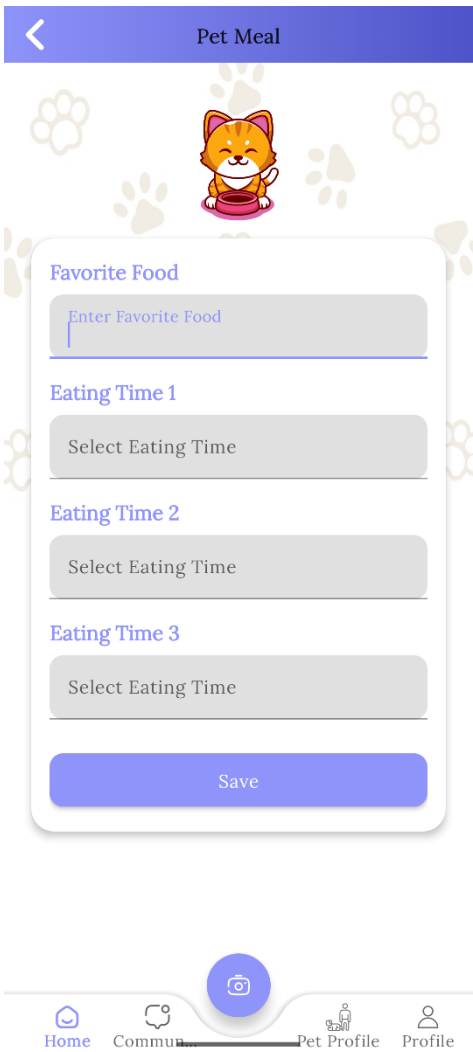
Test Case 3: Add Pet

Test Case ID	TC_03
Test Case Name	Add pet
Objective	Verify either user can add their pets
Preconditions	User is logged in
Test Steps	1. Click pet screen 2. Click add pet button 3. Add pet details 4. Click add pet button 5. Verify to check added pet in list
Expected Result	Added pet into list
Actual Result	Added
Pass/Fail	Pass



Test Case 5: Push Notification Functionality

Test Case ID	TC_05
Test Case Name	Push Notification Functionality
Objective	Check if push notifications are received via Firebase
Preconditions	Notification is scheduled
Test Steps	1. Wait for reminder (e.g., vet visit/feed time) 2. Check notification panel
Expected Result	Notification is received on time
Actual Result	Notification didn't received on time
Pass/Fail	fail



Chapter 5

EXPERIMENTAL EVALUATIONS & RESULTS

Models Training

1. Text Model Training

The screenshot shows the Visual Studio Code interface with the 'requirements.txt' file open in the editor. The file lists the following dependencies:

```
1 torch
2 torchvision
3 tqdm
4 scikit-learn
5 Pillow
6 numpy
7 fastapi
8 uvicorn
9 pydantic
10 python-multipart
```

The terminal window at the bottom shows the command `python text_train.py` being executed. The output indicates that the requirements are already satisfied and provides a notice about a new release of pip (25.0.1 to 25.1.1).

The screenshot shows the Visual Studio Code interface with the 'text_train.py' file open in the editor. The file contains the following code:

```
1 import torch
2 import torch.optim as optim
3 import torch.nn as nn
4 from torch.utils.data import DataLoader
5 from transformers import DistilBertTokenizer
6 from text_dataset_loader import load_dataset
7 from text_model import PetDiseaseTextClassifier
8 from collections import Counter
9
10 # Set device (Use GPU if available)
11 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

The terminal window at the bottom shows the command `python text_train.py` being executed. The output indicates that the requirements are already satisfied and provides a notice about a new release of pip (25.0.1 to 25.1.1). The training process is shown, including the training species mapping and training disease mapping.

The screenshot shows a VS Code editor with the following components:

- Left Panel (File Explorer):** Displays the project structure for 'PET-DISEASE-PREDICTION-SYSTEM'. The 'requirements.txt' file is selected.
- Editor:** Shows the contents of 'requirements.txt':

```
1 torch
2 torchvision
3 tqdm
4 scikit-learn
5 Pillow
6 numpy
7 fastapi
8 uvicorn
9 pydantic
10 python-multipart
```
- Terminal:** Shows the output of running 'python image_train.py'. It reports the number of images loaded from the dataset and the disease class distribution:

```
(venv) PS D:\SmartPaw AI\Pet-Disease-Prediction-System> python image_train.py
Loaded 2211 images from dataset\train
Loaded 397 images from dataset\val
Loaded 295 images from dataset\test
Disease Class Distribution: {
  "0": 117,
  "1": 102,
  "2": 135,
  "3": 155,
  "4": 160,
  "5": 120,
  "6": 143,
  "7": 160,
  "8": 166,
  "9": 90,
  "10": 99,
  "11": 140,
```

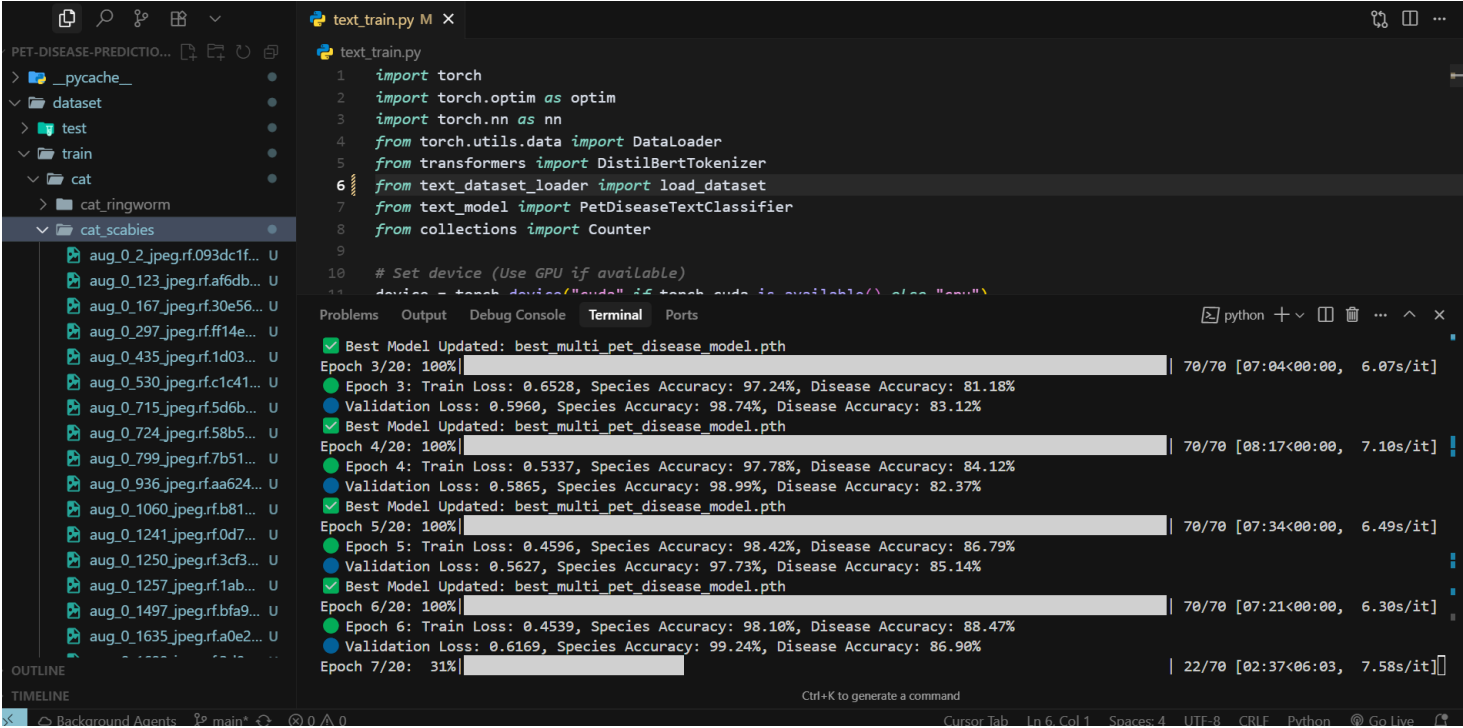
The screenshot shows a VS Code editor with the following components:

- Left Panel (File Explorer):** Displays the project structure for 'PET-DISEASE-PREDICTION-SYSTEM'. The 'requirements.txt' file is selected.
- Editor:** Shows the contents of 'requirements.txt':

```
1 torch
2 torchvision
3 tqdm
4 scikit-learn
5 Pillow
6 numpy
7 fastapi
8 uvicorn
9 pydantic
10 python-multipart
```
- Terminal:** Shows the output of running 'python image_train.py'. It displays the training progress for 12 epochs, including loss, species accuracy, and disease accuracy:

```
'flea_allergy': 4, 'Dog_Ringworm': 5, 'Dog_Scabies': 6, 'Healthy_Dog': 7, 'Hotspot': 8, 'Aeromoniasis Bacterial diseases': 9, 'Bacterial disease gill': 10, 'Bacterial Red disease': 11, 'Fungal Saprolegniasis diseases': 12, 'Healthy Fish': 13, 'Parasitic diseases': 14, 'Viral White disease diseases tail': 15}
Epoch 1/15, Loss: 105.6937, Species Accuracy: 84.22%, Disease Accuracy: 32.81%
Epoch 2/15, Loss: 29.1461, Species Accuracy: 99.84%, Disease Accuracy: 87.81%
Epoch 3/15, Loss: 5.3728, Species Accuracy: 100.00%, Disease Accuracy: 99.84%
Epoch 4/15, Loss: 2.2268, Species Accuracy: 100.00%, Disease Accuracy: 100.00%
Epoch 5/15, Loss: 1.4791, Species Accuracy: 100.00%, Disease Accuracy: 100.00%
Epoch 6/15, Loss: 1.1196, Species Accuracy: 100.00%, Disease Accuracy: 100.00%
Epoch 7/15, Loss: 0.9053, Species Accuracy: 100.00%, Disease Accuracy: 100.00%
Epoch 8/15, Loss: 0.7636, Species Accuracy: 100.00%, Disease Accuracy: 100.00%
Epoch 9/15, Loss: 0.6641, Species Accuracy: 100.00%, Disease Accuracy: 100.00%
Epoch 10/15, Loss: 0.5955, Species Accuracy: 100.00%, Disease Accuracy: 100.00%
Epoch 11/15, Loss: 0.5159, Species Accuracy: 100.00%, Disease Accuracy: 100.00%
Epoch 12/15, Loss: 0.4670, Species Accuracy: 100.00%, Disease Accuracy: 100.00%
```

2. Image Model Training



CHAPTER 6

CONCLUSION AND DISCUSSION

6.1 Strengths of this Project

Accurate Disease Predictions

SmartPaw employs a dual-model architecture: a Convolutional Neural Network (EfficientNet) for analyzing images of pet symptoms and a Transformer-based model (DistilBERT) for interpreting textual symptom descriptions. This hybrid approach significantly enhances the accuracy of disease detection across different breeds and conditions.

Real-Time Analysis

The app supports real-time image capture and symptom input, enabling pet owners to receive instant predictions without visiting a veterinary clinic. This immediate feedback is vital in emergencies or for early-stage interventions.

User-Centered Mobile Experience

SmartPaw features an intuitive and responsive UI designed for Android devices. Users can navigate through their pet profiles, health tips, and AI predictions with ease. No technical expertise is required, making it accessible to all pet owners.

Integrated and Scalable Architecture

The system integrates Android frontend with FastAPI-powered backend endpoints deployed on AWS. It ensures smooth communication between the app and the AI models using Python libraries like Torch, Transformers, and FastAPI, enabling efficient, secure, and scalable performance.

Health Management Ecosystem

Beyond disease detection, SmartPaw offers a comprehensive care solution:

- Pet meal reminders using Firebase notifications
- Health tips for prevention and daily care
- Emergency consultation section with access to nearby veterinarians
- Report generation and exportable AI results for follow-up care

6.2 Limitations and Future Work

Current Limitations

- Prediction accuracy depends on the quality and clarity of the input image and user-provided symptom descriptions
- AI models are not yet capable of handling very rare diseases due to limited training data
- The app requires a stable internet connection for communicating with the backend FastAPI services
- User profile information is currently static and not editable

Future Enhancements

- Expand the training datasets with more annotated pet images and detailed veterinary symptom data
- Introduce multilingual support to make the app more globally accessible
- Enhance the consultation section with chat or video call integration with vets
- Add a feature to track past symptom entries and AI results for longitudinal health tracking
- Periodically fine-tune both CNN and Transformer models using real-world user submissions
- Build a dashboard for vets to receive predictions and reports shared by users

6.3 Potential Challenges and Risks

Data Quality and Variability

- Poor lighting or unclear symptom images can reduce the reliability of CNN model predictions
- Vague or overly brief symptom descriptions can lead to low-confidence text model outputs

Model Complexity and Maintenance

- Training and maintaining EfficientNet and DistilBERT models requires GPU resources and careful tuning
- Regular updates and model retraining are necessary to ensure continued accuracy and relevance

System Integration

- Syncing frontend and backend with reliable API communication involves robust error handling
- Handling image uploads, user sessions, and Firebase notifications must remain smooth and crash-free under varying conditions

User Adoption and Trust

- Pet owners may initially be skeptical of AI-based diagnostics
- Building trust through clear explanations, vet-approved suggestions, and consistent accuracy is essential

Security and Compliance

- Pet data, while less sensitive than human data, must still be securely stored and handled
- Compliance with future regulations around AI diagnostics in veterinary practice may be required

Conclusion

Developing SmartPaw has been an enriching and interdisciplinary journey that brought together AI modeling, mobile development, backend engineering, and user-centric design. By combining a lightweight Android app with powerful image and text-based AI models, we've built a tool that empowers pet owners with accessible, real-time health predictions. While the system is already highly functional, continued enhancements and feedback-driven updates will allow SmartPaw to grow into a trusted, intelligent companion in the domain of pet care.

A1A. PROJECT PROPOSAL AND VISION DOCUMENT

1. INTRODUCTION

Smart Paw is an Android app that helps pet owners spot health issues early. It uses two AI models—EfficientNet to analyze photos of symptoms and DistilBERT to read written descriptions. Users take a picture, type in what they see, and get back a diagnosis. The app is built in Java with XML layouts and Lottie animations, connects to a NestJS backend, and sends reminders via Firebase.

2. PROBLEM STATEMENT

- Pet owners may miss subtle signs of illness. Smart Paw's AI models improve accuracy.
- Most apps use only text or images; Smart Paw combines both for a fuller picture.
- Health data can be confusing; our app shows results in clear alerts and stores history.
- Finding a vet quickly isn't always easy; the app includes a list of nearby veterinarians.
- Users need to trust the AI; Smart Paw lets them see how each model performed.
- Pet care should be available anywhere; Smart Paw works offline for reading history and online for new predictions.

3. OBJECTIVE

Create an easy-to-use mobile app that predicts pet diseases. The app will let users upload photos and text, run two AI models, and display results instantly. It will also send reminders, save past cases, and allow admins to manage users and view model accuracy.

4. PROJECT SCOPE

- Photo capture and symptom text input for AI analysis
- Two AI services—one for images, one for text—combined into one result
- Instant pop-up with diagnosis and a saved history users can search
- Bottom navigation with:
 - Community blogs
 - Pet profiles and add-new-pet flow
 - Health tips and feeding reminders
 - Vet directory for consultations
 - User profile and logout

- Firebase for login, data storage, and push notifications
- Admin panel for user management and performance tracking
- Report export of past cases and reminder schedules

5. METHODOLOGY

We use **Evolutionary Prototyping** to build Smart Paw in steps, gather feedback, and improve each release.

Research-Based Prototyping

- Purpose: Try out different AI models and data setups to find what works best.
- Focus: Compare EfficientNet and DistilBERT versions in Python using PyTorch and FastAPI.
- Method: Build quick test versions, tweak parameters, and measure accuracy.
- Benefit: Learn which AI choices give reliable results before full integration.

Software-Based Prototyping:

- Purpose: Build a functional Android application that demonstrates the core Smart Paw features—pet image capture, symptom entry, AI inference, and result display—so stakeholders can interact with a working demo.
- Focus: Iteratively develop and polish the mobile UI (Java/XML, Lottie), backend APIs (NestJS), and integrate the two AI endpoints (FastAPI for EfficientNet and FastAPI for DistilBERT).
- Methodology: Produce an initial APK supporting user registration, pet case submission, dual-model predictions, and notification reminders. Refine based on tester feedback until the app flows smoothly.
- Benefits: Early hands-on use uncovers real-world usability issues, guiding prioritization of enhancements before full-scale release.

.8.1 Project Limitations

The Smart Paw app relies exclusively on two AI services—EfficientNet for image analysis and DistilBERT for text interpretation—so no alternative diagnostic methods are supported in this phase.

3. FEASIBILITY STUDY

Technical Feasibility:

- Confirm that the team's existing Android, NestJS, FastAPI, and PyTorch/Transformers expertise suffices to implement and deploy Smart Paw.
- Verify compatibility of development tools (Android Studio, Node.js, Python libraries) with CI/CD pipelines and target devices.

Data Availability and Quality:

- Assess access to labeled pet health images and veterinarian-verified symptom descriptions for training both models.
- Evaluate dataset size and diversity to ensure models can generalize across breeds and conditions.

Market Feasibility:

- Identify pet-care demand for app-based early diagnosis and compare against competing solutions.
- Survey veterinary clinics, shelters, and pet-owner communities to gauge interest and adoption potential.

Financial Feasibility:

- Estimate costs for AI model hosting (cloud servers), mobile development, and Firebase usage.
- Project possible subscription or service revenue and calculate ROI based on reduced vet visits and improved pet health outcomes.

Operational Feasibility:

- Ensure the app integrates smoothly into pet owners' daily routines and veterinary workflows.
- Confirm that the organization can maintain backend services, update models, and provide user support.

Risk Analysis:**i. Risks Involved:**

Data Quality and Availability

Risk: Insufficient or unbalanced pet health data can lead to unreliable diagnoses.

Mitigation: Source additional datasets, apply data augmentation, and request user feedback to improve data diversity.

Model Overfitting

Risk: EfficientNet or DistilBERT may memorize training samples, failing on new cases.

Mitigation: Use validation splits, regularization techniques, and continuous monitoring of model performance on fresh data.

Integration Challenges

Risk: API mismatches between Android client and FastAPI services could break functionality.

Mitigation: Define clear API contracts, implement versioning, and conduct end-to-end tests on each build.

User Adoption and Feedback

Risk: Pet owners may hesitate to trust AI-based advice.

Mitigation: Incorporate in-app tutorials, vet endorsements, and transparent display of model confidence scores.

Legal and Regulatory Compliance

Risk: Handling pet health data may invoke privacy regulations.

Mitigation: Encrypt data in transit (HTTPS) and at rest, obtain user consent, and adhere to regional data-protection guidelines.

1. Resource Requirement**Hardware Resources:**

- Android devices (8.0+) for development and testing
- Cloud or on-prem servers with GPU support for model inference

Software Resources:

- Android Studio, Node.js, Python 3, FastAPI
- Libraries: PyTorch, Transformers, Firebase SDK
- Version control: Git/GitHub

Financial Resources:

- Budget for cloud hosting, development tools, and potential data licensing
- Funding for ongoing maintenance, user support, and marketing

2. SOLUTION APPLICATION AREAS

Companion Pet Health: Early detection of skin conditions, infections, and behavioral issues via smartphone.

Veterinary Triage: Preliminary screening tool for clinics to streamline in-office visits.

Pet Insurance: Automated case logs for claim validation and risk assessment.

Animal Shelters: Rapid health assessments for large numbers of intake pets.

Tele-veterinary Services: Remote consultations augmented by AI-driven symptom analysis.

1. TOOLS & TECHNOLOGY

Our Smart Paw app uses a compact, high-performance stack optimized for mobile pet diagnostics and reliable notifications:

Frontend (Android)

- Java (Android Studio)
- XML layouts
- Lottie animations

Backend & Hosting

- NestJS (Node.js framework)
- Deployed on AWS EC2 (Linux)

AI / Deep Learning

- Python 3
- FastAPI + Uvicorn for model serving
- PyTorch & TorchVision (EfficientNet)

- HuggingFace Transformers (DistilBERT)
- NumPy, Pillow, scikit-learn

Push Notifications

- Firebase Cloud Messaging

Version Control

- Git
- GitHub

2. RESPONSIBILITIES OF THE THREE-PERSON TEAM

- Data Scientist: Assemble and preprocess image/text datasets; train, validate, and fine-tune EfficientNet and DistilBERT models.
- Full-Stack Engineer: Build the Android app (Java/XML/Lottie), develop NestJS APIs on AWS EC2, integrate FastAPI AI endpoints, and configure Firebase Cloud Messaging.
- QA & DevOps Engineer: Design and run test suites for UI flows, API endpoints, and AI accuracy and push-notification delivery.

• 1. MILESTONES

Milestone 1: Data Collection and Preparation

- Gather a diverse set of pet health images and veterinarian-verified symptom descriptions.
- Clean and label the data, handling missing entries and normalizing formats for model training.

• **Milestone 2: Model Development and Evaluation**

- Implement and train the EfficientNet image classifier and the DistilBERT text classifier on the prepared datasets.
- Measure each model's accuracy, precision, and recall to assess performance.

• **Milestone 3: Model Selection**

- Compare the two AI models on validation data to select the one that delivers the best overall diagnostic accuracy.

• **Milestone 4: Mobile App Implementation**

- Build the Android interface in Java/XML with Lottie animations for symptom capture, input forms, and result display.

- **Milestone 5: AI Integration**

- Connect the selected AI model endpoint (FastAPI on EC2) to the Android app, enabling on-demand inference.

- **Milestone 6: Results Visualization**

- Add UI components to present predictions in alert dialogs and maintain a searchable history of past cases.

- **Milestone 7: Testing and Quality Assurance**

- Perform unit and integration testing on the app, backend APIs, and AI services to validate end-to-end workflows.

- **Milestone 8: Deployment**

- Deploy backend services on AWS EC2, release the Android APK, and configure Firebase Cloud Messaging for reminders.

2. LITERATURE REVIEW

- **Dermatological Conditions:** Studies on CNN-based skin lesion detection in veterinary medicine.
- **Gastrointestinal Disorders:** Research on text-based symptom analysis for pet digestive issues.
- **Respiratory Illnesses:** Applications of image classification to detect respiratory distress in animals.
- **Behavioral Health:** Transformer models applied to owner-reported behavioral symptom logs.
- **Allergy Identification:** Multimodal approaches combining visual signs and text descriptions to diagnose allergic reactions.
- **Dental and Oral Health:** AI methods for detecting dental disease from intraoral images.

Comparative analysis chart:

Aspect	Image-Only Research	Text-Only Research	Existing Pet Apps	Smart Paw
Data Inputs	✔ Image	✔ Text	✔ Owner Notes	✔ Image + Text
AI Models	✔ CNN	✔ Transformer	✗ None	✔ CNN + Transformer
Multimodal Fusion	✗	✗	✗	✔
Real-Time Prediction	✗	✗	✗	✔
User Interface	✗ Research Prototype	✗ Research Prototype	✔ Basic Forms	✔ Mobile App (Android)
Notifications	✗	✗	✔ Reminders	✔ Push Alerts (FCM)
Deployment	✗	✗	✔ App Stores	✔ AWS EC2 + Play Store
Dataset Scale	✗ Small	✗ Small	✗ Unspecified	✔ Custom Multimodal
Extensibility	✗	✗	✗	✔ Modular Design
User Feedback Loop	✗	✗	✔ Ratings/Comments	✔ History & Confidence Scores

.3 RACI Chart

Task / Deliverable	Subhan(571-2019)	Haris(1516-2020)	Asaad(2602-2021)	Sir Abdul Razzaque(Supervisor)
Define AI Models (CNN, Transformer)	A	R	C	I
Dataset Collection & Preprocessing	R	R	C	I
Backend API with NestJS	R	A	R	I
Android App UI (Java/XML)	C	R	A	I
Image & Text Model Integration	A	R	R	I
Firebase Notification Integration	A	R	C	I
AWS EC2 Deployment	R	R	A	I
App Testing & Debugging	R	R	R	I
Documentation (SRS, Report, Diagrams)	A	C	C	R
Final Project Presentation	A	A	A	R

A1B. COPY OF PROPOSAL EVALUATION COMMENTS BY JURY

Hamdard University
Faculty of Engineering Sciences and Technology
Department of Computing

FINAL YEAR PROJECT - PROPOSAL EVALUATION

Topic: Pet Health care Industry
ID: FYP-041/FL24 Project Track: _____
Domain: A2 Evaluation Date: _____
Supervisor Name: Engr. Abdul-Razaque Co-Supervisor Name: _____

Member(s):

Name	CMS ID
<u>Subhan</u>	<u>571-2019</u>
<u>M. Asaad Ibrahim</u>	<u>2602-2021</u>
<u>Harris bin Mohsin</u>	<u>156-2020</u>

"Approved" on 22/10/24
Umm

A2. REQUIREMENT SPECIFICATIONS

1. INTRODUCTION

This section presents an overview of the "SmartPaw – AI-Powered Pet Disease Detection" system. The application is designed to assist pet owners in identifying possible health conditions in their pets through AI-based predictions. It utilizes two separate deep learning models — one image-based (CNN using EfficientNet) and one text-based (Transformer using DistilBERT) — to analyze symptoms either visually or through text descriptions.

SmartPaw offers a mobile application with an easy-to-use interface, allowing users to capture pet symptoms using a camera or by describing them through text. The data is processed through FastAPI endpoints hosted on an AWS EC2 Linux instance, where predictions are generated and displayed to the user in real-time. The application includes additional features such as pet management, health tips, feeding reminders via Firebase notifications, and doctor consultation listings. All administrative and diagnostic functionalities are designed with the user in mind, ensuring a smooth and informative experience.

1.1 Purpose of Document

The Requirement Specification Document (RSD) outlines all critical aspects of the SmartPaw application to ensure a clear understanding among all stakeholders — including developers, testers, and end users. Its purpose is to define the application's goals, scope, functionalities, and constraints in a structured way.

The document covers both **functional requirements** (e.g., symptom input, disease prediction, notification system, blog and doctor module) and **non-functional requirements** (e.g., security, scalability, performance). It also serves as a reference point for testing, validation, and ongoing development, helping the team prioritize tasks and ensure that all components align with the intended user experience. The RSD documents project limitations, technical dependencies, and legal considerations to keep the development process organized, goal-driven, and compliant with best practices.

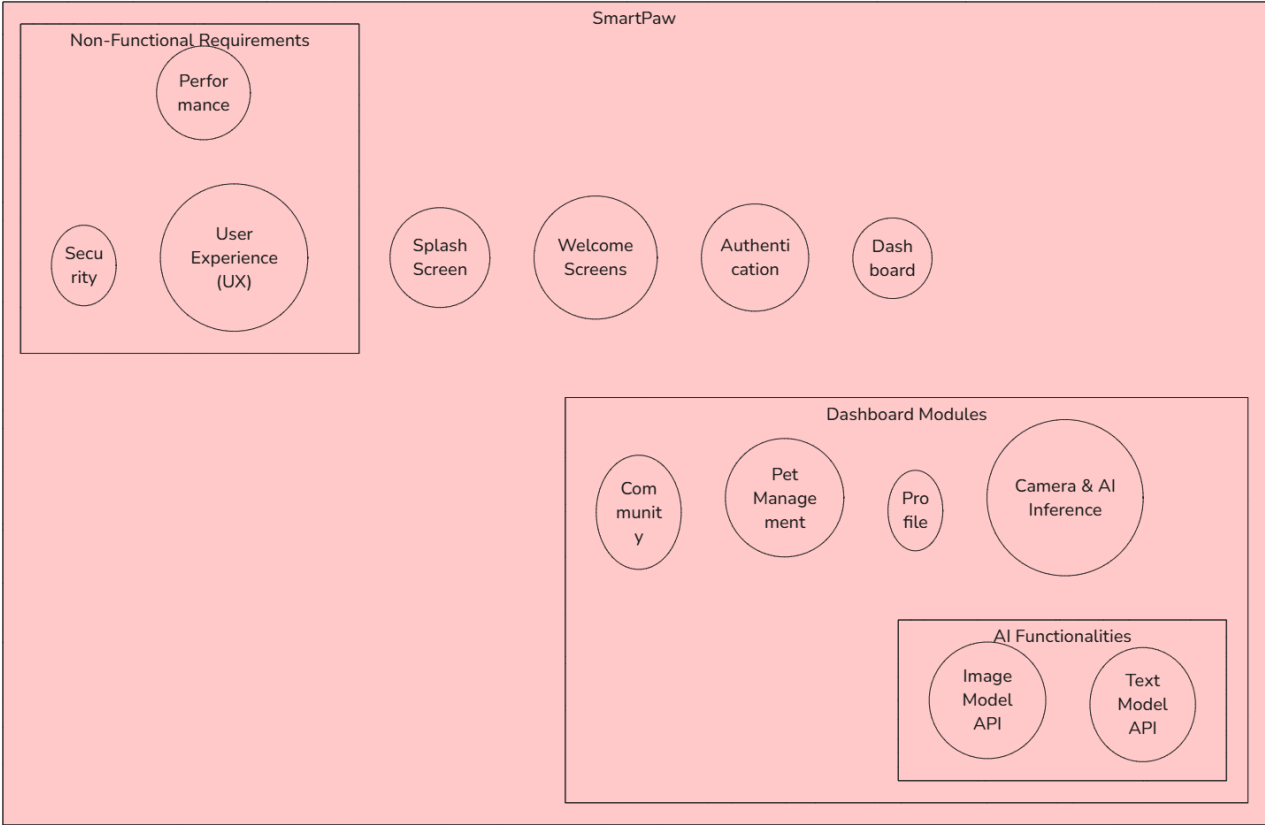
1.2 Intended Audience

This document is intended for all individuals involved in the development and usage of the SmartPaw application. This includes:

- **Pet Owners:** Users seeking early detection of pet diseases through an intuitive AI-powered platform.
- **Veterinary Professionals:** Doctors who can be consulted via the app in emergencies.
- **Developers & QA Testers:** The team responsible for building and testing the application based on outlined requirements.
- **Project Supervisors & Evaluators:** Reviewers assessing the functionality and completeness of the application.

Overall System Description

Software requirements Specification (SRS)



A2. REQUIREMENT SPECIFICATIONS

1.1 Project Background

In today's world, pet health is an area of growing concern. Timely detection of pet diseases plays a crucial role in their well-being. The SmartPaw application is designed to address this need by offering an AI-powered solution for detecting pet diseases based on both visual symptoms and user-described indicators. This innovative mobile application uses deep learning models—one for analyzing images and the other for processing text—to predict possible diseases in pets. By empowering pet owners with early diagnostic tools, SmartPaw contributes to better health outcomes for animals and facilitates proactive care.

1.2 Problem Statement

Pet owners often struggle to recognize early signs of illness in their pets due to a lack of medical knowledge. This can lead to delayed diagnosis and treatment. Moreover, veterinary access is sometimes limited in certain regions or emergencies. SmartPaw aims to solve this by providing an easy-to-use AI-based application that can detect symptoms from images and user inputs, offering early alerts and possible diagnoses, thereby enabling timely care.

1.3 Project Scope

- The application will be an Android-based mobile app with Firebase used only for push notifications.
- It will use two AI models: one for image-based detection and another for text-based symptom analysis.
- Users will register, log in, and navigate through a dashboard with multiple features including a blog community, pet profiles, and consultation options.
- A central floating action button allows users to upload pet symptom images and input symptoms. Results from both models are shown together.
- The app will also include health tips, feeding reminders, and access to doctors for consultation.

1.4 Not in Scope

This project does not involve real-time video analysis, integration with IoT pet health devices, or editable medical histories. It is also not intended to replace professional veterinary advice, only to assist in early detection.

1.5 Project Objectives

- To develop an AI-powered mobile app that detects pet diseases using both image and text-based deep learning models.
- To provide pet owners with early insights into potential illnesses.
- To enable reminders and health tips for better daily care of pets.
- To offer a user-friendly interface for easy navigation and real-time results.
- To bridge the gap between pet owners and emergency vet consultations.

1.6 Stakeholders & Affected Groups

Stakeholders include pet owners who use the app, veterinarians listed for consultation, the app development team, and backend service providers. Indirect stakeholders could include pet product businesses and animal welfare groups.

1.7 Operating Environment

SmartPaw is an Android-based application developed in Java with XML layouts. The backend is powered by NestJS, hosted on an AWS EC2 Linux instance. AI models are deployed using FastAPI. Users need an Android device with an internet connection.

1.8 System Constraints

- Software constraints: The application must validate image and text inputs to prevent misuse or irrelevant data.
- Hardware constraints: Requires Android devices with adequate RAM and camera functionality.
- Cultural constraints: Interface is in English, requiring users to be able to read and understand the language.
- Legal constraints: All data usage must comply with privacy laws. User consent is required for uploading pet images and descriptions.
- Environmental constraints: Internet access is mandatory for core features like AI prediction and doctor consultations.
- User constraints: Only registered users can access disease detection and consultation features.

1.9 Assumptions & Dependencies

The system assumes that users will provide accurate images and symptom descriptions. Predictions are based on predefined model capabilities and may not always be perfect. Backend services depend on AWS uptime and FastAPI endpoints. User authentication and push notifications are managed through Firebase.

Users should be able to:

- Use the interface to upload data (image and text).
- View prediction results from both models.
- Receive push notifications for feeding schedules or health alerts.
- Access medical blogs and doctor listings.

4.1. External Interface Requirements

1. 1.1 Hardware Interfaces

2. • The SmartPaw application can be accessed through any modern smartphone or tablet with Android OS.
- A stable internet connection is required to interact with the backend services hosted on an AWS EC2 instance.
- Users must have devices with functional camera modules, as image capture is a core feature for pet disease detection.

3. 1.2 Software Interfaces

4. SmartPaw interacts with multiple external software components to deliver its functionalities.
- **Firebase:** Integrated to send real-time push notifications such as pet feeding reminders.
- **NestJS (Backend):** REST APIs developed in NestJS power core operations like image/text diagnosis and data management.
- **FastAPI (AI Services):** Two separate FastAPI endpoints manage prediction responses for image-based and text-based input using pre-trained AI models.
- **Android (Frontend):** Built in Java with XML layouts, the app features intuitive UI, fragment navigation, and Lottie animations.
- **AI Libraries Used:** PyTorch, Transformers, FastAPI, NumPy, TorchVision, Pillow, and scikit-learn support image and text prediction workflows.

These components exchange data through secure API calls. Each prediction request involves sending

5. 1.3 Communications Interfaces

6. SmartPaw relies on an internet connection (Wi-Fi or mobile data) for the mobile app to communicate with cloud-hosted backend services.

All data transfers between the app and server use HTTPS to ensure secure communication.

Notifications are delivered via Firebase Cloud Messaging (FCM).

7. 2. System Functions / Functional Requirements

8. 2.1 System Functions

9. • The SmartPaw application is an AI-powered Android app for pet disease detection and health assistance.
- Users must register through a sign-up process to access app functionalities.
 - Authenticated users can manage their pet profiles and consult veterinary professionals.
 - The floating action button (FAB) allows users to capture an image and enter symptoms for diagnosis.
 - The backend handles image and text inputs via two separate AI models for disease detection.
 - Predictions are displayed using an alert dialog with model confidence and disease details.
 - The system sends feeding reminders through scheduled Firebase push notifications.
 - Users can view health tips, manage pet food schedules, and interact with blog content in the community section.

10. 2.2 Use Cases

11. SmartPaw is designed to serve a variety of user needs across pet care, disease diagnosis, and professional consultation.

13. 1. System

Role: Acts as the core framework handling input, processing, AI predictions, and communication.

Responsibilities:

- Process captured images and textual symptom descriptions.
- Route data to appropriate AI services (image/text).
- Authenticate users and authorize their actions.
- Generate and present prediction results.
- Deliver push notifications and manage data flow.

14. 2. Pet Owners (Users)

Role: Main users who interact with the app to diagnose pet health issues and receive recommendations.

Responsibilities:

- Register and log into the application.
- Add and manage pet profiles.
- Capture images and input symptoms.
- Submit diagnosis requests.
- Review diagnosis results and suggested actions.
- Access health tips, reminders, and blogs.

15. 3. Veterinary Professionals (Doctors)

Role: Experts listed in the app that users can contact for emergency or expert guidance.

Responsibilities:

- Display verified contact info on the consultation screen.
- Optionally update availability or accept feedback (if extended in future versions).

16. 4. Administrator

Role: Manages backend data flow, user account verifications, and application maintenance.

Responsibilities:

- Deploy and manage backend services on AWS EC2.
- Monitor AI services for uptime and performance.
- Review push notification configurations and database health.
- Ensure all AI models are up-to-date and functioning correctly.
- Support end-users through feedback collection and bug resolution.

2.1.1 List of Use Cases (Updated for SmartPaw)

1. User Management:

- Register New Users
- Login / Logout
- Manage User Sessions

2. Pet Profile Management:

- Add New Pet
- View Pet Details
- Edit or Remove Pet

3. Disease Detection:

- Capture Image via Camera
- Input Textual Symptoms
- Send Data for Diagnosis
- View Combined Prediction Results

4. Consultation:

- View List of Veterinary Doctors
- Contact Doctor via Provided Info

5. Health & Reminder Management:

- View Daily Health Tips
- Set Food Time Reminders
- Receive Push Notifications

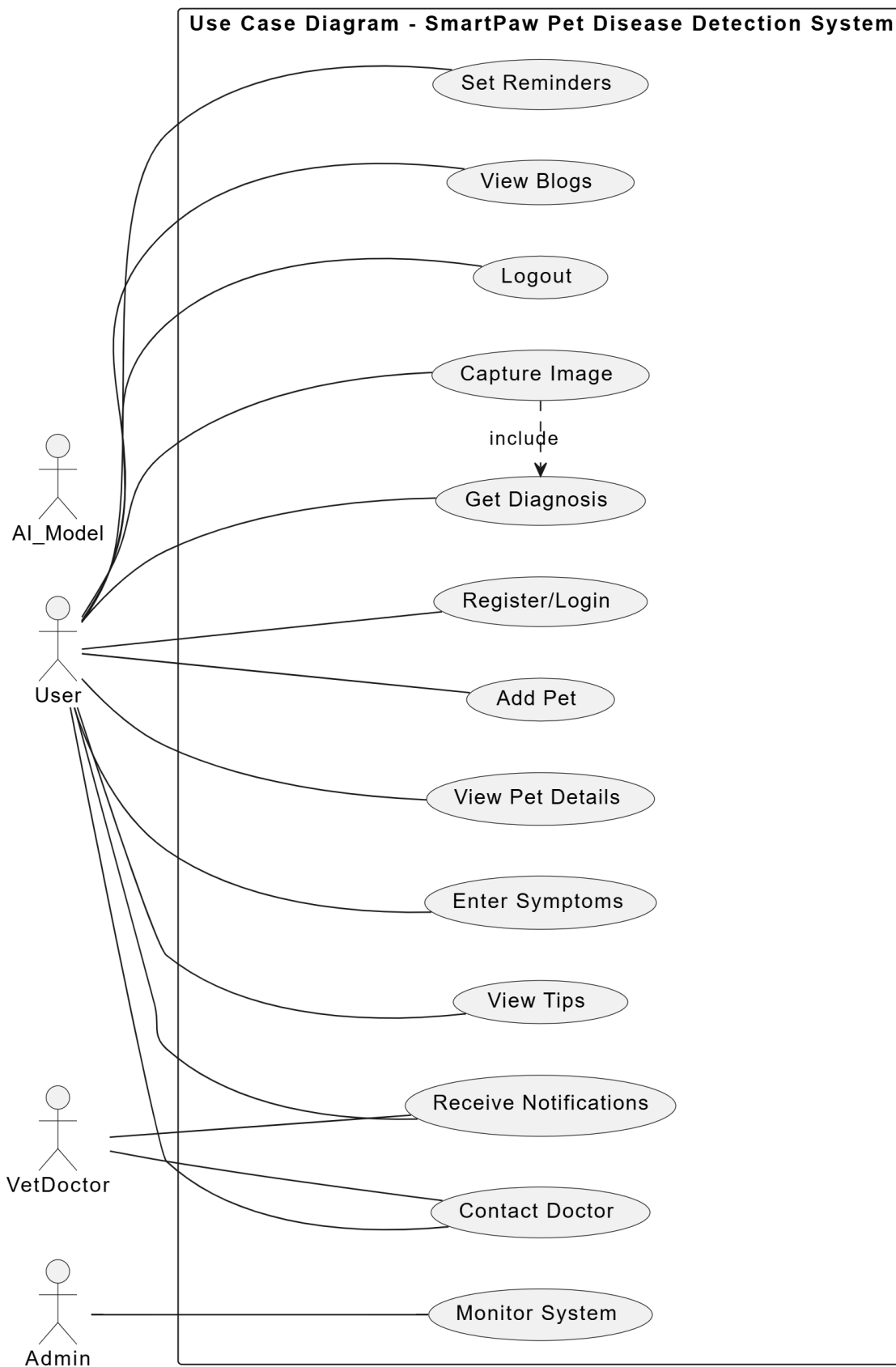
6. Community & Blogs:

- Access Blog List
- Read Individual Blog Posts

7. System Administration:

- Monitor Backend Performance
- Maintain Model APIs (Image + Text)
- Ensure Uptime of AWS-Hosted Services

16.1.1 Use Case Diagram



1.1.1 Description of Use Cases

Use Case Table – SmartPaw

Use Case ID	Use Case Name	Actors Involved	Description	Preconditions	Postconditions
UC01	Register / Login	User	Allows user to create an account or log in to access app features.	User has the app installed.	User is authenticated and taken to dashboard.
UC02	Add Pet	User	Enables users to add their pets' profiles with breed and type info.	User must be logged in.	Pet profile is saved and listed.
UC03	View Pet Details	User	Shows pet information in a bottom sheet format.	Pet must be added.	Details are shown to the user.
UC04	Capture Image of Symptoms	User, Camera	Takes an image of the pet's visible symptoms via device camera.	Camera permission is granted.	Image is sent to image model API.
UC05	Enter Symptom Description	User	Allows text input of pet's symptoms.	User is on disease detection screen.	Text is sent to text model API.
UC06	Get Diagnosis	AI Model (Image, Text)	AI processes image and text to return disease predictions.	Image and symptom data submitted.	Prediction is shown in alert dialog.
UC07	View Health Tips	User	Displays random or curated health tips for pets.	User is logged in.	Tips shown on the dashboard.
UC08	Set Food Reminders	User, Firebase	Allows setting of feeding times with notification alerts.	Notification permission granted.	Firebase sends scheduled alerts.
UC09	View Blogs	User	Displays blogs from community members.	User is logged in.	User can read blogs.
UC10	Contact Doctor	User, Vet Doctor	Allows user to view and contact listed veterinary doctors.	Doctor list is loaded.	Contact method (e.g., call) is initiated.
UC11	Receive Push Notifications	User, Firebase	Sends timely notifications about reminders or tips.	User has enabled notifications.	User receives alert via Firebase.
UC12	Logout	User	Logs the user out and ends the current session.	User is logged in.	User is logged out and returned to login.
UC13	Monitor System	Admin	Ensures backend/API uptime and model response monitoring.	Admin access is granted.	System logs and performance are reviewed.

2. Non-Functional Requirements

2.1 Performance Requirements

SmartPaw is hosted on an online server and may take some time to load initially depending on the user's device and internet speed. The system's responsiveness is influenced by the device hardware and internet bandwidth. The performance of disease detection is highly dependent on the quality of the images and accuracy of the text input provided by the user. The system may not be effective in rare or unseen pet health conditions, as predictions rely on previously trained data patterns.

2.2 Safety Requirements

All code for SmartPaw is securely stored in version control systems. The platform has restricted access—only authenticated users and the administrator have permission to operate sensitive features. In case of high server load or failure, SmartPaw will display an appropriate error message. If there are issues retrieving data or connecting to the model APIs, the app will notify the user. Regular maintenance cycles are required to keep the system stable, responsive, and secure.

2.3 Security Requirements

The app ensures user authentication through secure login credentials. Sensitive data related to user profiles and pet health is safeguarded against unauthorized access. Since the system transmits data for analysis, secure channels and validation protocols are in place to maintain data privacy. Push notifications through Firebase are strictly limited to reminders and health tips.

2.4 Reliability Requirements

SmartPaw aims for high uptime to ensure users can reliably access features like pet diagnosis and reminders.

- **Availability:** The application should maintain a 99.9% uptime.
- **Error Handling:** In case of input errors or model failure, users receive appropriate alerts without affecting the overall experience.
- **Data Consistency:** Pet profiles, health records, and model results must remain intact. The system will perform routine data backups and checks.

2.5 Usability Requirements

SmartPaw is built with user-friendliness in mind, particularly for non-technical pet owners.

- **Simple UI:** The interface is intuitive, offering easy navigation for adding pets, capturing symptoms, and viewing results.
- **Accessibility:** The app supports accessible design standards including clear fonts, high contrast, and keyboard compatibility.
- **Support Tools:** Users are provided with tooltips, FAQs, and short guides to ensure they can easily understand how to use each feature.

2.6 Supportability Requirements

SmartPaw is structured to allow future maintenance and scalability.

1. **Modular Design:** Components like disease detection, reminders, and user management are independently maintainable.
2. **Technical Documentation:** Developers will maintain thorough documentation covering backend services (NestJS), AI models (FastAPI), and Android integration.
3. **Monitoring:** The system includes backend logging and analytics to track errors, usage, and performance issues.

References

1. **Tan & Le (2019)** – EfficientNet for image-based deep learning.
2. **Sanh et al. (2019)** – DistilBERT: A lightweight NLP model.
3. **Paszke et al. (2019)** – PyTorch: A popular deep learning library.
4. **Abadi et al. (2016)** – TensorFlow: Scalable ML framework.
5. **Pedregosa et al. (2011)** – Scikit-learn: Traditional ML tools.
6. **FastAPI Docs** – High-performance Python web framework.
7. **NestJS Docs** – Node.js backend framework.
8. **Firebase Docs** – Cloud messaging for mobile notifications.
9. **Android Developers** – Guide for Java/XML Android development.
10. **Uvicorn Docs** – ASGI server for FastAPI deployment.
11. **Miotto et al. (2018)** – Review of deep learning in healthcare.
12. **Esteva et al. (2017)** – AI for skin disease detection.

13. **Sommerville (2015)** – Standard text for software engineering.
14. **Pressman (2014)** – Software engineering practices.
15. **Wiegers & Beatty (2013)** – Software requirements specification.
16. **IEEE Std 830-1998** – SRS documentation standard.
17. **OWASP** – Mobile app security best practices.
18. **ISO/IEC 27001** – Information security compliance.
19. **Git/GitHub Docs** – Version control systems.
20. **PyPI** – Official Python package repository (e.g. Pillow, tqdm, transformers).

A3. DESIGN SPECIFICATIONS

1. Introduction

2. Purpose of Document

This document outlines the technical design and implementation strategy for the development of **SmartPaw**, an AI-powered mobile application for detecting pet diseases using both image and text-based deep learning models. It details the architecture, design methodologies, and processes needed to ensure a scalable and reliable solution. The design follows an Object-Oriented approach to promote modularity, reusability, and maintainability across components like image classification, symptom analysis, and user interface design.

Intended Audience

This document is intended for:

- **Project Managers:** To track progress and ensure deliverables are met.
 - **Mobile Developers:** To follow the Android application architecture and component integration.
 - **Machine Learning Engineers:** To implement and refine image and text detection models.
 - **Quality Assurance Engineers:** To validate app functionality and stability.
 - **Stakeholders and Supervisors:** To ensure alignment with the app's vision and user needs.
-

Project Overview

Functional Requirements

- The app shall be mobile-based and accessible to pet owners.
- Users must register/login before accessing the core features.
- A dashboard with bottom navigation will allow users to switch between screens (Community, Pet Info, User Profile, and Consultation).
- A center floating action button enables users to take a picture of their pet's symptoms.
- Users can submit both image and text data for diagnosis.
- Backend prediction services are powered by deep learning models and hosted on AWS EC2.
- Firebase is integrated solely for push notifications (e.g., food reminders).
- Disease predictions will be presented in a dual-model result card.

Scope

Project Scope

- The application allows users to detect diseases using either photo input or text descriptions.
- Users can register their pets, view pet health tips, set feeding schedules, and consult veterinarians in emergencies.
- Admin privileges are handled server-side for backend management.
- The app provides a clean UI for capturing and submitting pet data.
- It supports the integration of both image (CNN) and text (Transformer) models for comprehensive detection.
- Data visualization includes displaying detected conditions and recommendations in alert dialogs.
- Firebase Cloud Messaging will handle health reminders.

4. Design Considerations

1. Scalability:

The backend architecture is hosted on AWS EC2 and can be scaled horizontally to accommodate high model query loads.

2. Performance:

Real-time predictions and fast UI responsiveness are ensured by optimizing image preprocessing and lightweight model loading.

3. Data Quality:

Proper preprocessing (resizing, tokenization) ensures clean input to AI models for more accurate outcomes.

4. Security:

Data is transmitted securely via HTTPS and stored with proper access control mechanisms in place.

5. Usability:

The mobile UI is intuitive, responsive, and beginner-friendly with clear icons, tooltips, and animations.

6. Integration:

The application integrates deep learning APIs via REST endpoints and uses Firebase SDK for push notifications.

7. Maintenance:

The modular design and documentation support easy updates of models or UI components.

8. Model Accuracy:

The image and text models are evaluated continuously using new symptom data and user feedback.

9. Deployment:

CI/CD pipelines and Docker containers are used to deploy and maintain backend services on AWS.

Assumptions and Dependencies

Assumptions:

- Users will have internet access to send/receive prediction data.
- Pet disease image data is assumed to be clear and relevant.
- User descriptions will be in understandable English for text processing.
- Device cameras will be functional for symptom capture.

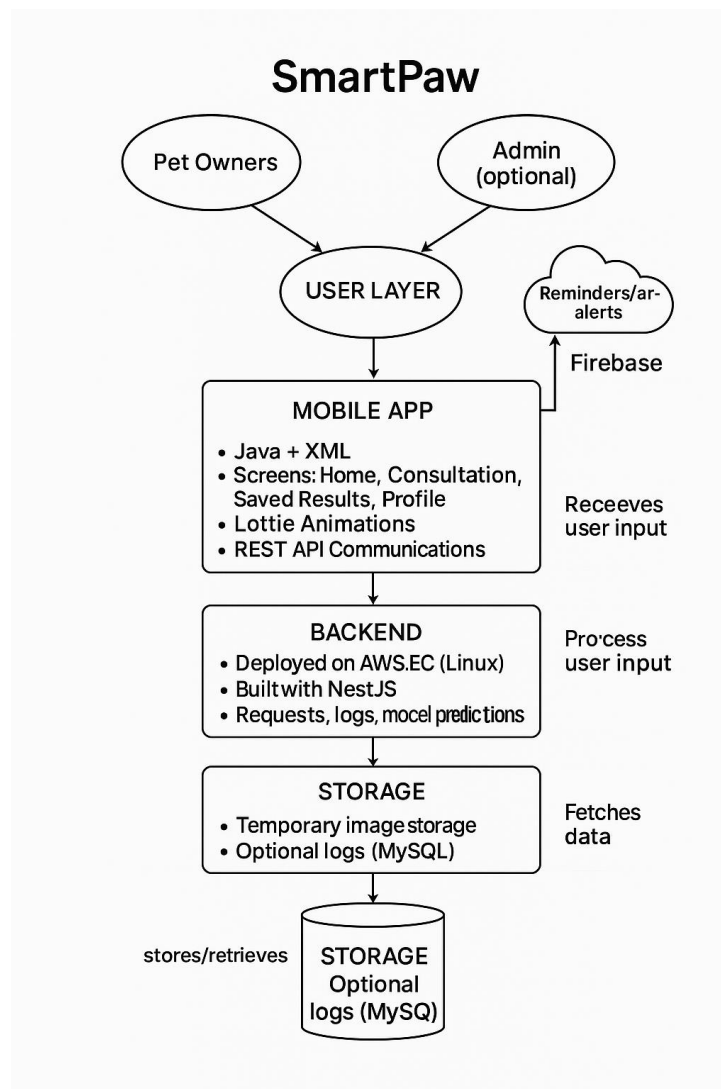
Dependencies:

- Image model based on CNN (EfficientNet) and text model based on Transformer (DistilBERT).
- Backend hosted on AWS EC2 (Linux instance) using FastAPI.
- Firebase for notification and reminder services.
- Android devices for app operation (Java + XML based).

Risks and Volatile Areas

- **Image Quality:** Blurry or unrelated images may reduce detection accuracy.
- **Text Misinterpretation:** Incorrect or vague symptom descriptions can affect predictions.
- **Model Drift:** Over time, newer diseases may not be accurately detected unless the models are retrained.
- **Firebase Downtime:** If Firebase service is down, users may miss scheduled reminders.
- **Security Risks:** Ensure secure API endpoints to prevent misuse of AI models.
- **Device Limitations:** Older Android phones might experience slower performance.

System Architecture



System Level Architecture

1. **1. System Decomposition:**
2. **• User Interface (UI):**

SmartPaw's front-end is a mobile application developed for Android using Java and XML. It features user-friendly screens such as Home, Consultation, Saved Results, and Profile, enriched with Lottie animations. It facilitates user interaction, allowing pet owners to submit symptoms or images for diagnosis and receive results. It also supports real-time notifications through Firebase.

3. • **Backend:**

The backend is built with NestJS and hosted on AWS EC2 (Linux). It handles all client-server communication via RESTful APIs. Core responsibilities include processing incoming data, managing model predictions, logging activities, and serving as the central link between the UI, AI models, and data storage.

4. • **AI Model Layer:**

SmartPaw employs two AI models: EfficientNet for image-based disease detection and DistilBERT for processing symptom descriptions in text. These models are triggered by the backend to analyze user inputs and return disease predictions.

5. • **Storage:**

Temporary storage of user-submitted images and optional logging of system activity is handled via a MySQL database. This module ensures efficient data access and aids in system analysis or debugging.

6. • **Notification Service:**

Firebase Cloud Messaging is integrated to deliver alerts and health-related reminders to users. This helps maintain user engagement and ensures timely actions for pet health.

7. **2. Relationships between Elements:**

8. • The **Mobile App** sends user inputs (text or images) via REST APIs to the **Backend**, which processes the data and communicates with the AI models. Results are returned to the app for user display.
9. • The **Backend** calls the **EfficientNet** or **DistilBERT** models depending on input type, retrieves predictions, and may log results to the **Storage** module.
10. • **Firebase** interacts directly with the mobile app to send push notifications based on system events or schedules.
11. • All input/output and model interactions are managed through the backend to maintain control and consistency.

12. 3. Interfaces to External Systems:

13. • Data Import:

The system may support image or text uploads from external apps or local device storage for disease detection.

14. • API Integrations:

RESTful endpoints allow future integration with veterinary services, pet health monitoring tools, or third-party analytics platforms.

15. _____

16. 4. Major Physical Design Issues:

17. • Execution Platform:

All backend operations and AI inference run on cloud infrastructure (AWS EC2). This allows scalability, uptime reliability, and centralized control.

18. • Global Design Strategies:

SmartPaw incorporates secure data handling (e.g., HTTPS and access control), structured logging for traceability, and optimized resource usage for model execution and storage. Error handling is built into the backend to manage faults and provide user feedback effectively.

Software Architecture

1. User Interface Layer:

- Provides the mobile front end developed using **Android (Java + XML)**.
- Includes key screens: **Home, AI Disease Detection, Consultation, Pet Screen, and Profile**.
- Utilizes **Lottie animations** for enhanced user experience.
- Sends requests and receives results via **REST APIs**.
- Supports **Firebase push notifications** for alerts and reminders.

2. Middle Tier:

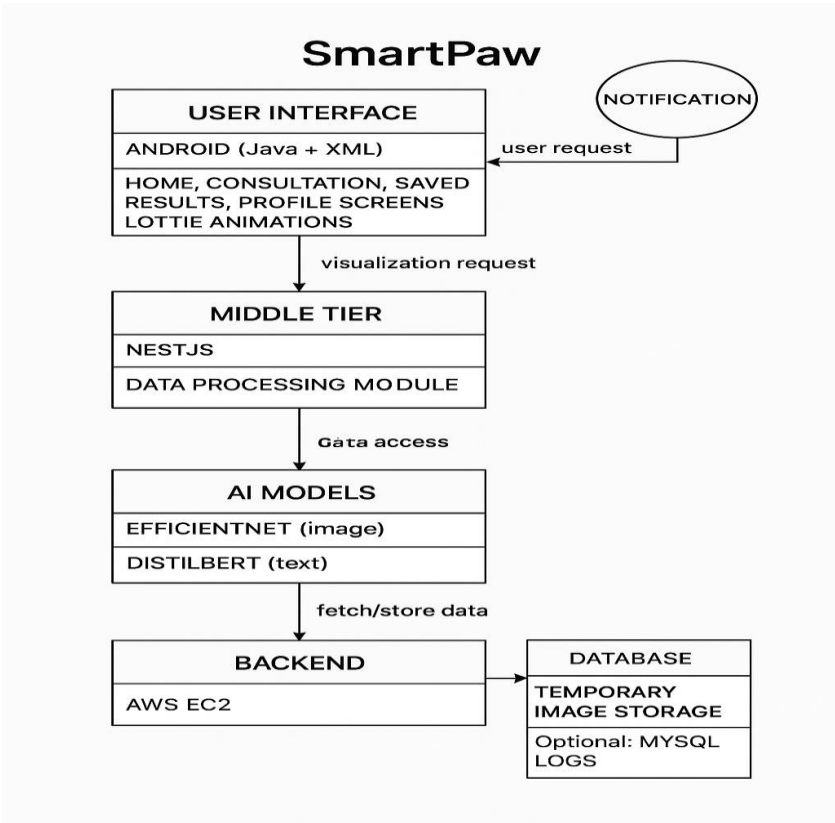
- Built with **NestJS**, this layer contains core **business logic**.
- Manages communication between the mobile app and backend components.
- Handles input validation, session management, and API routing.
- Includes a **Data Processing Module** to prepare data for AI model inference.

3. Data Access Layer:

- Manages secure and optimized data retrieval and storage operations.
- Communicates with storage systems using structured queries.
- Acts as a mediator between the backend logic and database infrastructure.

4. Database Layer:

- Stores all persistent and temporary data used by the system.
- Includes:
- **MySQL database** for logs and user-related data
- Ensures data integrity and supports fast retrieval for inference and user history.



6. Design Strategy

The design strategy for the **SmartPaw** pet disease detection system emphasizes performance, scalability, modularity, and user-friendly experience. The system is architected to deliver real-time AI-powered predictions while remaining flexible enough for future enhancements.

1. Future System Extension or Enhancement

- **Modular Design:**
SmartPaw follows a modular architecture, enabling independent upgrades or feature additions—such as integrating new AI models or expanding with vet teleconsultation—without disrupting the current flow.
- **API-Centric Approach:**
The backend, developed in **NestJS**, exposes RESTful APIs to ensure seamless integration between the Android app, AI models, and any future third-party services (e.g., vet booking, health record sync).

2. System Reuse

- **Reusable Components:**
Core modules like image preprocessing, text classification, and result formatting are designed for reuse across various features of the application, ensuring consistency and reducing development overhead.
- **Library Usage:**
The system utilizes well-established libraries such as **TensorFlow (EfficientNet)**, **Transformers (DistilBERT)**, and **Express/NestJS** for backend logic, all structured to promote clean, reusable code.

3. User Interface Paradigms

- **Responsive Design:**
The Android app is built with adaptable layouts to support a range of device sizes, offering smooth usability across phones and tablets.
- **Intuitive Interaction:**
The interface includes animated Lottie-based interactions, clear navigation, and clean visualizations, such as pie charts for predictions and chip-based selectors to simplify user input.

4. Data Management (Storage, Distribution, Persistence)

- **Temporary Storage:**
Images and consultation logs are temporarily stored (e.g., in MySQL) for prediction processing and optional future access.
- **Data Caching:**
Frequently accessed data like previous results or model configurations are cached for improved responsiveness.
- **Scalability:**
Hosted on **AWS EC2 Linux**, the backend is designed to scale with increasing user load and AI

- processing demands. The architecture allows containerization and deployment via CI/CD pipelines when needed.

5. Concurrency and Synchronization

- **Asynchronous Processing:**
AI prediction requests are handled asynchronously. Image uploads and model inference occur in the background to prevent delays in the UI and improve app responsiveness.
- **Thread Safety:**
Backend endpoints handling parallel user requests (e.g., predictions, data retrieval) are protected with thread-safe mechanisms to avoid race conditions.
- **Locking Mechanisms:**
Critical operations involving shared resources—like temporary file access or concurrent prediction queues—use locking and queueing mechanisms to ensure integrity and consistency.

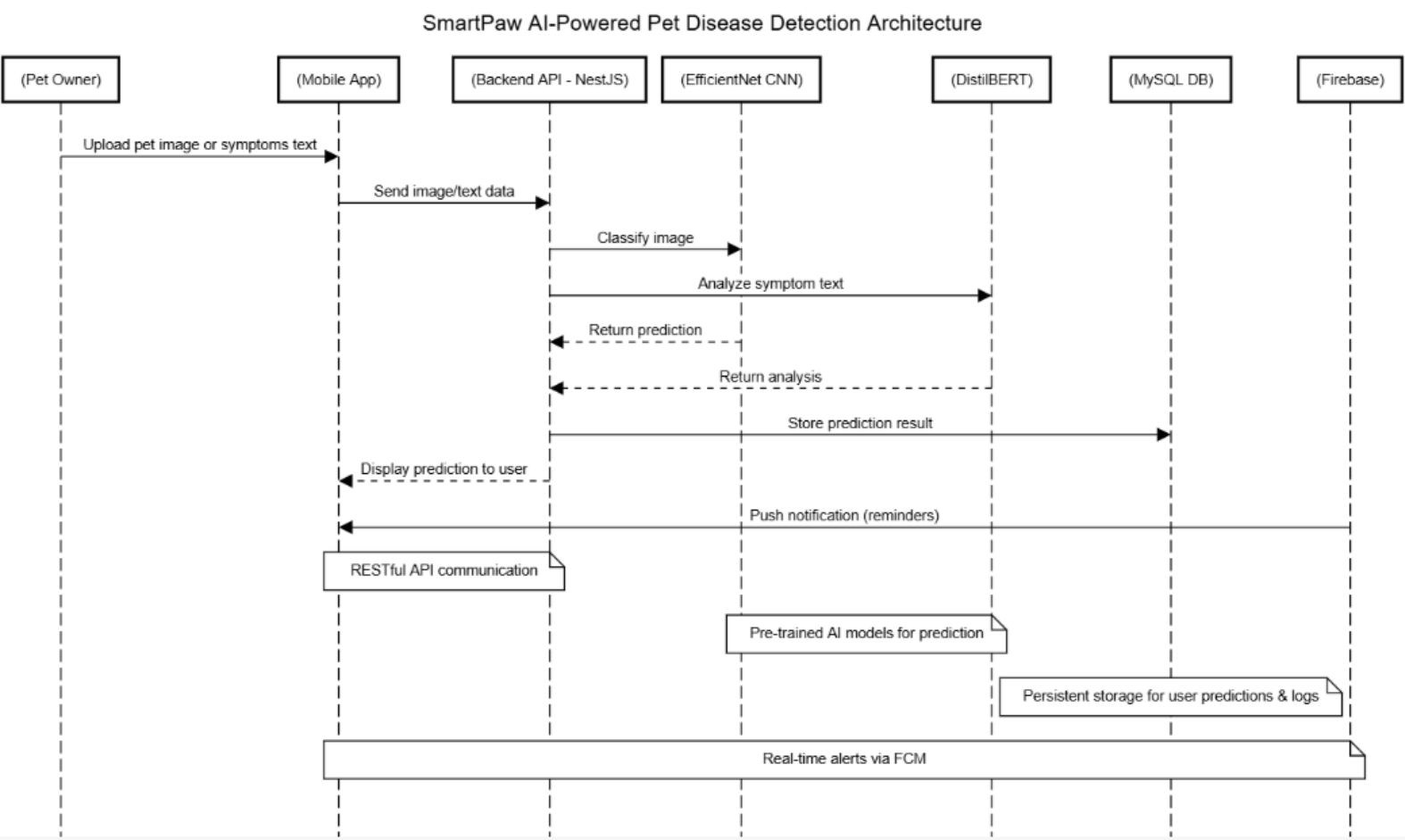
5. Detailed System Design

Database Design

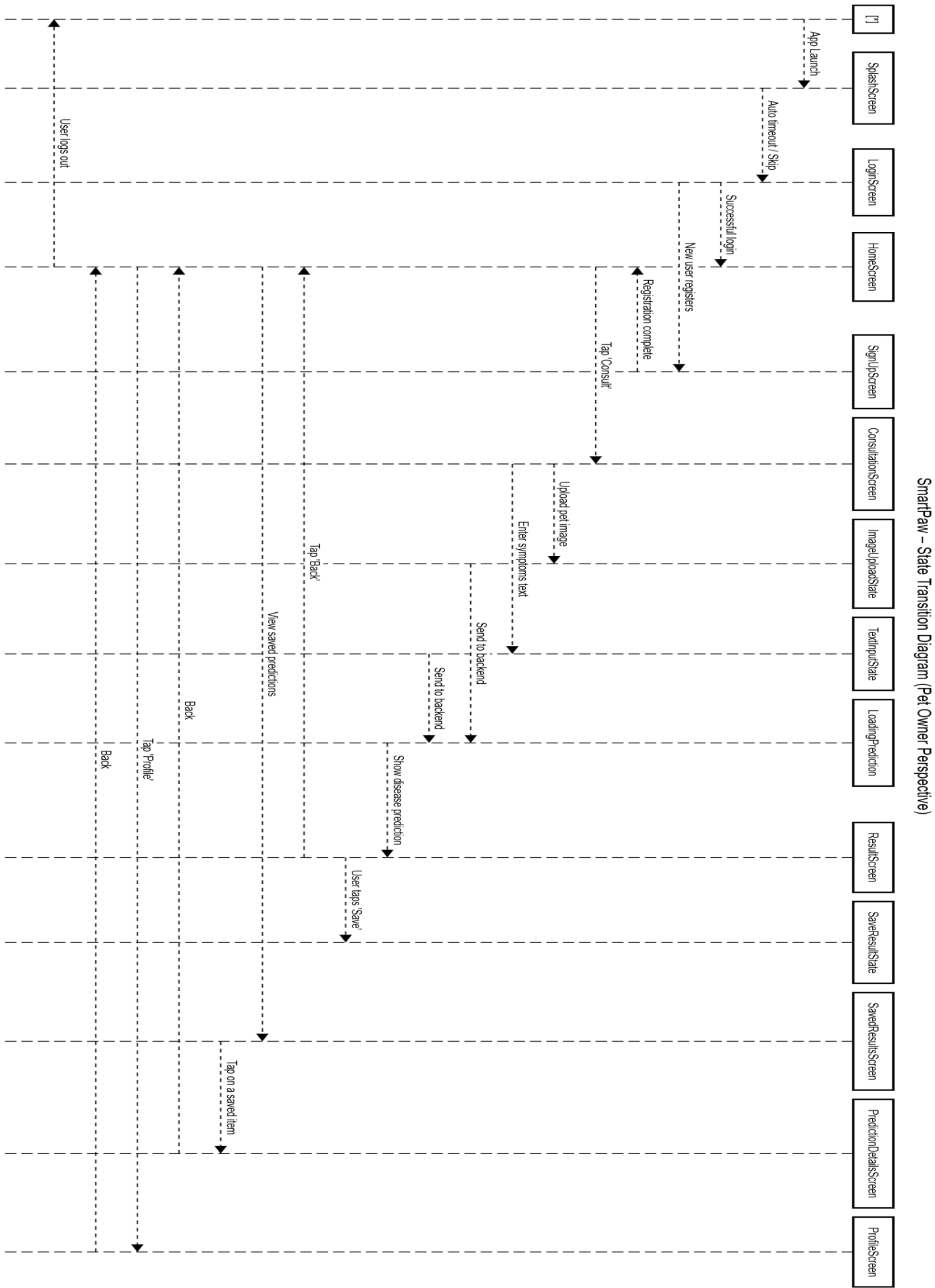
Data Dictionary

- **Data Collection:**
Structured data is gathered through user input (images, symptoms, descriptions) via the mobile application. Additional reference datasets (e.g., known pet diseases) may be sourced through publicly available APIs or licensed datasets.
- **Data Storage:**
The collected data—including consultation history, image metadata, and prediction results—is stored in a **relational database (MySQL)** on the backend server. Storage is optimized for retrieval speed and consistency.
- **Data Preprocessing:**
Input data is cleaned before being passed to AI models. This includes resizing images, removing noise, normalizing pixel values, and processing text input (tokenization, stop-word removal) for accurate classification.
- **Data Integration:**
The system supports integration with pre-labeled datasets (used during model training) and may also merge real-time input with previously stored consultation records to improve model relevance and prediction accuracy.
- **Data Analysis:**
AI models (EfficientNet for image, DistilBERT for text) analyze incoming data to detect patterns and predict potential diseases. The backend also logs key performance data for future tuning, analytics, or model retraining.

Application Design: Sequence Diagram



State Transition Diagram



Sequence Diagram

Sequence Diagram 1: User Login Sign up flow

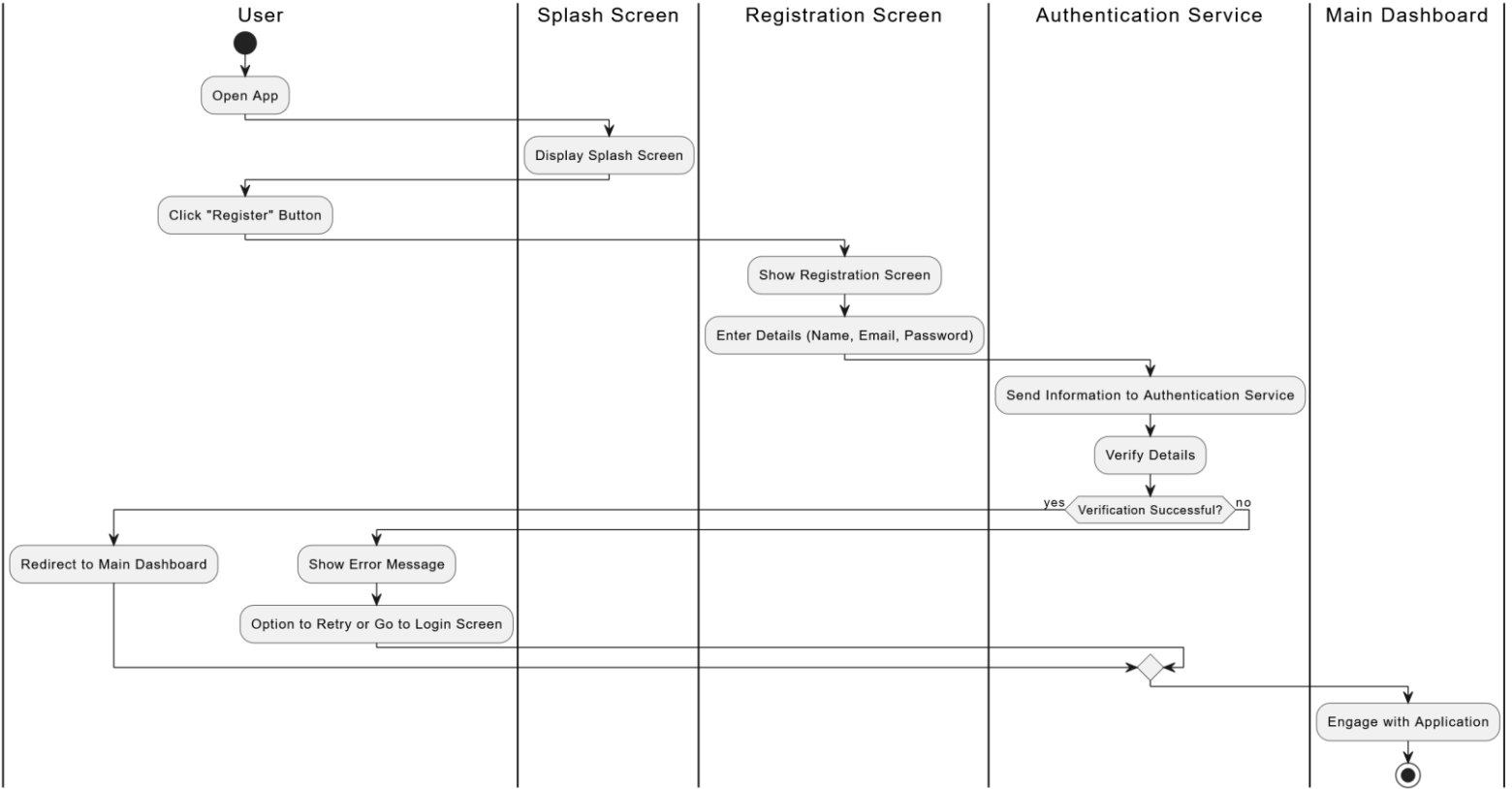


Figure 13 Sequence Diagram 1

Process:

• User Signup/Login:

The pet owner opens the SmartPaw mobile app and either signs up for a new account or logs in with existing credentials.

• Mobile App Interface:

The app collects the user's email and password via input fields and sends the data to the backend through a standard HTTP request (e.g., POST for login/signup).

- **Backend Server (NestJS on AWS EC2):**

Receives the request and performs the following:

- For **signup**: Validates input, checks if the user already exists, and if not, stores the new user record in the database.
- For **login**: Validates input and compares submitted credentials against stored values.

- **Database (MySQL):**

Stores and retrieves user records securely. Passwords are stored in hashed format (e.g., bcrypt).

- **Response Sent to App:**

If credentials are valid or signup is successful, the backend responds with a status (e.g., "Login Success", "Signup Successful").

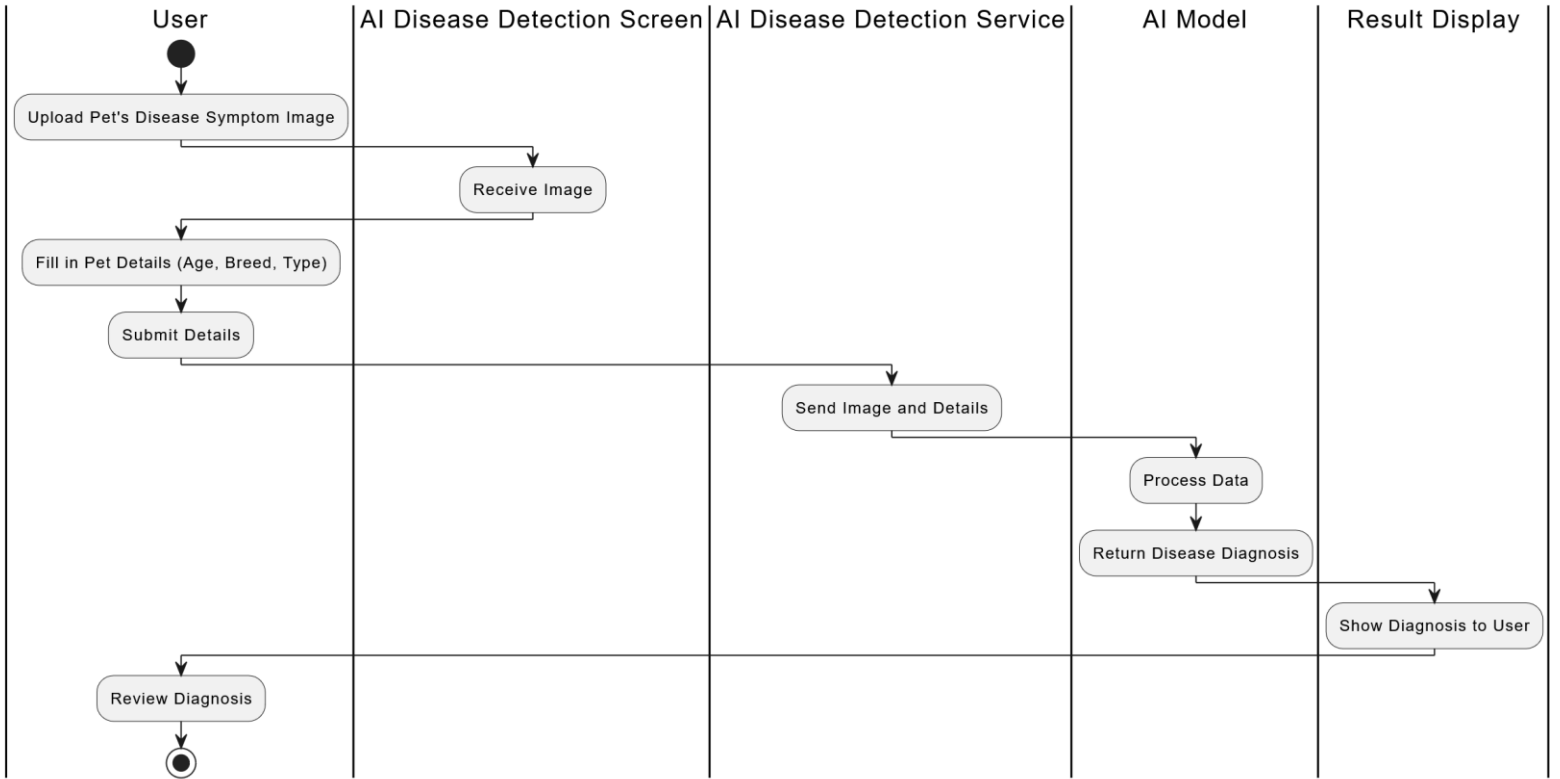
- **Access to App Granted:**

Upon receiving a success response, the app navigates the user to the home screen. No token or session mechanism is used — access is managed through stateful navigation in the app.

Theory:

This login/signup flow provides basic access control for SmartPaw without implementing session tokens or OAuth. While it simplifies implementation, password encryption and input validation are still essential to ensure user data safety.

Sequence Diagram 2: AI Disease Detection Model Output



Purpose:

This diagram illustrates how SmartPaw handles the flow when a user captures a photo, provides pet details, and submits the data for prediction through two separate AI models — one for image analysis and the other for symptom classification.

Process:

- User Captures & Submits Data:**
The pet owner taps the camera button in the SmartPaw mobile app, captures the image of the pet, enters additional details such as pet name, category (dog/cat), breed, and visible symptoms, then submits the form.
- Mobile App Interface:**
The data (image + text) is packaged and sent to the backend server via a RESTful API.
- Backend (NestJS on AWS EC2):**
Receives the request and orchestrates processing:

- Sends the image data to the **EfficientNet CNN model**.
- Sends the symptom text to the **DistilBERT Transformer model**.

- **AI Models (Image + Text):**

- **EfficientNet CNN** processes the pet image to identify potential visual symptoms (e.g., rashes, eye issues).
- **DistilBERT Transformer** analyzes the textual symptoms to match against disease patterns in the language model.

- **Prediction Aggregation:**

The backend receives predictions from both models and combines them into a unified disease diagnosis result, which may include a confidence score, suggested diseases, or a recommendation to consult a vet.

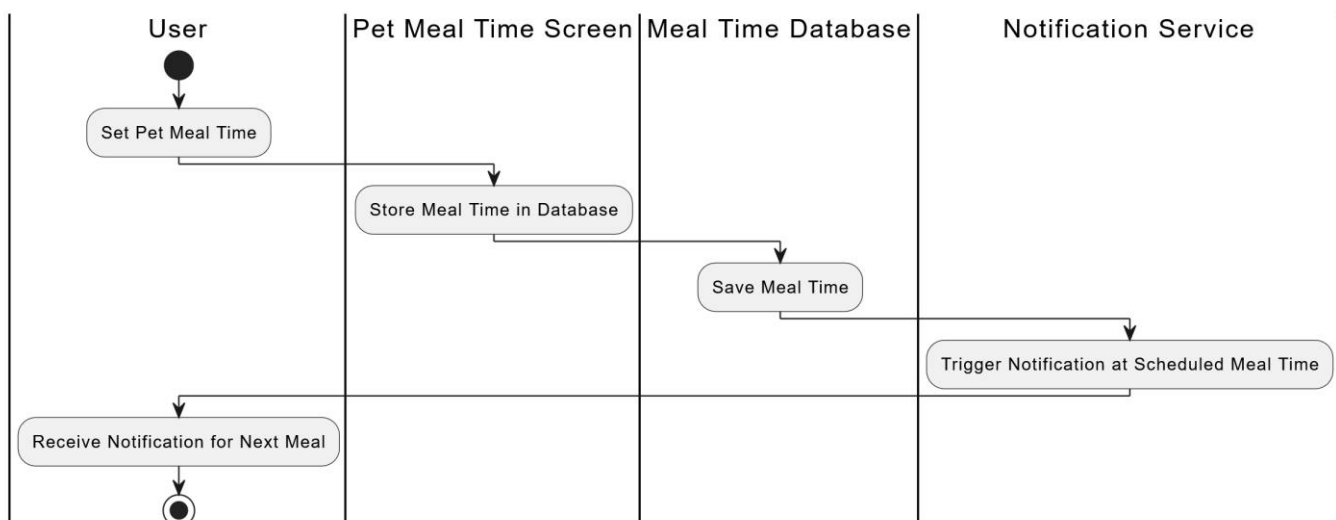
- **User Notified:**

The SmartPaw app receives the final prediction and presents it on the result screen with supportive visuals or advice. The case can optionally be saved to history.

Theory:

This process ensures SmartPaw provides intelligent, AI-powered disease detection using multimodal data (image + text). EfficientNet and DistilBERT complement each other — one analyzing visible symptoms, the other interpreting the user's input — delivering a more accurate and holistic diagnosis.

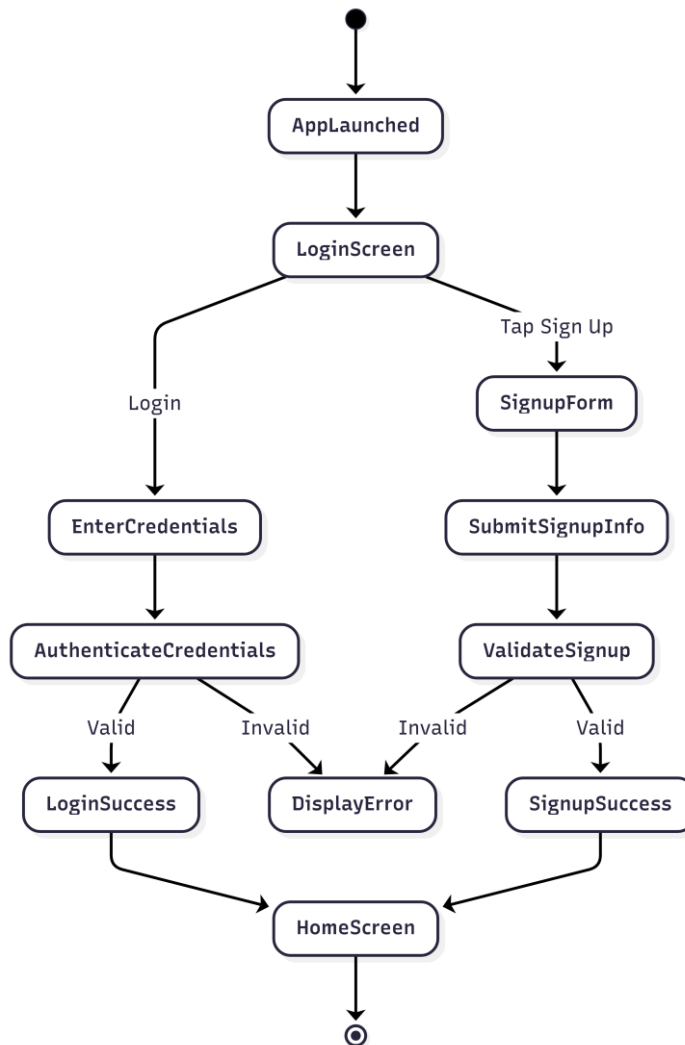
Sequence Diagram 3: Pet Meal Time reminder



State Diagram

State Diagram 1: Data Upload and Preparation

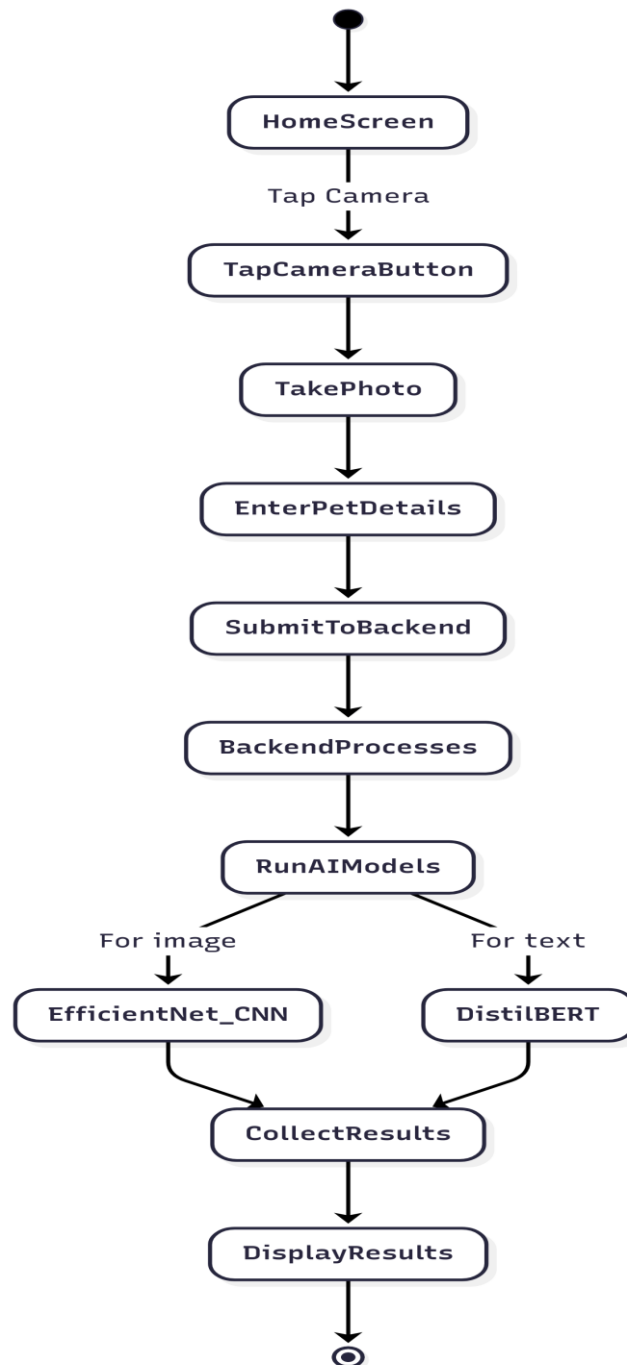
Figure 16 State Diagram 1



Key Points:

- Users land on the login screen after launching the app.
- Option to sign up is available for new users.
- Login requires valid email and password for authentication.
- Signup collects pet owner details and creates account after validation.
- Successful login or signup leads the user to the main dashboard.

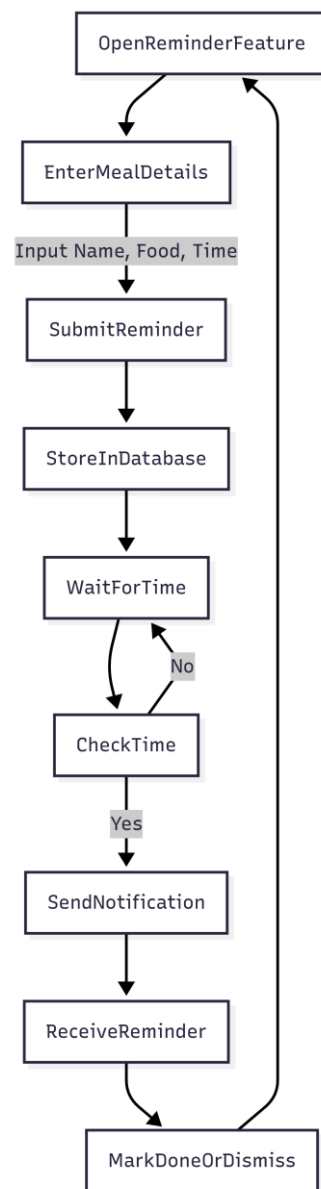
State Diagram 2: Model Training



Key Points:

- User initiates detection by taking a photo through the app.
- Pet details are entered to enhance prediction accuracy.
- Data is sent to backend where two AI models process it.
- Image goes to EfficientNet CNN, text to DistilBERT Transformer.
- The diagnosis results are shown clearly with possible conditions.

State Diagram 3: Forecast Generation and Visualization



Key Points:

- Pet owners can schedule meal reminders using time and food inputs.
- Backend stores these reminders and monitors time continuously.
- Firebase sends a push notification when mealtime is near.
- The user is alerted even if the app is closed.
- Pet owners can confirm the reminder or dismiss it after feeding.

a. UI/UX Detail Document

UI/UX Details:

1. User Inputs:

- **Pet Category Selection:** A dropdown where users select the pet type (e.g., Dog, Cat).
- **Breed Selection:** A breed selector based on the chosen pet category.
- **Symptom Description:** A text input where users describe the visible symptoms.
- **Image Upload:** Option to capture or upload an image showing the pet's condition.

2. Action Buttons:

- **Submit:** Sends the pet image and text data to two different AI models (image and text-based) for disease detection.
- **Generate Report:** Produces a combined summary from both AI model outputs with possible disease predictions and health suggestions.

3. Graphical Visualization:

- **AI Results Dialog:** An alert dialog presents both image and text-based prediction results with visual indicators like icons, confidence levels, or cards for diseases.
- **Pet Stats Charts:** Optional area or line charts showing health data trends or activity logs (e.g., meal times, health tip reads).

4. User Interface Design:

- **Intuitive Layout:** Clean and user-friendly design with bottom navigation for quick access to main features.
- **Responsive:** Fully compatible with various Android devices, maintaining consistency across screen sizes.

Daily Symptom Detection Flow:**1. Input Fields:**

- **Pet Type:** Dropdown to select Dog, Cat, or other types.
- **Breed:** Automatically populated based on pet type.
- **Symptom Description:** Text input describing the pet's health condition.
- **Image Capture:** Launches the camera to capture a symptom photo.

2. Submit Button:

- **Submit:** Sends text and image data to respective FastAPI endpoints for processing.

3. Generate Report Button:

- **Generate Report:** Shows detailed feedback based on both AI models including disease name, urgency level, and next steps.

4. Prediction Dialog:

- **Dialog Display:** Displays prediction results as cards with disease name, description, and confidence percentage.

Weekly Health Monitoring:

This section allows weekly pet care engagement:

- **Inputs:** Users may log weekly symptoms or updates through text or images.
- **Actions:** Clicking "Submit" will log the entry and optionally run detection.
- **Visualization:** Shows historical data of pet health symptoms, meal logs, or activity levels over a week.

Monthly Health Insights:

This section shows monthly summaries for the pet's health:

- **Inputs:** No inputs required unless users log new entries.
- **Actions:** Auto-generates monthly health insights.
- **Graph:** Charts showing symptom frequency, reported issues, and health tip views.

Yearly Pet Health Overview:

1. User Inputs:

- **Pet Selector:** Choose one of the added pets.
- **Time Period:** From Date to To Date to analyze long-term patterns.
- **Chart Type:** Choose how to view the insights — pie or line chart.

2. Action Buttons:

- **Submit:** Runs analysis on past entries and AI prediction history.
- **Generate Report:** Summarizes overall pet health trends.
- **Show in Pie Chart:** Displays a visual breakdown of symptoms or disease categories.

Functionality:

- **Data Entry:** Users provide symptom description, image, and pet type.
- **AI Detection:** On "Submit," data is processed by both CNN (image) and Transformer (text) models via FastAPI.
- **Reporting:** "Generate Report" presents combined AI results in a readable format for users.

b. CODING STANDARDS

1. Code Formatting:

- **Indentation:** Use 4 spaces consistently.
- **Braces:** Use same-line braces for conditionals and methods.
- **Spacing:** Maintain clean spacing around operators, commas, and between parameters.

2. Naming Conventions:

- **Variables:** Use camelCase (e.g., `petSymptomInput`, `userProfileData`).
- **Functions:** Use camelCase (e.g., `submitSymptoms()`, `getPredictionResult()`).
- **Classes:** Use PascalCase (e.g., `PetAnalyzer`, `SymptomUploader`).
- **Constants:** Use UPPER_CASE_WITH_UNDERSCORES (e.g., `DEFAULT_IMAGE_PATH`).
- **Files:** Use lowercase with underscores or hyphens (e.g., `image_handler.java`, `ai_response_adapter.java`).

3. Comments:

- **Inline Comments:** Only for complex logic (e.g., `// Combine image and text AI results`).
- **Block Comments:** For full function or section descriptions (`/* Uploads symptom image to FastAPI */`).
- **Docstrings:** Used in Python FastAPI endpoints to describe the request and response format.

4. Error Handling:

- **Mechanisms:** Use try-catch in Android and try-except in Python.
- **Messages:** Display user-friendly errors like “Unable to connect to server.”
- **Logging:** Log errors in backend without exposing user data.

5. File Organization:

- **Structure:** Use folders like `activities/`, `fragments/`, `services/`, `adapters/`, and `models/`.
- **Naming:** Descriptive names like `DiseaseResultAdapter.java`, `ConsultationService.kt`.

6. Version Control:

- **System:** Use Git for source control.
- **Commits:** Follow the format `feat:`, `fix:`, or `refactor:` for commit messages.
- **Branching:** Use feature-based branches like `feature/image-upload`.

7. Performance:

- **Efficiency:** Reuse bitmap images, limit heavy background tasks.
- **Profiling:** Use Android Profiler and Python performance tools to track memory and speed.

8. Security:

- **Best Practices:** Don't store API keys in code; use secure storage.
- **Validation:** Always validate form fields before sending to the backend.

9. Review and Testing:

- **Code Review:** Peer reviews on GitHub or GitLab before merging.
- **Testing:** Unit test FastAPI routes, Android ViewModels, and AI response formatting.

b. USER DOCUMENT

1. Accessing the Application

- Launch the SmartPaw app from your Android device.
- On startup, you'll see the splash screen followed by introductory welcome screens.
- Proceed to the login screen where you can either log in with your existing credentials or sign up as a new user.

2. Exploring the Dashboard

- After logging in, you will be directed to the main dashboard.
- The dashboard features a bottom navigation bar that allows access to different sections: Community, Pets, Profile, and the central Camera button.
- Each section helps you manage your pet's health, explore blog content, or consult with a vet.

3. Using the Symptom Detection Feature

- Tap the center **Camera** button on the bottom navigation bar.
- Capture a clear image of your pet's visible symptoms.
- Select the pet type (e.g., Dog, Cat) and corresponding breed from the dropdowns.

- Describe the symptoms in the text field provided.
- Tap **Submit** to send this data to the AI system. The app will process the image and text inputs using separate models and return a disease prediction.

4. Viewing AI Predictions

- After submitting the image and symptom details, an alert dialog will appear showing:
 - Predicted disease(s) based on image input.
 - Predicted disease(s) based on text input.
 - Confidence level or probability of each result.
- The results may include suggestions for care or urgency indicators (e.g., "Visit Vet Immediately").

5. Using Meal Reminders and Health Tips

- From the main dashboard, scroll to find the **Meal Time Reminder** section.
- Enter your pet's food type and meal time to schedule reminders.
- The app will send timely notifications using Firebase.
- Health tips are available on the home screen to provide daily wellness advice for pet care.

6. Consulting a Veterinarian

- Tap on the **Consultation** section from the bottom navigation.
- View a list of available veterinarians with contact options.
- In case of emergency, you can directly initiate a call or message from this section.

7. Viewing Health Trends in Charts

- Go to the Pet Profile section to view health summaries and previous reports.
- Tap on chart icons to visualize trends such as disease predictions over time or meal history.
- You may toggle between line and pie chart views for better comparison.

8. Saving and Exporting Reports

- After receiving AI predictions, you have the option to **Generate Report**.
- Tap the **Export** button to download the report in formats such as PDF or image snapshot.
- Reports can be shared with a vet or saved for future tracking of your pet's condition.

b. Project Policy Document

1. Introduction

This document outlines the policy for developing the *SmartPaw* application—an AI-driven mobile solution designed to detect pet diseases using deep learning. The project leverages advanced image and text analysis models to provide timely, reliable disease predictions. It empowers pet owners with accessible health insights, direct vet consultation options, and proactive care recommendations, all within a user-friendly mobile platform.

2. Objectives

- **Develop a Mobile Application:** Build an Android-based app for real-time pet health diagnostics.
- **Integrate Dual AI Models:** Employ CNN (image-based) and Transformer (text-based) models for accurate symptom analysis.
- **Real-Time Prediction:** Enable live image capture and symptom input for instant disease detection.
- **Interactive Feedback:** Provide visual and textual prediction results through an intuitive alert dialog.
- **Pet Data Management:** Allow users to add, manage, and review pet profiles, health logs, and reports.
- **User-Friendly Interface:** Ensure a seamless, accessible design across Android devices.

3. Scope

- **Target Audience:** The app is designed for pet owners and caretakers across diverse regions.
- **Input Sources:** Combines image input from camera and text descriptions from user forms.
- **User Roles:** Supports general users (pet owners), emergency responders, and veterinarians.
- **Prediction Features:** Offers short-term health predictions, disease descriptions, and urgency indicators.
- **Security:** Implements Firebase Auth, input validation, and secure data handling for pet and user privacy.

4. Implementation Plan

- **Phase 1: Research and Analysis**

- o Study existing veterinary diagnostic systems and AI health apps.
- o Define functional requirements for mobile disease detection.
- o Identify image datasets, breed classifications, and symptom tags.

- **Phase 2: Design and Development**

- o Architect the app's structure including splash screen, onboarding, dashboard, and API communication.
- o Integrate CNN and Transformer models using FastAPI and Python.
- o Implement features like camera input, Firebase notifications, and UI components.

- **Phase 3: Testing and Evaluation**

- o Test AI models for prediction accuracy on various breeds and symptoms.
- o Conduct usability testing with real users.
- o Evaluate backend API performance and model response times.

- **Phase 4: Deployment and Maintenance**

- o Deploy the backend using NestJS on AWS and release the Android app via Play Store.
- o Monitor user feedback and continuously improve app functionality and accuracy.

5. Project Team and Responsibilities

- **Project Manager:** Manages overall progress, team coordination, and stakeholder communication.
- **AI Engineers:** Train, evaluate, and fine-tune CNN and Transformer models.
- **Android Developers:** Build the frontend, integrate FastAPI endpoints, and manage Firebase services.
- **Backend Developers:** Develop RESTful APIs with NestJS and manage AWS deployment.
- **UI/UX Designers:** Design the app flow and ensure intuitive pet owner interactions.
- **Quality Assurance Team:** Perform app testing, bug tracking, and performance checks.
- **Support Team:** Handle post-launch queries and assist users with troubleshooting.

6. **Timeline**

- The project is targeted for completion within [insert timeframe, e.g., 6 months].
- Key milestones include prototype release, AI model deployment, beta testing, and final launch.
- Status updates and progress reports will be shared after each development phase.

7. **Budget**

- The project budget will cover AI development, app design, backend deployment, and maintenance.
- Funding will be managed internally, with optional partnerships or grants from animal welfare organizations.

8. **Risk Management**


- **AI Accuracy Risks:** Address misdiagnoses through continuous training and real-world testing.
- **User Input Risks:** Validate user-provided images and symptom descriptions to ensure reliable results.
- **Technical Delays:** Have contingency plans for potential delays in model integration or API downtime.

9. **Stakeholder Communication**

- Keep all project stakeholders (developers, testers, investors, and pet health advisors) informed via weekly reports, meetings, and shared documentation.
- Communicate progress, feature updates, and release plans transparently to all parties involved.

A5. FLYER & POSTER DESIGN

F21



PROJECT NAME
Pet Health Care Industry

DESCRIPTION
This project develops an AI-powered pet care app to monitor health, track symptoms, and provide personalized care schedules, fostering convenience and community for pet owners.

AI

PROJECT STATUS
Second Evolution

SUPERVISOR
Sir Razaque

TEAM MEMBERS
Haris Bin Mohsin
M Asaad Ibrahim
Subhan

A6. COPY OF EVALUATION COMMENTS

COPY OF EVALUATION COMMENTS BY SUPERVISOR FOR PROJECT – I MID SEMESTER EVALUATION

Hamdard University
Faculty of Engineering Sciences and Technology
Department of Computing

FINAL YEAR PROJECT - PROPOSAL EVALUATION

File: Pat health care Industry
ID: FYP-041/FL24 Project Track: AI
Domain: AI Evaluation Date: 22/10/24
Supervisor Name: Eng. Abdul Razaque Co-Supervisor Name: Umar

Member(s):

Name	CMS ID
Subhan	571-2019
M. Asaad Ibrahim	2602-2021
Harris bin Mohsin	156-2020

Evaluators only:

Evaluation Parameters	Please select the appropriate option E: Excellent G: Good S: Just Satisfactory N: Not Satisfactory			
	Evaluator #1	Evaluator #2	Evaluator #3	Evaluator #4
Subject Knowledge	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N
Problem Statement	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N
Organization & Content of Presentation	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N
Project Scope Defined	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N
Methodology	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N
Language & Grammar	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N
Attire, Delivery and Presentation Skills	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N
Work Division	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N	<input checked="" type="checkbox"/> E <input type="checkbox"/> G <input type="checkbox"/> S <input type="checkbox"/> N
Name & Sign of Evaluator:	Osama Ahmed Khan			

Suggestions of evaluators:

Literature Review still not conducted. AI/feature ML has been included in the scope now.

For FYP Committee only:

Result Summary

On basis of evaluations, recommended action decided in FYP committee meeting:

☐ Approved ☐ Approved (with Revision) ☐ Re-Evaluate

Date: _____ Name and Sign of Convener FYP Committee: _____

**A7. COPY OF EVALUATION COMMENTS BY SUPERVISOR
FOR PROJECT – II MID SEMESTER EVALUATION**

A8. MEETINGS' MINUTES & Sign-Off Sheet

FYP Project Meeting

Minutes of Meeting

Meeting Date: 27/02/2024

Meeting Location: Hamdard University Meeting

Time: 10:00 am

Project Title: SMART PAW LEARNING

1- List of Participants

Name	Project Role
Subhan	Model Integration & Dataset Preparation
Haris	Mobile App UI/UX & Firebase Integration
Asaad	FastAPI & Backend Services

2- Meeting Agenda

- Review of AI models and project setup for FYP Phase II
- Dataset availability and image/text preprocessing

3- Agenda Points discussed in meeting

- Confirmed the use of EfficientNet and DistilBERT as AI models
- Discussed proper formatting and cleaning of image and text datasets
- Established deadlines for completing AI model training

A9. FYP fortnightly sign-up sheet

• Course: BSCS BSAI BSSE □ FYP-1

FYP-2 Project

Code: FYP-041/FL24 Project Name: SMART PAW LEARNING

Group Members Names & Reg#: Subhan(571-2019) Haris Bin Mohsin(1516-2020) M Asaad Ibrahim(2602-2021)

Supervisor Name: Sir Razaque Co-Supervisor's Name:

Meeting #	Date	Agenda (Brief Statement)	Attended By (Students' Names Only)	Supervisor's Sign
1	19-Dec-2024	Initial discussion on SmartPaw scope, tech stack, and roles.	Subhan, Haris, Asaad	
2	04-Jan-2025	Finalized datasets and backend architecture using FastAPI.	Subhan, Haris, Asaad	
3	18-Jan-2025	Developed layout designs and started implementing EfficientNet model.	Subhan, Haris, Asaad	
4	30-Jan-2025	Integrated image model API with Android frontend and reviewed login/pet UI.	Subhan, Haris, Asaad	
5	13-Feb-2025	Implemented text model (DistilBERT) and symptom input.	Subhan, Haris, Asaad	
6	27-Feb-2025	Connected Firebase for pet meal reminders and push notifications.	Subhan, Haris, Asaad	
7	12-Mar-2025	Developed doctor consultation fragment and refined bottom navigation.	Subhan, Haris, Asaad	
8	30-Mar-2025	Completed full AI flow (image + text) and tested prediction result dialogs.	Subhan, Haris, Asaad	
9	15-Apr-2025	Polished UI/UX, optimized performance, and prepared demo build.	Subhan, Haris, Asaad	
10	30-Apr-2025	Conducted internal demo with supervisor and received feedback for improvements.	Subhan, Haris, Asaad	
11	16-Jun-2025	Submitted final report, documentation, and prepared for external evaluation.	Subhan, Haris, Asaad	