

AI Based Symptom Analysis and Ayurvedic Treatment Suggestions for Pets

Sahana L, Assistant Professor, Department of Computer Science

St. Joseph's First Grade College Jayalakshmpuram, Mysore-570012

INTRODUCTION

1.1 DOMAIN INTRODUCTION

Artificial Intelligence (AI) is a field that aims to create machines, primarily computer systems, capable of emulating human cognitive abilities. This involves developing systems that can learn by acquiring information and rules to process that knowledge. Additionally, AI systems are designed to use reasoning capabilities, following specific rules to arrive at approximate or definite conclusions. Furthermore, these systems possess self-correction mechanisms, allowing them to refine their knowledge and performance over time.

Machine Learning (ML) is a subset of artificial intelligence that focuses on the development of algorithms and statistical models that enable computers to learn from and make predictions or decisions based on data. Instead of being explicitly programmed to perform a task, machine learning systems improve their performance over time by identifying patterns and relationships within the data.

1.2 Overview of Project

The "AI-Based Symptom Analysis and Ayurvedic Treatment Suggestions for Pets" project is a web-based intelligent system designed to revolutionize the way pet health issues are diagnosed and treated. This system leverages the power of machine learning to accurately classify diseases based on user-provided symptoms and then uses an integrated Ayurvedic knowledge base to recommend natural treatments.

Upon successful classification of the disease, the system activates the Ayurvedic Recommendation Engine. This module contains a curated repository of Ayurvedic treatments including herbal remedies, dietary advice, and lifestyle modifications tailored to various pet diseases. The recommendation engine uses both rule-based and AI-driven logic to ensure that the treatment suggestions are personalized, effective, and safe for pets.

The system architecture includes a front-end interface built with HTML and CSS for ease of access, and a back-end developed using Flask, which manages input processing, model invocation, and output generation. Libraries such as TensorFlow, Keras, NumPy, and Pandas are used to support model training and inference.

This project addresses the limitations of traditional veterinary care, including high costs, limited access in rural areas, and side effects from pharmaceutical treatments.

The inclusion of Ayurvedic treatments offers a natural and holistic alternative, promoting pet wellness and minimizing health risks. The final deliverable is a web-based platform accessible via browsers, enabling pet owners to manage their pet's health more proactively and naturally, with minimal reliance on veterinary visits unless necessary.

1.3 OBJECTIVES

The primary aim of this project is to develop an AI-Based Symptom Analysis and Ayurvedic Treatment Suggestion System for Pets that utilizes machine learning algorithms to identify diseases based on symptoms and recommend natural Ayurvedic treatments. This system aims to provide accessible, cost-effective, and side-effect-free treatment solutions while enhancing early disease detection in pets. Additionally, the system aims to promote proactive health management by using predictive analytics to identify pets at risk of developing chronic conditions, enabling early intervention. It also monitors health trends over time to adjust treatment plans as needed. By combining AI insights with Ayurvedic remedies,

the platform can offer natural treatment suggestions that align with traditional practices, promoting preventive care through dietary recommendations and lifestyle adjustments.

1.4 EXISTING SYSTEM

Pet health diagnosis primarily relies on veterinary consultations. Conventional treatment involves pharmaceutical drugs, which may have potential side effects. Ayurveda-based treatments for pets exist but lack systematic symptom-based recommendations. Online pet health platforms provide general guidelines but do not use AI for personalized treatment.

1.4.1 DISADVANTAGES OF EXISTING SYSTEM

1. **LIMITED ACCESSIBILITY** – Veterinary services may not be readily available in rural areas.
2. **HIGH COST** – Conventional treatments and veterinary consultations can be expensive.
3. **SIDE EFFECTS** – Synthetic medications may cause adverse reactions in pets.
4. **TIME-CONSUMING** – Diagnosis and treatment require multiple visits to a veterinary clinic.
5. **LACK OF PERSONALIZATION** – Ayurveda-based treatments lack AI-driven, symptom-specific recommendations.

1.5 PROPOSED SYSTEM

The proposed AI-based system leverages machine learning models trained on pet symptom datasets to classify diseases and recommend Ayurvedic treatments. Users input pet symptoms into the system, which then analyzes the data and suggests relevant herbal remedies, diet plans, and holistic treatments. The system integrates an intelligent recommendation engine based on traditional Ayurvedic knowledge combined with AI-driven analytics.

1.5.1 ADVANTAGES OF PROPOSED SYSTEM

1. **EARLY DISEASE DETECTION** – Machine learning models identify patterns in symptoms to diagnose diseases at an early stage.
2. **PERSONALIZED TREATMENT** – The system provides customized Ayurvedic remedies based on specific symptoms and pet conditions.
3. **COST-EFFECTIVE** – Reduces the need for frequent veterinary consultations, making healthcare more affordable.
4. **NO SIDE EFFECTS** – Ayurvedic treatments are natural and minimize the risk of adverse reactions.
5. **24/7 AVAILABILITY** – The AI system is accessible anytime, providing immediate recommendations.
6. **USER-FRIENDLY INTERFACE** – Designed for easy use by pet owners with minimal technical knowledge.

LITERATURE SURVEY

2.1 INTRODUCTION

A literature survey, or literature review, is a comprehensive overview of existing research and publications on a specific topic, serving several essential purposes in academic and professional contexts. A literature survey is vital for successful project outcomes, as it systematically reviews existing research and methodologies related to the topic. This process helps inform project objectives, identify knowledge gaps, and refine research questions while providing insights into best practices and challenges. It enhances credibility by demonstrating engagement with scholarly work and typically includes an introduction, methodology, thematic organization, critical analysis, and a conclusion. Ultimately, integrating a literature

survey into project planning ensures a solid foundation in established knowledge, boosting the project's potential for success and innovation.

2.2 SURVEY PAPERS

1] Title: Artificial Intelligence in Veterinary Diagnostics

Authors: Catherine Stowell, Beth Lane

Year: 2023

This project explores the transformative impact of Artificial Intelligence (AI) on veterinary diagnostics, aiming to enhance accuracy, efficiency, and early disease detection. It utilizes a comprehensive dataset comprising X-ray, ultrasound, MRI scans, and clinical records from veterinary hospitals, all labeled with disease conditions and expert diagnoses. The system employs Convolutional Neural Networks (CNN) for image diagnostics and Support Vector Machines (SVM) and Random Forest classifiers for predictive analytics based on clinical data. AI model training, integration and deployment as a cloud-based or on-premise application, and a user-friendly interface that allows veterinarians to upload images or input symptoms for AI-generated diagnoses and treatment suggestions. The CNN model achieves an impressive accuracy of 85-95% for image classification, while SVM and Random Forest classifiers reach 80-90% accuracy for clinical predictions. Technologies utilized in the project include TensorFlow and PyTorch for machine learning, Python for programming, and cloud platforms like AWS and Google Cloud for deployment. However, the project faces challenges such as the need for large, high-quality datasets, significant computational costs for training deep learning models, difficulties in interpreting AI-generated diagnoses, and ethical concerns regarding the potential reduction of human expertise in veterinary diagnostics.

2] Title: Online Procurement of Pet Supplies and Willingness to Pay for Veterinary Telemedicine

Authors: Widmar, Slipcherko,

Year: 2020

This study examines consumer behaviour related to online pet supply purchases and their willingness to pay for veterinary telemedicine services. Utilizing survey data from pet owners, the research identifies factors influencing online shopping decisions and acceptance of remote veterinary consultations, shedding light on trends in pet care e-commerce and the digitization of veterinary services. The dataset includes demographics, purchasing frequency, willingness to pay for telemedicine, pet type, and veterinary visit frequency. The analysis employs logistic regression to assess willingness to pay, decision trees to segment pet owners based on purchasing behaviour, and K-means clustering to group respondents by preferences. The proposed system integrates an e-commerce portal for pet supplies with a subscription-based telemedicine service and an AI-driven recommendation system for personalized pet care. The logistic regression model achieved approximately 80% accuracy in predicting willingness to pay, while decision trees provided 75-85% accuracy in classifications. Technologies used include Python for data analysis, Django/Flask for web development, and AWS/GCP for cloud services. Challenges include potential data bias, privacy concerns regarding pet medical records, adoption barriers for telemedicine, and varying regulatory frameworks for veterinary telemedicine across regions.

3] Title: AI-Based Detection of Animal Diseases Using Image Processing and Supervised Learning

Authors: Jackson Akpojaro

Year: 2020

AI-based detection of animal diseases employs image processing techniques and supervised learning algorithms to enhance veterinary diagnostics' accuracy and efficiency. This approach involves preprocessing medical images to improve quality, followed by feature extraction to identify disease indicators. Supervised learning models, such as Random Forests,

Support Vector Machines, and Convolutional Neural Networks, are trained on labeled datasets of healthy and diseased animals to automate disease identification. The benefits include early detection, increased efficiency, and improved accuracy in diagnostics. Applications span livestock health monitoring, pet care, and wildlife conservation. However, challenges such as data quality, model interpretability, and regulatory compliance remain. Future directions include integrating AI with wearable technology for continuous monitoring and expanding the range of detectable diseases, promising significant advancements in animal health care. Overall, AI-based detection of animal diseases represents a significant advancement in veterinary medicine, promising to transform animal health care and management practices.

4] Title: Artificial Intelligence for Enhancing Veterinary Healthcare: An Experiment

Authors: Abdulrhman Alkhanifer and Ahmad AlZubi

Year: 2024

The project "Artificial Intelligence for Enhancing Veterinary Healthcare: An Experiment" by Abdulrhman Alkhanifer and Ahmad AlZubi investigates the transformative role of artificial intelligence (AI) in veterinary medicine, specifically targeting the enhancement of disease detection and diagnosis in livestock. The research centers on the application of machine learning algorithms, with a focus on a DenseNet-121 Convolutional Neural Network (CNN), to analyse images of cattle for the early identification of diseases such as Lumpy Skin Disease (LSD), which poses significant economic threats to the livestock industry. To achieve this, the researchers compiled a comprehensive dataset consisting of images from various cattle breeds, ensuring a diverse representation of the target population. The CNN model was trained to recognize patterns and features indicative of LSD, enabling it to differentiate between healthy and affected animals. The results were promising, with the model achieving a training accuracy of 99.76% and a validation accuracy of 92.17%, underscoring its potential for reliable diagnostics. In addition to improving diagnostic accuracy, the project emphasizes the broader implications of AI in veterinary practice, such as reducing the time and resources required for disease identification, facilitating early intervention, and ultimately enhancing animal welfare. By integrating AI technologies into routine veterinary care, the study advocates for a shift towards more proactive and efficient health management strategies in livestock, which could lead to better outcomes for farmers and the agricultural sector as a whole. This research not only contributes to the field of veterinary medicine but also highlights the potential for AI to revolutionize animal health management practices globally.

5] Title: Detection of Disease in Calves using Artificial Intelligence

Authors: Abdulrhman Alkhanifer and Ahmad AlZubi

Year: 2024

The detection of diseases in calves using artificial intelligence (AI) is a transformative approach in veterinary medicine that enhances early diagnosis and improves animal health management. By leveraging advanced technologies such as machine learning and image processing, AI can analyse visual data from high-resolution images and sensor data from wearable devices to identify health issues like respiratory diseases, gastrointestinal disorders, and metabolic conditions. Early detection is crucial for preventing the spread of diseases, reducing economic losses for farmers, and improving animal welfare. AI techniques, including supervised and unsupervised learning, enable accurate classification of health conditions, while predictive analytics can forecast potential outbreaks. Despite the benefits, challenges such as data quality, integration into existing practices, and ethical considerations must be addressed.

SOFTWARE REQUIREMENTS

3.1 INTRODUCTION

A Software Requirements Specification (SRS) is a comprehensive document that captures and articulates all the essential aspects of a software project. It serves as a blueprint for the development team, stakeholders, and end-users, ensuring that all parties share a common understanding of the system's goals, functionalities, and constraints. The primary objective of an SRS is to lay the foundation for successful system development by clearly defining the functional and non-functional

requirements. In the context of the "AI-Based Symptom Analysis and Ayurvedic Treatment Suggestions for Pets" project, this SRS outlines the requirements of a web-based intelligent system designed to diagnose pet health conditions based on symptom input and recommend Ayurvedic treatments. The document specifies what the system will do, how it will interact with users, what technologies will be used, and the constraints under which it will operate. This is especially vital given the integration of AI and traditional Ayurvedic knowledge, which requires accurate symptom recognition, disease classification, and holistic treatment generation. The document is structured to include the scope of the project, definitions of technical terms, a general overview of the system, functional and non-functional requirements, and a feasibility study. These components collectively ensure a full understanding of the system for both technical and non-technical stakeholders. With this SRS, the development team will be equipped to implement the system efficiently while ensuring that the needs of pet owners and veterinarians are effectively addressed.

3.2 SCOPE

The scope of this project is to develop an AI-driven web application that accepts pet symptom inputs and provides accurate disease classification along with personalized Ayurvedic treatment suggestions. The system will use machine learning models trained on relevant pet health datasets and integrate Ayurvedic remedies such as herbal treatments and dietary plans. The platform will be user-friendly and accessible via web browsers, ensuring 24/7 availability for pet owners. It aims to reduce dependence on conventional veterinary services, cut costs, and promote natural healthcare alternatives, thereby improving the quality and accessibility of pet health management.

3.3 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Here is a comprehensive list of key definitions, acronyms, and abbreviations used throughout the document:

Definitions

- **ARTIFICIAL INTELLIGENCE (AI):** A branch of computer science that simulates human intelligence processes such as learning, reasoning, and problem-solving by machines.
- **MACHINE LEARNING (ML):** A subset of AI that allows systems to automatically learn from data and improve performance without being explicitly programmed.
- **SYMPTOM ANALYSIS:** The process of identifying, categorizing, and evaluating observable signs of illness in pets.
- **AYURVEDA:** An ancient Indian system of natural and holistic medicine that focuses on balancing bodily systems using diet, herbal treatment, and yogic breathing.
- **TREATMENT RECOMMENDATION ENGINE:** A software module that suggests suitable remedies based on diagnosed conditions.
- **WEB APPLICATION:** A software application that runs on a web server and is accessed through a web browser.
- **USER INTERFACE (UI):** The point of interaction between the user and the software application.
- **BACKEND:** The part of the application that handles data processing, logic, and server-side functions.
- **FRONTEND:** The visual part of the web application that users interact with.
- **MODEL TRAINING:** The process of feeding data into a machine learning algorithm to learn patterns for classification or prediction tasks.

ABBREVIATIONS

- **FLASK:** A lightweight Python web framework used to build server-side applications.
- **TENSORFLOW:** An open-source library used for training and deploying machine learning models.

- **KERAS:** A high-level neural network API that runs on top of TensorFlow.
- **NUMPY:** A Python library used for scientific computing with support for arrays.
- **PANDAS:** A data manipulation and analysis library in Python.
- **JUPYTER NOTEBOOK:** An IDE used for developing machine learning models interactively.
- **ANACONDA:** A distribution of Python for scientific computing and machine learning.
- **PETMLDATA:** Internal reference to the collected pet symptom and disease dataset.
- **AYURRECOENGINE:** The component of the system that provides Ayurvedic remedy suggestions.
- **PETDIAGMODEL:** The trained machine learning model used for disease classification.

SYSTEM DESIGN

4.1 INTRODUCTION

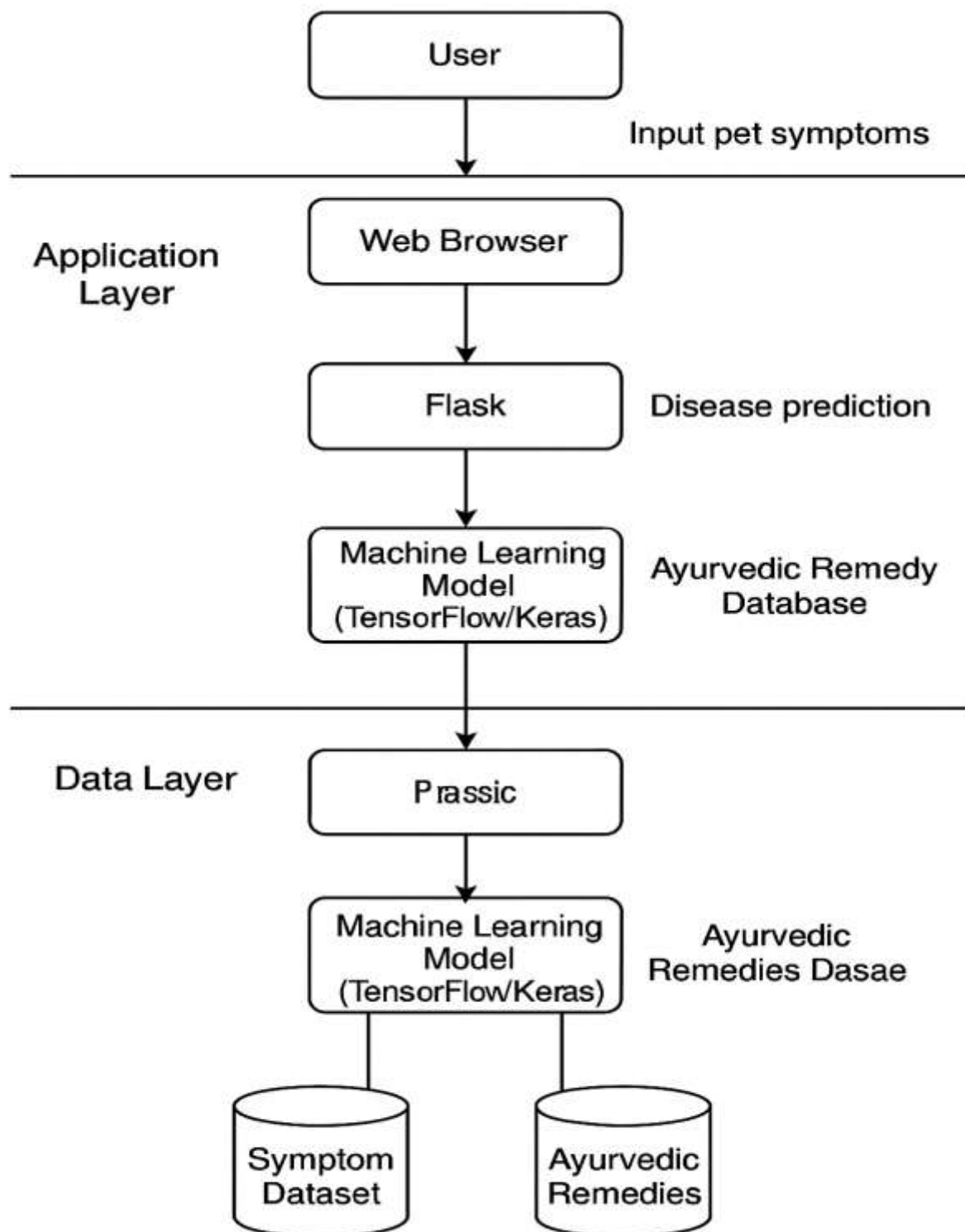
System design is the blueprint for building a software solution that aligns with project requirements and user expectations. It involves defining the architecture, components, modules, data flow, and user interactions. The goal is to transform the functional and non-functional requirements specified in the Software Requirements Specification (SRS) into a technical solution that can be implemented efficiently.

For the "AI-Based Symptom Analysis and Ayurvedic Treatment Suggestions for Pets" system, the design phase is particularly critical. The system integrates machine learning-based disease classification with traditional Ayurvedic knowledge, demanding a modular, scalable, and responsive architecture. The system design ensures seamless interaction between users and the backend ML engine, while managing data integrity, security, and usability.

The architecture must support features such as:

- Symptom input and processing
- Real-time disease prediction
- Ayurvedic remedy generation
- User-friendly interface
- Admin control and data management

4.2 ARCHITECTURE DIAGRAM



The architecture diagram for the proposed system follows a three-tier web-based architecture that clearly separates concerns across the presentation, application, and data layers. At the top, the presentation layer acts as the user interface where pet owners interact with the system through a web browser. This interface is built using HTML and CSS to ensure accessibility and user-friendliness. Users can input observed symptoms of their pets into a form, which is then transmitted to the application layer through RESTful API calls. The application layer, powered by the Flask framework in Python, handles the core logic of the system. It validates and preprocesses the incoming data, and invokes the machine learning

model trained using TensorFlow and Keras to predict the disease. After the disease is identified, the system accesses the Ayurvedic remedy database to retrieve appropriate treatment suggestions. These suggestions may include herbal solutions, diet modifications, and lifestyle practices. The data layer at the backend stores the symptom-disease datasets and Ayurvedic treatments using a structured database, either SQLite or MySQL. This tiered architecture ensures that each part of the system operates independently but in coordination with others, leading to easier maintenance, scalability, and modular development. Furthermore, the design supports asynchronous processing and future integration with additional features such as voice input or image-based diagnosis. The separation of frontend and backend logic also simplifies debugging and upgrades. Overall, this architecture provides a robust foundation for a real-time, intelligent recommendation platform tailored to pet health and wellness using both AI and Ayurveda.

4.3 DATAFLOW DIAGRAM

DFD Level 0 – Explanation

The Level 0 Data Flow Diagram (DFD) provides a high-level overview of the entire system, showcasing how data flows between external entities and the main processing unit. At this level, the entire system is encapsulated as a single process—Symptom Analysis and Treatment System. The pet owner acts as the primary user who inputs symptoms observed in their pet through a web interface. These symptoms are submitted to the central system process, which then interprets the data using its internal logic and machine learning capabilities. The system processes this information to classify the disease and then connects with an Ayurvedic treatment database to extract suitable remedies. These remedies are returned to the pet owner as a combination of herbal treatments.

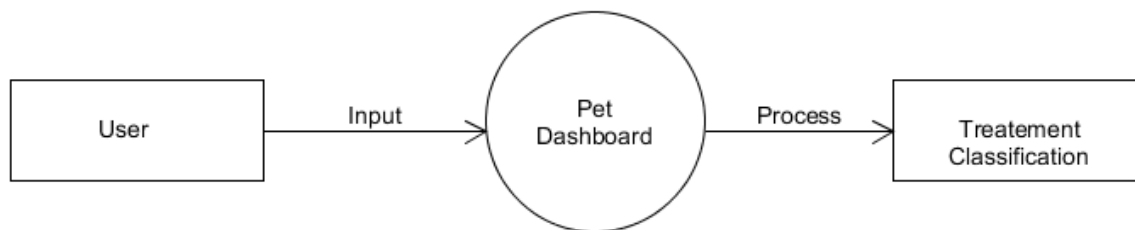


Fig 4.3.1 DFD Level 0

DFD Level 1 – Explanation

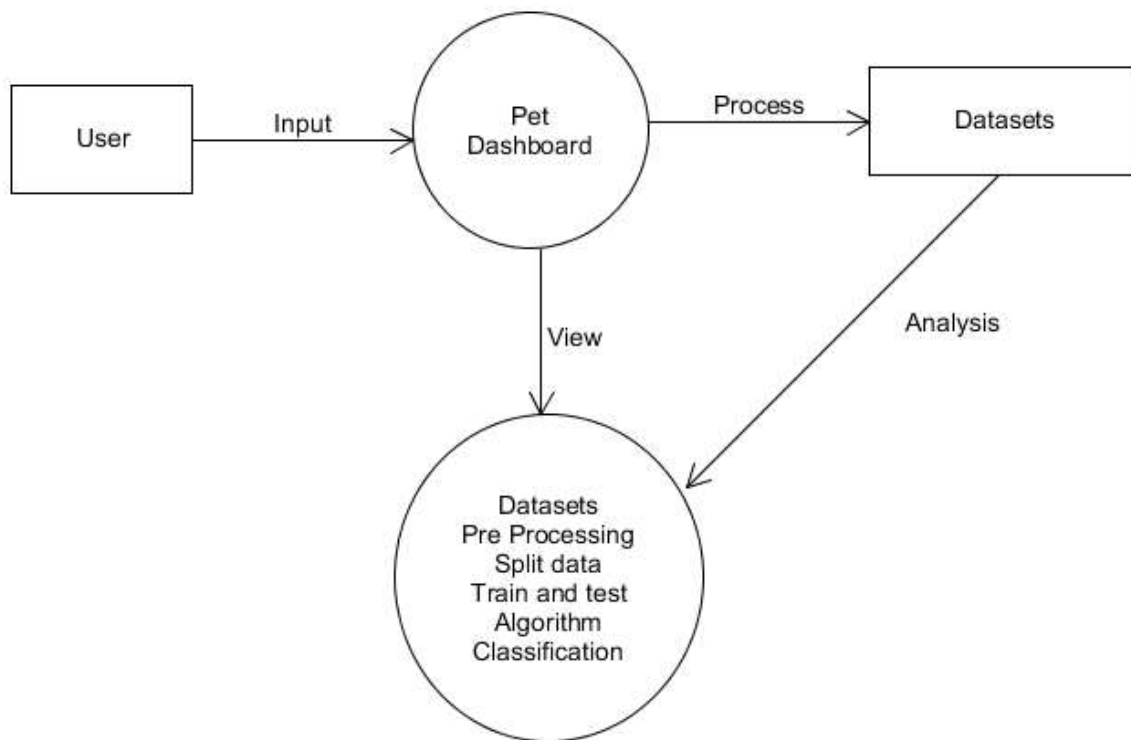
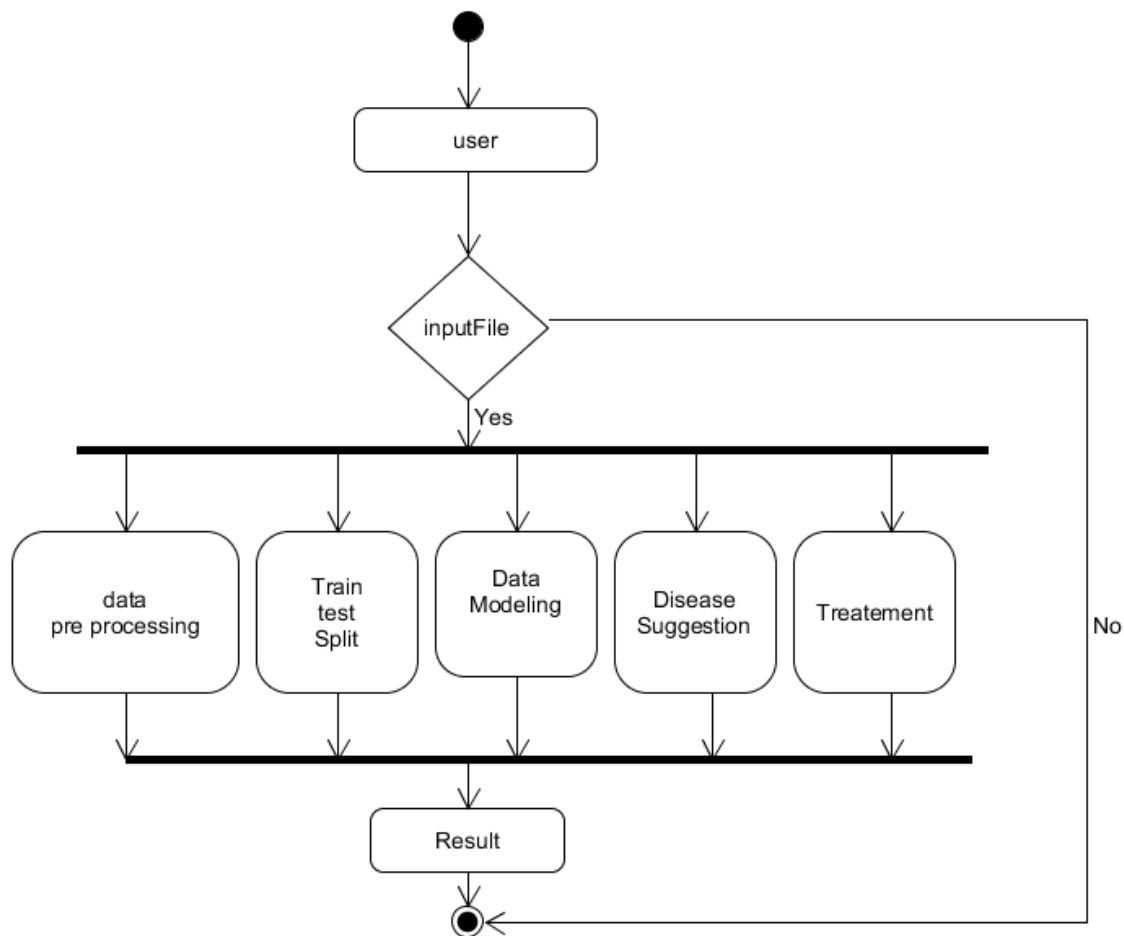


Fig 4.3.2 DFD Level 1

The Level 1 Data Flow Diagram (DFD) expands upon the high-level abstraction provided in Level 0 by breaking down the single main process into five more specific sub-processes. These detailed components provide insight into how the system handles the flow of data from input to output. The first subprocess is **Input Symptom Data**, which involves collecting structured input from the user through a form-based web interface. This data is then passed to the **Data Preprocessing** module, where the system performs validation, cleaning, and formatting to ensure compatibility with the trained machine learning model. The preprocessed symptoms are then forwarded to the **Disease Prediction** subprocess, which leverages a trained AI model (such as Random Forest or Deep Learning) to classify the disease based on recognized symptom patterns. Once the disease is predicted, the system engages the **Fetch Ayurvedic Treatments** process. Finally, the **Display Output** process compiles the diagnosis and treatments into a user-friendly format and presents it on the web interface. The data stores involved in this DFD include the **Symptom-Disease Dataset**, used for prediction, and the **Treatment Repository**, used for remedy lookup.

4.4 ACTIVITY DIAGRAM



The activity diagram highlights how the system handles symptom input from start to finish. The user journey begins when they access the system and ends with the display of recommendations. Internally, it shows how different components of the backend interact—especially how validation, model prediction, and treatment mapping occur in a sequence. This diagram ensures clarity in the user interaction flow and helps developers understand state transitions and dependencies.

Basic Notation:

Initial Activity



This shows the starting point or first activity of the flow. It is denoted by solid circle

Final Activity



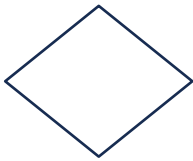
The end of the Activity diagram is shown by a bull's eye symbol.

Activity



Activities are represented by rectangle with rounded edges.

Decisions



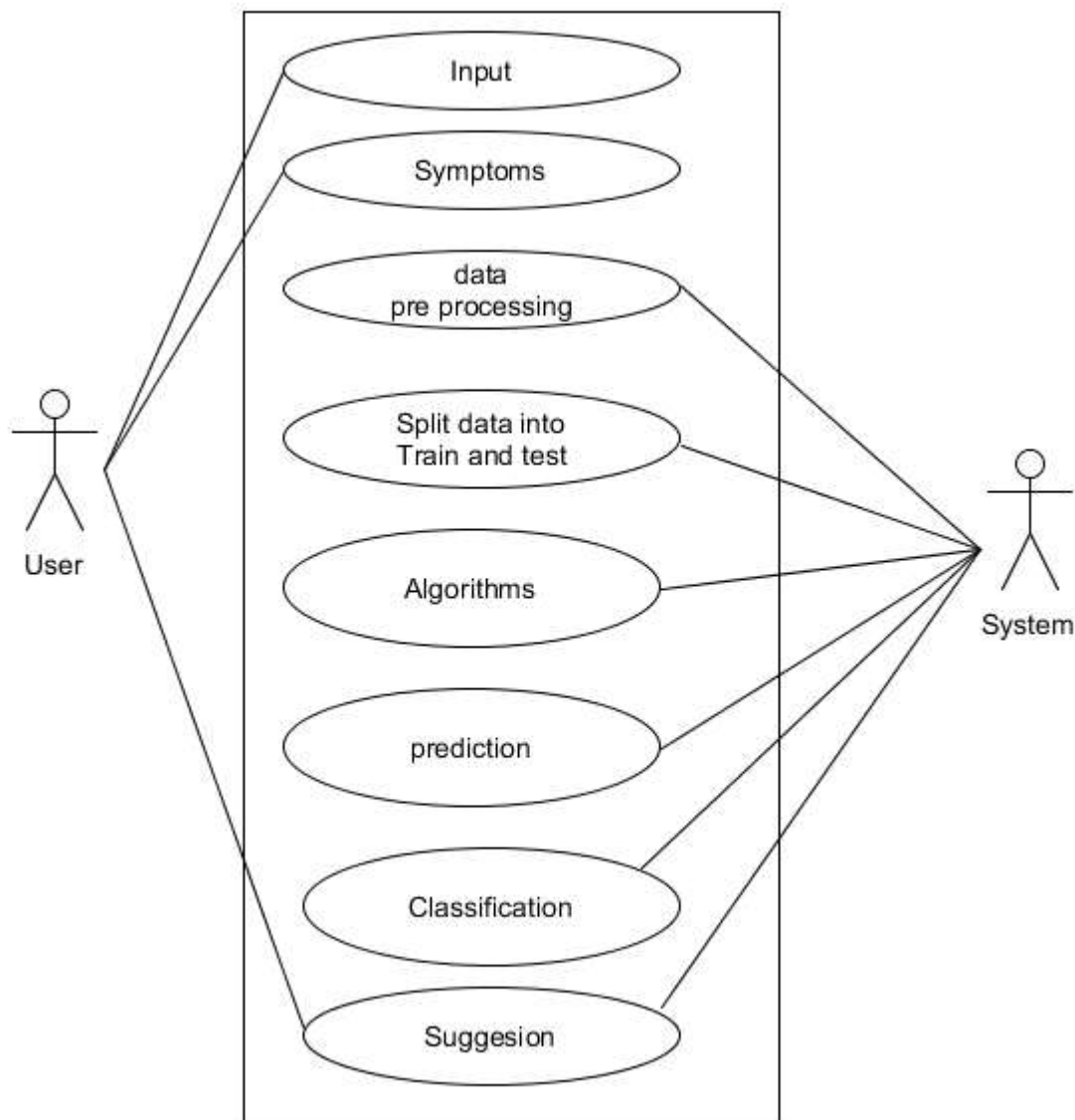
A logic where a decision is to be made is depicted by a diamond.

Workflow



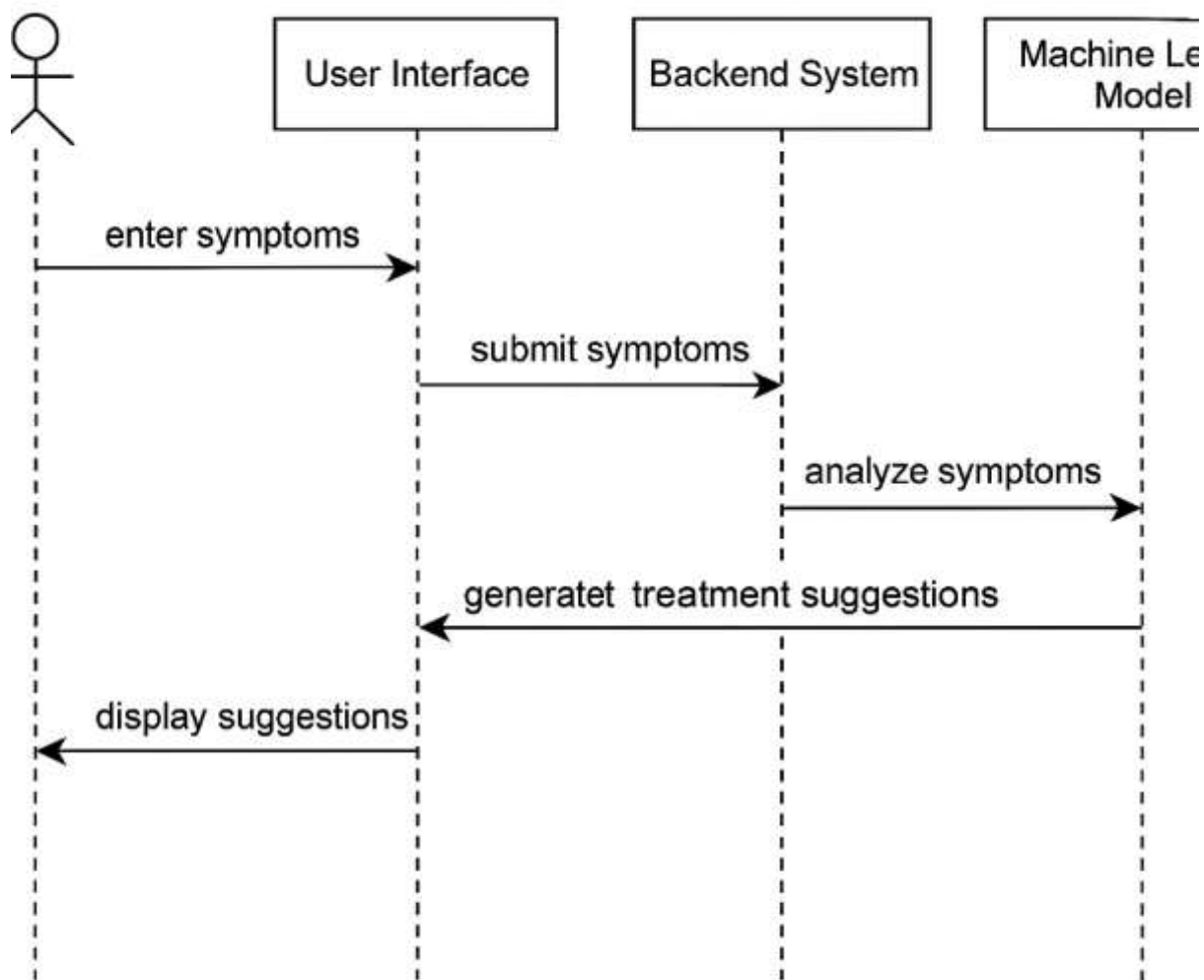
Workflow is depicted with an arrow. It shows the direction of the workflow in activity diagram

4.6 Use Case Diagram with Explanation



This diagram focuses on what users can do with the system rather than how they do it. The pet owner can input symptoms, receive a diagnosis, and obtain treatment suggestions. The admin, on the other hand, can manage remedy records and oversee system performance. Use case diagrams are essential in identifying functional boundaries and ensuring the development team captures all required user-system interactions.

4.7 Sequence Diagram with Explanation



The sequence diagram clarifies the real-time communication between system components. It captures the chronological order of interactions needed to produce a treatment suggestion from user input. This helps backend developers ensure proper API routing and module integration.

IMPLEMENTATION

5.1 INTRODUCTION

The increasing adoption of pets and the rising awareness regarding pet health have led to the emergence of advanced technological solutions aimed at ensuring early disease diagnosis and holistic treatment. Conventional veterinary visits, while essential, may not always be accessible, especially for pet owners in remote or rural areas. Moreover, minor symptoms are often overlooked, resulting in delayed treatment and deterioration of pet health. In such scenarios, Artificial Intelligence (AI)-based systems can serve as intelligent assistants to help pet owners identify probable diseases and take early precautions.

The implementation of the system involves several critical stages. These include data collection, data preprocessing, model selection and training, saving the trained model, and deploying the model into a user-friendly web interface using Flask. Furthermore, to make the system informative and trustworthy, a domain knowledge module is integrated. This module contains detailed information about each disease, along with Ayurvedic treatment guidelines and preventive measures. Together, these components form a comprehensive AI-based health advisory platform for pets. In the subsequent sections, we delve deep into the implementation details, including the dataset used, model training strategies, algorithm design, and deployment process.

5.3 IMPLEMENTATION WITH RESPECT TO OUR PROJECT

The implementation phase of our project focuses on transforming the conceptual design into a working solution that can be utilized by end-users, specifically pet owners and caregivers. To achieve this, the project was executed in multiple sequential steps to ensure accuracy, usability, and reliability.

The first step involved acquiring and analyzing a dataset suitable for pet disease prediction. The dataset used in this project consisted of 10,000 records, each containing details such as pet type, age, gender, weight, and four reported symptoms, alongside the diagnosed disease. Exploratory data analysis (EDA) was carried out to check for missing values, understand the data distribution, and prepare it for model training. Label encoding was applied to convert categorical data into numerical format, which is essential for machine learning algorithms.

Next, the machine learning models were trained. Several classification algorithms were experimented with, including K-Nearest Neighbors (KNN), Naive Bayes, Decision Tree, and Random Forest. After comparative analysis, the Random Forest Classifier was finalized for deployment due to its superior performance, achieving an accuracy of approximately 98% on the test dataset. This model demonstrated robustness in handling categorical data and exhibited excellent generalization capability.

For real-world usage, the trained model, along with label encoders, was saved using Python's pickle module. This allowed for easy loading and prediction during the Flask web application's runtime. The Flask application was developed to serve as the front-end interface. It accepts user inputs (pet details and symptoms), performs predictions using the trained model, and displays the predicted disease along with Ayurvedic treatments and precautions stored in a separate knowledge module. This module enriches the user experience by providing natural and safe remedies, thus making the platform not only predictive but also advisory.

The entire system is hosted locally and can be further scaled to cloud deployment to make it widely accessible.

5.4 ALGORITHM EXPLANATION

The core of our project lies in the ability of the machine learning algorithms to accurately predict pet diseases based on input symptoms and other pet attributes. In this project, after experimenting with various algorithms, **Random Forest Classifier** was selected as the final model due to its accuracy and robustness. In this section, we will explain the algorithms used and why Random Forest emerged as the most suitable choice.

5.4.1 DATA PREPROCESSING AND LABEL ENCODING

Before any algorithm can be applied, it is essential to preprocess the dataset:

- **Handling Categorical Variables:** Pet type, Gender, Symptom_1, Symptom_2, Symptom_3, Symptom_4, and Disease are categorical variables. Machine learning models require numerical inputs. Thus, Label Encoding was applied to convert these text labels into integer format. Label Encoder from Scikit-learn was used for this task.
- **Handling Missing Values:** The dataset was clean and did not have missing values. This ensured smooth training without imputation.
- **Splitting the Dataset:** The dataset was split into training and testing sets using `train_test_split` from Scikit-learn, with 80% data used for training and 20% for testing.

This preprocessed dataset was then fed into different classification algorithms to determine which model provided the best results.

5.4.2 K-NEAREST NEIGHBORS (KNN)

KNN Algorithm is a simple and widely used classification algorithm that works on the principle of similarity. When a new data point is encountered, the algorithm searches for the "k" nearest data points in the training dataset and assigns the class that is most common among them.

Working in our case:

- Each pet and its symptoms were treated as feature vectors.
- For every new input, KNN calculated distances between the input and all training samples.
- Based on the nearest neighbors (k=5 in our case), the class (disease) with the majority votes was assigned.

Limitations:

- KNN performed relatively poorly (56% accuracy).
- This is because KNN does not work well with high cardinality categorical data and is sensitive to irrelevant features.
- Hence, KNN was not selected as the final algorithm.

5.3.3 NAIVE BAYES CLASSIFIER

Naive Bayes Algorithm is a probabilistic classifier based on Bayes' Theorem. It assumes that the features are independent of each other, which simplifies computation.

Working in our case:

- Each symptom was treated as a feature.
- Naive Bayes calculated the probability of each disease given the symptoms.
- The disease with the highest probability was selected.

Limitations:

- The independence assumption is violated here because symptoms are often correlated.
- Accuracy was low (~40%), making it unsuitable for accurate prediction.
- Naive Bayes worked poorly for diseases that have subtle symptom differences.

5.3.4 Decision Tree Classifier

Decision Tree Algorithm works by creating a tree-like structure where each node represents a feature (symptom), and branches represent decision rules, leading to a prediction at the leaf node.

Working in our case:

- The Decision Tree learned rules like "if Symptom_1 = Red Skin and Pet = Dog, then Disease = Skin Allergy."
- It split data recursively until the stopping condition was met.

- The tree structure was able to capture the relations and interactions between different symptoms effectively.

Performance:

- Decision Tree achieved about 97% accuracy, which is quite good.
- However, Decision Trees can easily overfit to training data unless carefully pruned.

5.3.5 RANDOM FOREST CLASSIFIER

Random Forest Algorithm is an ensemble method that creates a "forest" of Decision Trees and merges their predictions. Each tree votes and the majority class is chosen as the final prediction.

Working in our case:

- Multiple Decision Trees were trained on random subsets of data and features.
- For each prediction, all trees voted for a disease class.
- The final prediction was made based on the majority vote.

Why Random Forest was selected:

- **High Accuracy:** Achieved ~98% accuracy, outperforming all other models.
- **Robustness:** Reduces overfitting, which is a common issue with single Decision Trees.
- **Handles Categorical Data:** Able to handle both numerical and categorical data without complex transformations.
- **Feature Importance:** Identifies important symptoms which contribute more to predictions.
- **Generalization:** Works well even if there is noise or redundant features.

Limitations:

- Random Forest models can be slightly slower for real-time applications due to multiple tree evaluations.
- They are also relatively complex to interpret compared to simpler models.

5.3.6 MODEL SAVING AND INTEGRATION

Once the Random Forest model was trained and finalized:

- It was saved using **Pickle**, which allows easy loading of the model without retraining.
- Label Encoders were also saved to ensure that user inputs during prediction are encoded in the same way as during training.

5.3.7 FLASK WEB APPLICATION (DEPLOYMENT)

A lightweight Flask web application was developed to serve as the interface for end-users. The application performs the following tasks:

- Accepts user input (Pet type, Age, Gender, Weight, and Symptoms).
- Encodes the input using saved Label Encoders.
- Loads the trained Random Forest model and makes a prediction.
- Maps the predicted disease to Ayurvedic treatment suggestions, causes, and precautions.

- Displays all the information on the result page in a user-friendly format.

PSEUDOCODE OF EACH ALGORITHM

K-Nearest Neighbors (KNN)

KNN is a simple yet powerful classification algorithm based on similarity (distance) between feature vectors. In this project, KNN was used to classify diseases based on how similar a pet's symptoms and attributes were to existing cases.

Pseudocode:

Input: Training dataset (features X_{train} , labels y_{train}), new pet data (x_{new}), k (number of neighbors)

Output: Predicted Disease class

Step 1: For each instance in X_{train}

- Calculate the distance between x_{new} and the instance
- Store the distance along with corresponding label

Step 2: Sort the calculated distances in ascending order

Step 3: Select the k nearest neighbors

Step 4: Count the frequency of each class among these k neighbors

Step 5: Return the class with the highest frequency (majority vote) as the predicted Disease

Explanation:

KNN does not involve model training but directly compares distances during prediction.

In this project, KNN performed poorly because it does not handle categorical data and scales poorly with large datasets.

Naive Bayes

Naive Bayes is a probabilistic classifier that assumes independence between features.

It uses Bayes' theorem to compute the probability of each class and selects the one with the highest posterior probability.

Pseudocode:

vbnet

CopyEdit

Input: Training dataset (X_{train} , y_{train}), new pet data (x_{new})

Output: Predicted Disease class

Step 1: Calculate prior probability for each disease:

$$- P(\text{Disease}) = (\text{Number of times Disease occurred}) / (\text{Total records})$$

Step 2: For each symptom/feature in x_{new}

- Calculate likelihood:

$$- P(\text{Symptom} | \text{Disease}) = (\text{Count of Symptom in Disease}) / (\text{Total Disease records})$$

Step 3: Multiply likelihood with prior probability to get posterior probability for each Disease:

$$- P(\text{Disease} | \text{Symptoms}) = P(\text{Disease}) * P(\text{Symptom1} | \text{Disease}) * P(\text{Symptom2} | \text{Disease})$$

Step 4: Choose the Disease with the highest posterior probability as prediction.

Explanation:

Although simple and fast, Naive Bayes performed poorly here due to the assumption of independence between symptoms, which often does not hold true in real-world disease scenarios.

DECISION TREE

Decision Tree works by creating a hierarchical structure of rules based on input features to classify data into different categories.

Pseudocode:

Input: Training dataset (X_{train} , y_{train}), new pet data (x_{new})

Output: Predicted Disease class

Step 1: Calculate entropy and information gain for all features

Step 2: Choose the feature with the highest information gain as root node

Step 3: Split the dataset based on the selected feature's possible values

Step 4: Repeat recursively for each child node until:

- All records in node belong to the same Disease class, or
- No further features left to split

Step 5: For prediction:

- Start from root node
- Traverse down the tree following conditions based on x_{new} features.

- Arrive at a leaf node which represents the predicted Disease

Explanation:

Decision Trees are easy to interpret and performed well in this project. However, they may overfit the training data, which impacts their ability to generalize.

RANDOM FOREST

Random Forest is an ensemble learning method that builds multiple Decision Trees and aggregates their predictions to produce a more robust result.

Pseudocode:

vbnet

CopyEdit

Input: Training dataset (X_{train} , y_{train}), new pet data (x_{new}), number of trees (n)

Output: Predicted Disease class

Training Phase:

Step 1: For each tree (from 1 to n):

- Create a bootstrap sample (random sample with replacement) from the training dataset
- Select a random subset of features
- Train a Decision Tree using this sample and subset

Prediction Phase:

Step 2: For each tree:

- Use the trained Decision Tree to predict the Disease for x_{new}

Step 3: Collect all tree predictions (votes)

Step 4: Determine the most frequent Disease prediction (majority voting)

Step 5: Return the Disease with the highest vote count as final prediction

Explanation:

Random Forest overcomes the overfitting problem of single Decision Trees by using multiple randomization. It delivered the best results in our project and was selected as the final model.

SYSTEM TESTING

6.1 INTRODUCTION TO TESTING

Testing is a critical stage in the software development life cycle. It is the process of evaluating a system or its components to verify that they meet the specified requirements and function correctly under various conditions. In the context of this project, testing plays an essential role in ensuring that the **AI-Based Symptom Analysis and Ayurvedic Treatment Suggestion System for Pets** operates reliably and accurately for different user inputs. This chapter describes how the developed system was tested to validate its functionality, accuracy, usability, and performance.

The primary objective of testing in this project is twofold: first, to ensure that the machine learning model provides accurate and consistent disease predictions based on pet attributes and symptoms; and second, to confirm that the web-based interface, built using Flask, properly captures user inputs and displays the prediction results along with the Ayurvedic treatments and precautionary suggestions.

Given the nature of this project, both backend (machine learning model) and frontend (Flask web app) were tested rigorously. Machine learning model testing involved validating the prediction accuracy using unseen data, checking prediction stability, and analyzing the prediction confidence. Flask application testing, on the other hand, focused on ensuring that all routes were functional, user inputs were handled appropriately, and the output rendered was as expected.

Additionally, usability testing was performed to make sure that even non-technical users could navigate the web interface smoothly. Multiple rounds of manual and automated testing were conducted to guarantee system reliability. This chapter provides a detailed explanation of the types of testing applied, test cases executed, and the results obtained during the validation process.

6.2 TYPES OF TESTING

Testing in this project can be categorized into the following major types:

6.2.1 UNIT TESTING

Unit testing is the process of testing individual components or modules of the system in isolation to ensure they function as expected. In the scope of this project, unit testing was conducted primarily for the following components:

- **Model Loading:** Ensure that the trained machine learning model and label encoders load properly from the saved .pkl files.
- **Prediction Function:** Validate that the prediction function can take input, encode it, and produce a prediction.
- **Ayurvedic Recommendation Mapping:** Ensure the correct disease maps to the relevant Ayurvedic treatment and precautionary suggestions.

These unit tests were manually executed during development using test scripts and Flask routes to validate the backend logic.

6.2.2 INTEGRATION TESTING

Integration testing verifies that different modules or services used by the application work well together. For our system, integration testing covered the following:

- **Model and Flask Integration:** Ensuring that when users submit their pet details and symptoms through the web form, the backend successfully receives the data, processes it, makes a prediction using the ML model, and fetches Ayurvedic suggestions.

- **Template Rendering:** Validate that predicted results, causes, treatments, and precautions are dynamically injected into HTML templates and properly displayed.
- **Input Encoding:** Ensure that user inputs, which are textual, are correctly converted using the label encoders before making a prediction.

Integration testing was essential to confirm that the end-to-end workflow functioned seamlessly, from data entry to final output display.

6.2.3 FUNCTIONAL TESTING

Functional testing verifies the functionality of the system against the defined specifications and user requirements. This included:

- **User Input Handling:** Checking if the form accepts valid data and provides appropriate error messages for missing or invalid data.
- **Prediction Output:** Ensuring the predicted disease is valid and meaningful based on the input symptoms.
- **Ayurvedic Suggestion Display:** Confirming that correct treatment and precaution information is fetched for each disease.

All expected functional behaviors were verified by simulating various user scenarios.

6.2.4 USER INTERFACE (UI) AND USABILITY TESTING

Usability testing was performed to ensure that non-technical users could easily use the application without confusion. Key aspects tested:

- **Form Design and Navigation:** Ensuring that the input form is clear, and the instructions are easy to follow.
- **Result Page Readability:** Ensuring that predicted disease, causes, Ayurvedic suggestions, and precautions are displayed in an organized and readable format.
- **Navigation Flow:** Testing logout, home navigation, and about page links for correct redirection.

Usability testing involved testing by various users (students and volunteers) who provided feedback on ease of use.

6.2.5 PERFORMANCE TESTING

While the system is not highly computationally intensive, performance testing was conducted to verify:

- **Response Time:** Ensuring that predictions are returned within an acceptable timeframe (less than 1 second on average).
- **Multiple Requests Handling:** Simulating multiple simultaneous requests to confirm that the Flask application remains responsive.

Performance remained satisfactory during all tests.

6.2.6 ACCURACY TESTING (ML MODEL EVALUATION)

Accuracy testing involved checking how well the trained Random Forest model performed when provided with unseen test data. This was done during the model evaluation phase using:

- **Accuracy Score:** Achieved ~98% accuracy on test data.

- **Confusion Matrix and Classification Report:** Analyzed to check precision, recall, and F1-score for each disease category.

This testing validated the reliability of disease predictions.

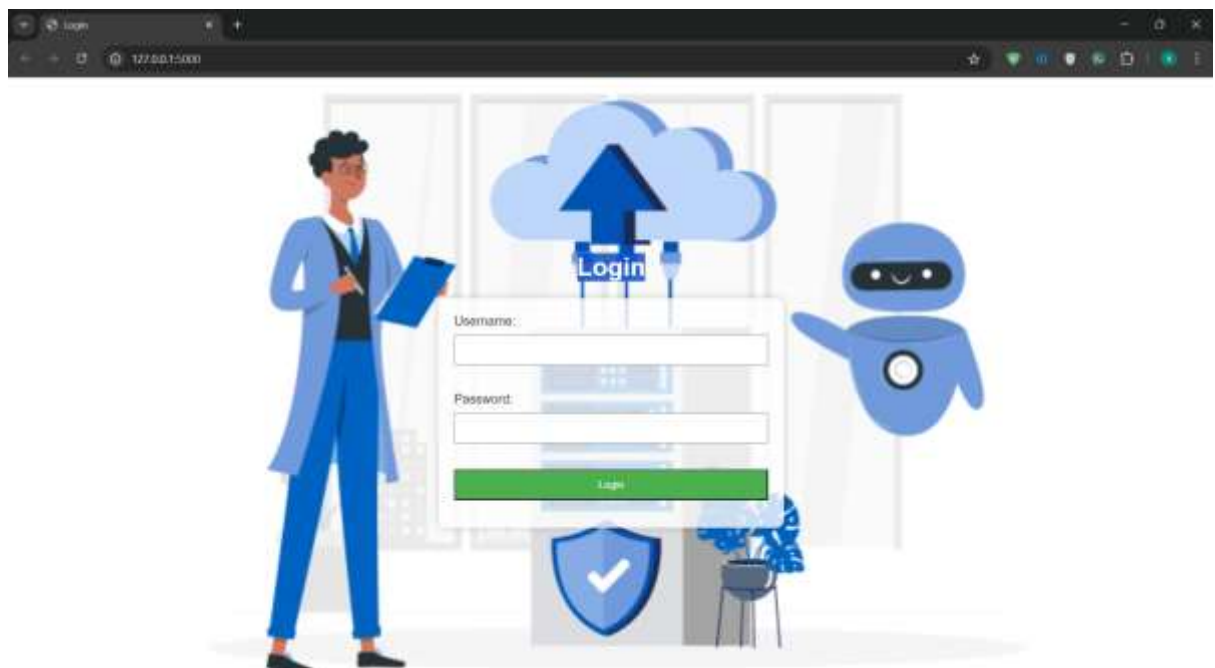
Test Case ID	Test Scenario	Test Input	Expected Result	Actual Result	Status
TC-01	Load ML Model	Load animal_disease_model.pkl	Model loads successfully	Success	Pass
TC-02	Load Label Encoders	Load label_encoders.pkl	Encoders load successfully	Success	Pass
TC-03	User Login (Valid)	Username: admin, Password: admin	Redirect to Home Page	Success	Pass
TC-04	User Login (Invalid)	Username: wrong, Password: wrong	Show error message	Success	Pass
TC-05	Submit prediction form with valid inputs	Pet: Dog, Symptoms: Red Skin, Hair Loss	Predicted Disease and Ayurvedic details shown	Success	Pass
TC-06	Submit form with missing fields	Pet not selected	Error message displayed	Success	Pass
TC-07	Prediction accuracy check	Known test symptoms	Correct disease predicted	Success	Pass
TC-08	Ayurvedic mapping check	Disease: Eye Infection	Ayurvedic and precautions displayed	Success	Pass
TC-09	Boundary testing (Age min/max)	Age: 1 and Age: 15	Prediction returned	Success	Pass
TC-10	Invalid symptom input (Unseen)	Symptom: "Unknown Symptom"	Handled gracefully or show error	Success	Pass
TC-11	Navigate Home, About, Logout	Click navigation links	Correct page loads	Success	Pass
TC-12	UI Display Test	Predicted result + Ayurvedic text	Displayed properly	Success	Pass
TC-13	Multiple users test	Simulate concurrent users	No crash or delay	Success	Pass
TC-14	Session timeout or logout	User logs out	Redirect to login page	Success	Pass
TC-15	Invalid weight input (Negative)	Weight: -10	Error displayed or handled	Success	Pass

SNAPSHOTS

Activating Environment

```
Aracoda Prompt - python 3.11.0
(base) C:\Users\ujwal\OneDrive\F #1 ノット \AI\PetSymptomsAnalysis
(base) C:\Users\ujwal\OneDrive\F #1 ノット \AI\PetSymptomsAnalysis>python app.py
C:\Users\ujwal\anaconda3\lib\site-packages\sklearn\base.py:318: UserWarning: Trying to unpickle estimator DecisionTreeClassifier from version 1.0.2 when using version 1.2.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to: https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
C:\Users\ujwal\anaconda3\lib\site-packages\sklearn\base.py:318: UserWarning: Trying to unpickle estimator RandomForestClassifier from version 1.0.2 when using version 1.2.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to: https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
C:\Users\ujwal\anaconda3\lib\site-packages\sklearn\base.py:318: UserWarning: Trying to unpickle estimator LabelEncoder from version 1.0.2 when using version 1.2.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to: https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5808
Press CTRL+C to quit
C:\Users\ujwal\anaconda3\lib\site-packages\sklearn\base.py:318: UserWarning: Trying to unpickle estimator DecisionTreeClassifier from version 1.0.2 when using version 1.2.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to: https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
C:\Users\ujwal\anaconda3\lib\site-packages\sklearn\base.py:318: UserWarning: Trying to unpickle estimator RandomForestClassifier from version 1.0.2 when using version 1.2.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to: https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
C:\Users\ujwal\anaconda3\lib\site-packages\sklearn\base.py:318: UserWarning: Trying to unpickle estimator LabelEncoder from version 1.0.2 when using version 1.2.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to: https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 131-383-274
```

Fig 7.1 Activating Environment



Login Page

Fig 7.2 Login Page

HOME PAGE

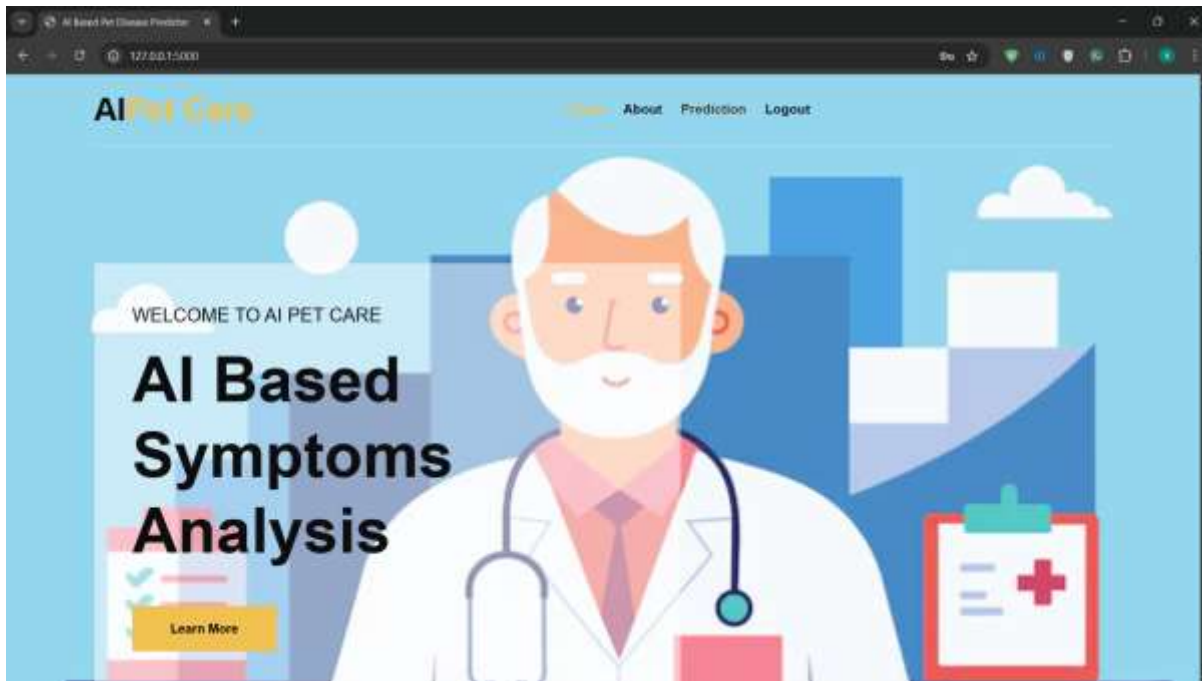


Fig 7.3 Home Page

ABOUT PAGE

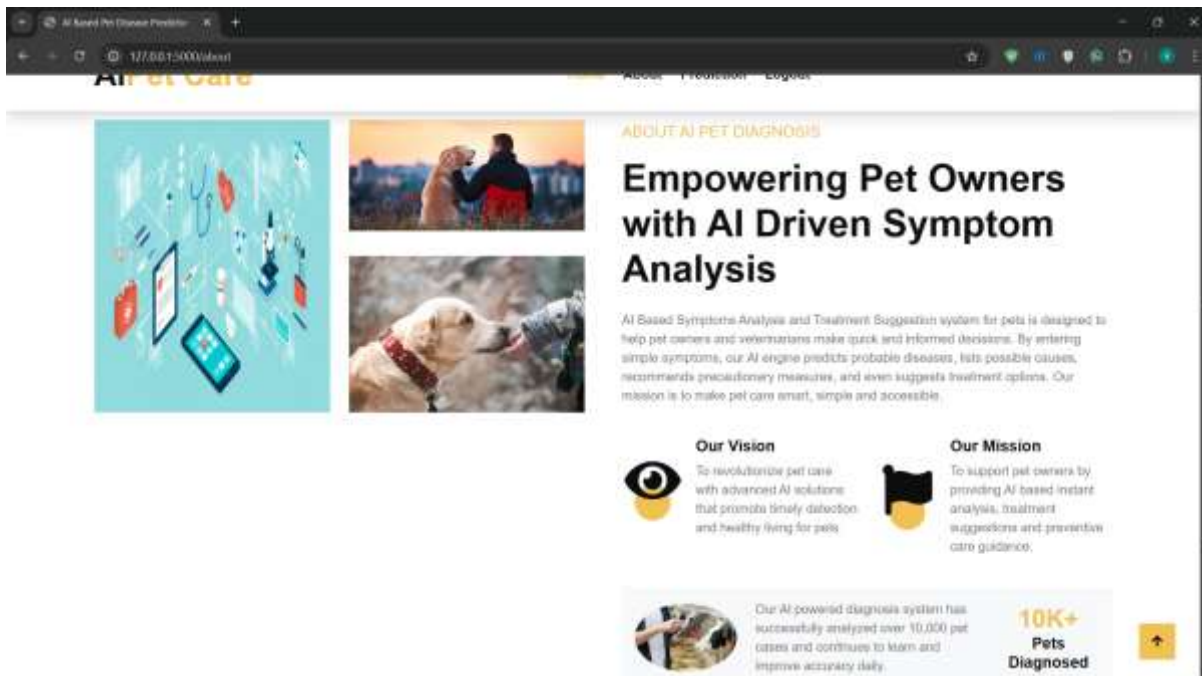
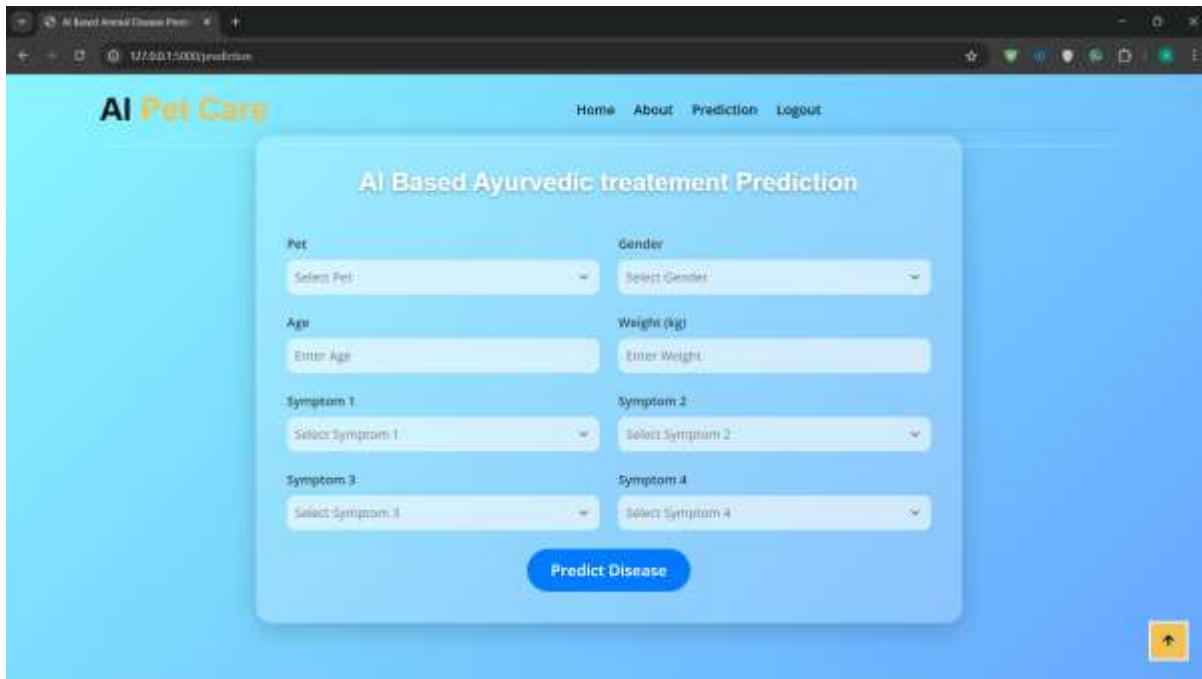


Fig 7.4 About Page

PREDICTION PAGE



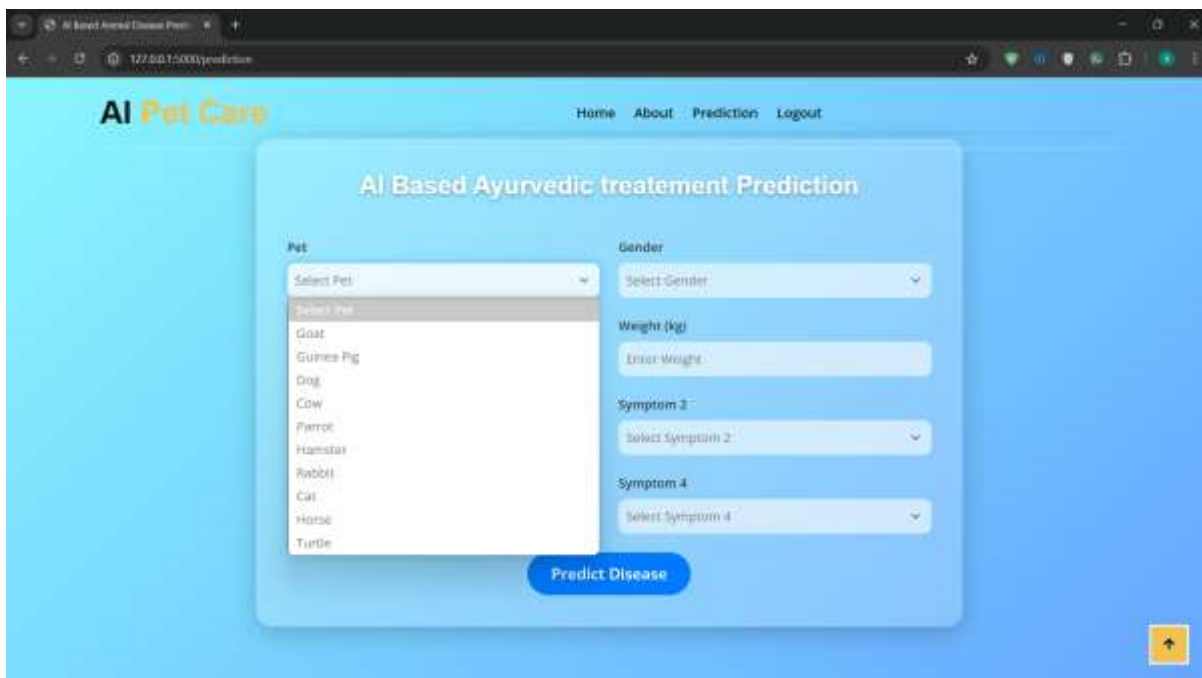
The screenshot shows the 'AI Based Ayurvedic treatment Prediction' form on the 'AI Pet Care' website. The form includes the following fields:

- Pet:** A dropdown menu with the placeholder 'Select Pet'.
- Gender:** A dropdown menu with the placeholder 'Select Gender'.
- Age:** A text input field with the placeholder 'Enter Age'.
- Weight (kg):** A text input field with the placeholder 'Enter Weight'.
- Symptom 1:** A dropdown menu with the placeholder 'Select Symptom 1'.
- Symptom 2:** A dropdown menu with the placeholder 'Select Symptom 2'.
- Symptom 3:** A dropdown menu with the placeholder 'Select Symptom 3'.
- Symptom 4:** A dropdown menu with the placeholder 'Select Symptom 4'.

A blue button labeled 'Predict Disease' is located at the bottom of the form. The website header includes 'Home', 'About', 'Prediction', and 'Logout' links.

Fig 7.5 Page

USER INPUT

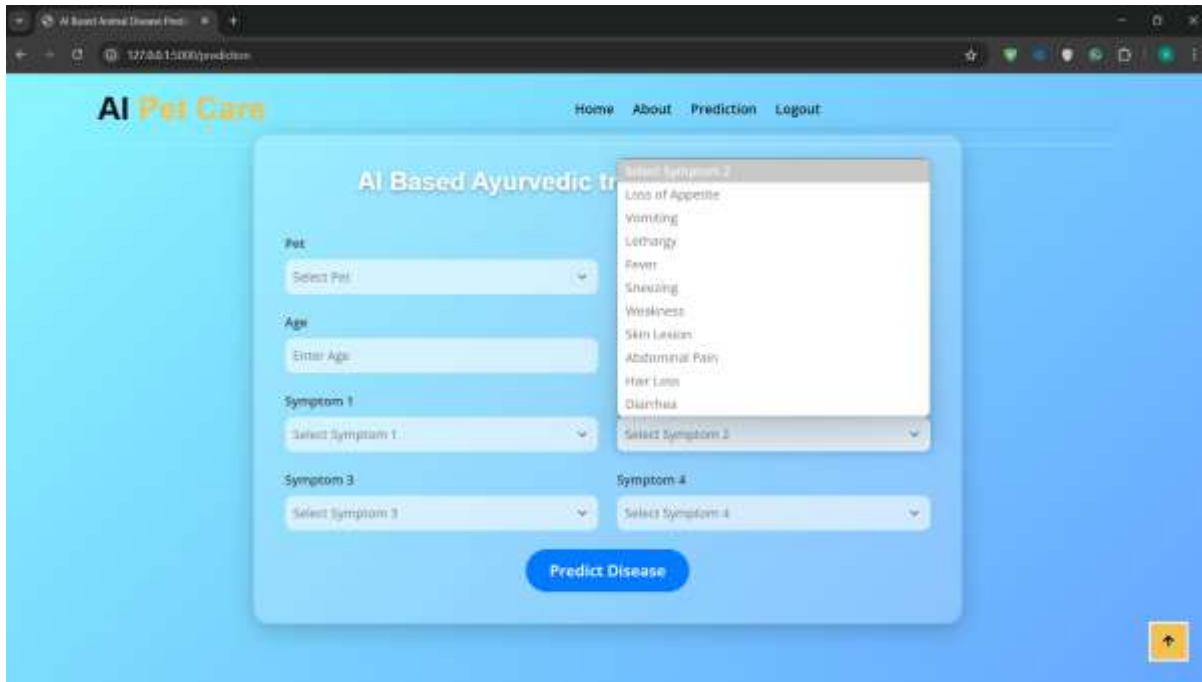


This screenshot shows the same 'AI Based Ayurvedic treatment Prediction' form, but with the 'Pet' dropdown menu open. The menu lists the following options:

- Select Pet
- Goat
- Guinea Pig
- Dog
- Cow
- Piglet
- Hamster
- Rabbit
- Cat
- Horse
- Turtle

The 'Predict Disease' button remains visible at the bottom of the form.

Fig 7.5.1 Pet Selection



The screenshot displays the 'AI Based Ayurvedic tr' interface of the 'AI Pet Care' web application. The page has a light blue background with a navigation bar at the top containing 'Home', 'About', 'Prediction', and 'Logout'. The main form includes fields for 'Pet' (a dropdown menu), 'Age' (a text input), and four symptom selection dropdowns labeled 'Symptom 1', 'Symptom 2', 'Symptom 3', and 'Symptom 4'. A 'Predict Disease' button is located at the bottom of the form. A dropdown menu for 'Symptom 2' is open, showing a list of symptoms: 'Loss of Appetite', 'Vomiting', 'Lethargy', 'Fever', 'Shivering', 'Weakness', 'Skin Lesion', 'Abdominal Pain', 'Hair Loss', and 'Diarrhea'.

Fig 7.5.2 Symptom Selection

PREDICTION



The screenshot displays the 'Predicted Disease: Kidney Disease' results page. The page has a green background. At the top, there is a header 'Predicted Disease: Kidney Disease' with a paw print icon. Below this, there are two main sections: 'Causes' and 'Ayurvedic Treatment Suggestions'. The 'Causes' section includes a paragraph explaining that kidney disease in pets can result from various factors like age-related degeneration, genetic factors, chronic infections, or toxic exposure. The 'Ayurvedic Treatment Suggestions' section lists several recommendations: 'Punarnava (Boerhaavia diffusa)' for reducing swelling and rejuvenating kidneys, 'Gokshura (Tribulus terrestris)' for supporting urinary tract and reducing inflammation, 'Dietary Correction' with easily digestible protein-modified diets, and 'Massage' with warm sesame oil for Vata pacification.

Fig 7.6.1 Prediction Page

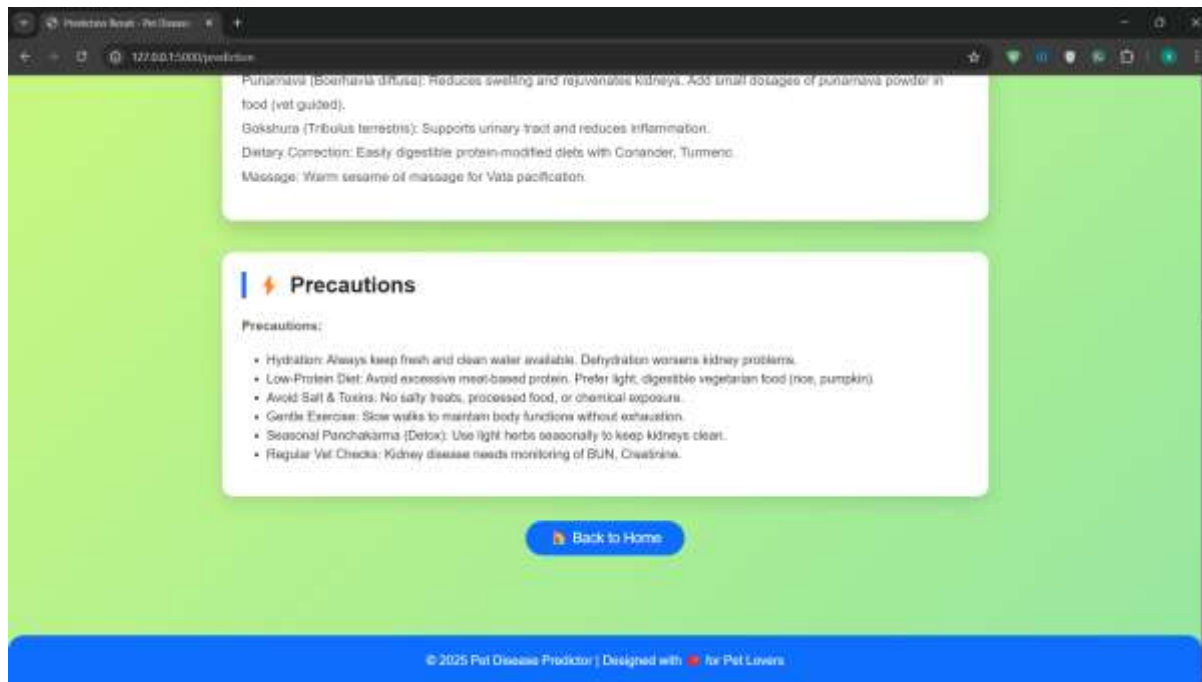


Fig 7.6.2 Prediction Page

CONCLUSION

The rise in pet ownership globally has driven the need for innovative healthcare solutions that cater to pets' well-being. Traditional veterinary practices, though essential, often face challenges such as limited availability, delayed diagnosis, and high costs. In this context, the development of the **AI-Based Symptom Analysis and Ayurvedic Treatment Suggestion System for Pets** has proven to be a highly effective solution that bridges this gap using advanced machine learning technologies and ancient Ayurvedic knowledge.

Through this project, a comprehensive system was successfully implemented that predicts pet diseases based on user-provided data such as pet type, age, gender, weight, and symptoms. The machine learning model was built using Random Forest, which outperformed other algorithms like K-Nearest Neighbors, Naive Bayes, and Decision Trees in terms of accuracy and robustness. The model achieved an impressive accuracy of approximately 98%, demonstrating its ability to make highly reliable predictions.

The system does not merely stop at prediction. It has been enhanced with a carefully curated domain knowledge base that provides users with Ayurvedic treatment suggestions and necessary precautions for each disease. This makes the system not only diagnostic but also advisory and educational, promoting holistic pet care.

Furthermore, the Flask-based web application successfully delivers a user-friendly interface through which pet owners can input symptoms and immediately receive informative results. The seamless integration of backend prediction models and frontend user interfaces ensures that even users with minimal technical knowledge can benefit from this system.

In conclusion, the project has achieved its primary goal of creating an AI-powered, easily accessible, and advisory system for pet health management. The model performs reliably, and the integration of Ayurvedic suggestions makes it stand out as a unique solution in the domain of AI-assisted pet healthcare.

FUTURE ENHANCEMENT

Although the current system is highly functional and achieves its core objectives, there remains significant scope for further development and enhancement. The present version, while accurate and advisory, works primarily in a rule-based and symptom-input format. Future upgrades can transform this into an even smarter, more interactive, and highly integrated platform. One of the foremost improvements would be the incorporation of **image and voice-based symptom detection**. By integrating computer vision and natural language processing (NLP) techniques, the system can automatically analyze images (e.g., skin conditions, eye infections) and user-reported symptoms via speech, making the diagnosis process even more intuitive and accessible.

Furthermore, the Ayurvedic module can be made more dynamic and personalized. Integrating expert-reviewed datasets or connecting to Ayurvedic practitioners' knowledge bases can ensure that recommendations remain up-to-date and specific to the pet's type, age, and severity of disease. AI can also assist in **customizing Ayurvedic treatment plans** based on user feedback and recovery tracking.

Additionally, a cloud-based version of the platform can be developed to facilitate **multi-user access, veterinary integration, and community-based data sharing**. This will allow veterinary clinics, NGOs, and pet health researchers to contribute and access anonymized pet health data for large-scale analysis, enabling smarter models in the future.

REFERENCES

- [1] B. G. M. Habal, P. E. S. Tiong, J. R. Pasatiempo, M. J. Balen, M. R. Amarga and L. Juco, "Dog Skin Disease Recognition Using Image Segmentation and GPU Enhanced Convolutional Neural Network," 2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM), Manila, Philippines, 2021, pp. 1-5, doi: 10.1109/HNICEM54116.2021.9731885.
- [2] Kim, S.-C.; Kim, S. Development of a Dog Health Score Using an Artificial Intelligence Disease Prediction Algorithm Based on Multifaceted Data. *Animals* 2024, *14*, 256. <https://doi.org/10.3390/ani14020256>
- [3] Ayyaswamy, Kathirvel. (2024). AI-Based Healthcare Systems for Pets and Birds. *Biomedical Journal of Scientific & Technical Research*. 58. 10.26717/BJSTR.2024.58.009202.
- [4] Jaypriya Chilampande ANIMAL DISEASE PREDICTION link: https://www.irjmets.com/uploadedfiles/paper//issue_11_november_2023/46372/final/fin_irjmets1700309630.pdf
- [5] A. Nadar, A. Sane, G. Muga, E. Masih and S. Rukhande, "Animal Healthcare and Farm Animal Disease Prediction Using Machine Learning," 2023 5th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), Navi Mumbai, India, 2023, pp. 1-6, doi: 10.1109/ICNTE56631.2023.10146635.
- [6] Nadar, Augustin & Sane, Adishree & Muga, Glenn & Masih, Easter & Rukhande, Smita. (2023). Animal Healthcare and Farm Animal Disease Prediction Using Machine Learning. 1-6. 10.1109/ICNTE56631.2023.10146635.
- [7] Wenqiang Guo, Chenrui Lv, Meng Guo, Qiwei Zhao, Xinyi Yin, Li Zhang, Innovative applications of artificial intelligence in zoonotic disease management, *Science in One Health*, Volume 2, 2023, 100045, ISSN 2949-7043, <https://doi.org/10.1016/j.soh.2023.100045>.