International Journal of Trend in Scientific Research and Development (IJTSRD)
Special Issue on Smart Innovations in Computer Science and Applications
Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

AI-Powered Smart Animal Health Monitoring and Diagnosis System

Ritika Ravindra Thakur
Department of Science and Technology G.H.Raisoni Skill Tech University, Nagpur, India

Abstract

Most pet owners genuinely care about their animals, but in reality, very few maintain proper digital health records. Vaccination dates, weight history, and previous medical issues are often stored in paper form or simply forgotten. Small signs like reduced appetite or unusual tiredness are sometimes ignored until the condition becomes serious.

To address this issue, a desktop-based Smart Animal Health Monitoring and Diagnosis System was developed. The application is built using Python, with PyQt5 for the graphical interface and SQLite for local database storage. It allows users to store pet details, track growth using graphical visualization, set reminders for vaccination or medication, analyze symptoms using a rule-based scoring system, and view nearby veterinary locations.

The system does not use complex medical datasets but instead applies a simple and interpretable scoring logic to classify health conditions into Low, Moderate, or High risk categories. The main objective of this project is to assist pet owners in early health awareness and organized record management rather than replacing professional veterinary advice.

Keywords: Smart Animal Health System, Decision Support System, Rule-Based Scoring Algorithm, Graphical User Interface (GUI), Local Database Management, Health Risk Classification, Modular Application Architecture, Preventive Healthcare Technology, Human–Computer Interaction, AI-Assisted Pet Care

1. Introduction

Pet healthcare management at the household level is often unstructured and inconsistent. While veterinary clinics maintain proper medical records, most pet owners do not keep a well-organized digital record of their pet's health information. Important details such as vaccination dates, medical history, weight records, and previous symptoms are usually written on paper prescriptions, stored in scattered files, or simply remembered. Over time, this information becomes difficult to track, especially during emergencies or vet visits when accurate history is required.

Another practical issue is that early symptoms in pets are frequently ignored. Since animals cannot directly communicate their discomfort, small signs like low appetite, tiredness, or slow growth may not be taken seriously at the beginning. Many owners assume these changes are temporary and delay medical consultation, which can sometimes lead to more serious health complications. This shows the need for a simple system that helps in early symptom awareness and structured health monitoring.

In recent years, AI-based systems have been widely used in human healthcare for symptom analysis and decision support. However, similar simple and offline applications focused on pet health monitoring are still limited. Most existing solutions are either complex, internet-dependent, or

designed only for a single function such as reminders or online consultation. There is a lack of beginner-friendly desktop applications that combine multiple useful features like record storage, health tracking, and risk indication in one place.

To address this gap, this project proposes a practical desktop-based system that helps pet owners manage their pet's health in a more organized and accessible way. The application integrates multiple features including pet details management, symptom-based health risk analysis, growth tracking with graphical visualization, reminder scheduling, AI-based chat assistance, and veterinary location viewing. Instead of relying on complex medical datasets, the system uses a rule-based scoring approach to estimate health risk levels such as Low, Moderate, and High, making it easy to understand and use.

The application is developed using Python in Visual Studio Code, and the graphical user interface is designed manually using PyQt5 layouts and widgets without using PyQt Designer. This manual GUI design helped in customizing the interface according to the project requirements and improving understanding of GUI structure. SQLite database is used for storing all pet records locally, which ensures fast performance, secure storage, and full offline functionality for core features (except vet locator).

During development, the focus was kept on simplicity, usability, and modular design so that each feature like pet records, reminders, growth tracker, and risk analyzer works smoothly within a single dashboard. The main goal of this project is not to replace professional veterinary diagnosis but to provide an easy and organized platform for preventive pet healthcare monitoring. By combining multiple essential features into one lightweight desktop application, the system supports better record management, early symptom awareness, and responsible pet care in a practical and user-friendly manner.

1. Literature Review

In recent years, Artificial Intelligence has been increasingly used in healthcare-related applications for early symptom analysis and decision support. Many systems developed for human healthcare use machine learning models to predict diseases based on user input symptoms, medical history, and behavioral patterns. These systems help in providing preliminary guidance before professional consultation.

Some studies show that AI-based diagnosis tools improve early detection and reduce the chances of delayed treatment. Researchers have used techniques like decision trees, rule-based systems, and supervised learning for disease prediction. These approaches are considered useful because they are easy to interpret and can give quick results.

However, most existing applications mainly focus on human health and are often web-based or require internet connectivity. There are very few simple desktop applications designed specifically for pet health monitoring, especially those that work in offline mode and include additional features like reminders, growth tracking, and record management.

Another limitation observed in previous systems is that they either focus only on diagnosis or only on record storage, but do not combine multiple useful modules in one platform. For pet owners, managing scattered information like vaccination dates, symptoms, and medical history becomes difficult over time.

Therefore, this project attempts to bridge this gap by developing a simple offline desktop application that combines symptom analysis, pet record management, reminders, and growth tracking in a single system. The aim is not to create a highly complex AI model, but to design a practical and user-friendly tool that can assist pet owners in daily health monitoring.

2. Problem Statement

In many households, pet healthcare management is not properly organized. Most pet owners rely only on memory or physical documents to track their pet's medical history, vaccination dates, and health conditions. Over time, these records get misplaced or forgotten, which creates problems during emergencies or vet visits.

Another major issue is that pet owners often fail to recognize early symptoms of health problems. Small signs like loss of appetite, low energy, or unusual behavior are sometimes ignored until the condition becomes serious. This delay in identification can negatively affect the pet's health.

Additionally, there is no single offline platform where users can:

- Store pet details
- Track growth data
- Set medical reminders
- Check possible health risks based on symptoms

Many available applications are either too complex, require internet access, or are designed mainly for professional veterinary use rather than regular pet owners.

Hence, there is a need for a simple, offline, and user-friendly desktop system that helps in organizing pet health records and providing basic symptom-based health risk awareness in an accessible manner.

2. Objective of the Project

The main objectives of this project are:

- To develop a desktop-based pet health management system
- To store and manage pet details in a structured digital format
- To provide symptom-based health risk analysis using a scoring method
- To implement reminder functionality for vaccination and medication
- To track pet growth using graphical representation
- To design a simple and attractive user interface using PyQt5
- To ensure the system works offline using a local SQLite database

3. Proposed System

The proposed system is an AI-powered smart animal health diagnosis desktop application developed using Python. The application is designed to be fully offline (except for veterinary location viewing) and easy to use for normal pet owners.

Instead of using heavy machine learning models, the system uses a rule-based scoring approach to analyze symptoms entered by the user. Based on the selected symptoms, the system calculates a score and classifies the health risk level into Low, Moderate, or High.

The application is divided into multiple modules to keep the structure organized:

- Pet Details Module
- Health Diagnosis Module
- Growth Tracker Module
- Reminder Module
- AI Chat Assistance Module

- Veterinary Locator Module

The user interface was designed manually using PyQt5 layouts in Visual Studio Code, without using PyQt Designer. This allowed better control over customization and layout design.

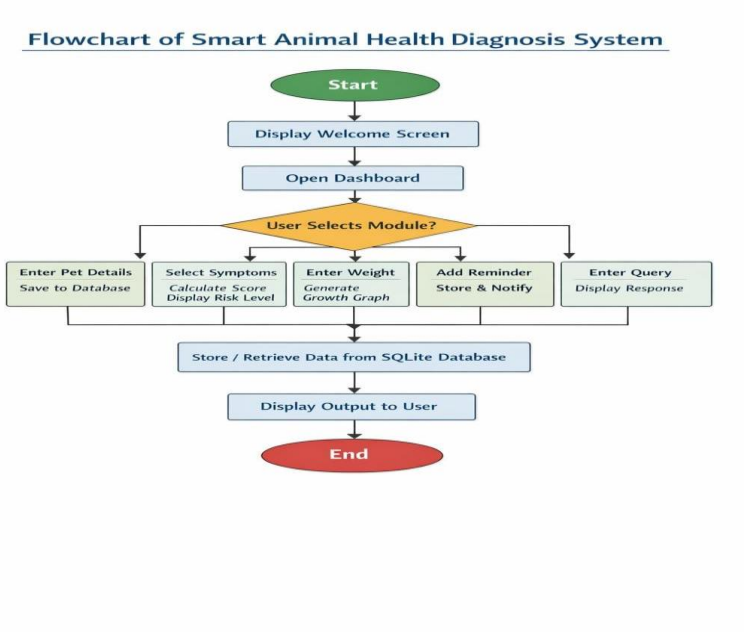


Fig 1. Flowchart of the proposed system

3.1 Data Flow Diagram (Level 0)

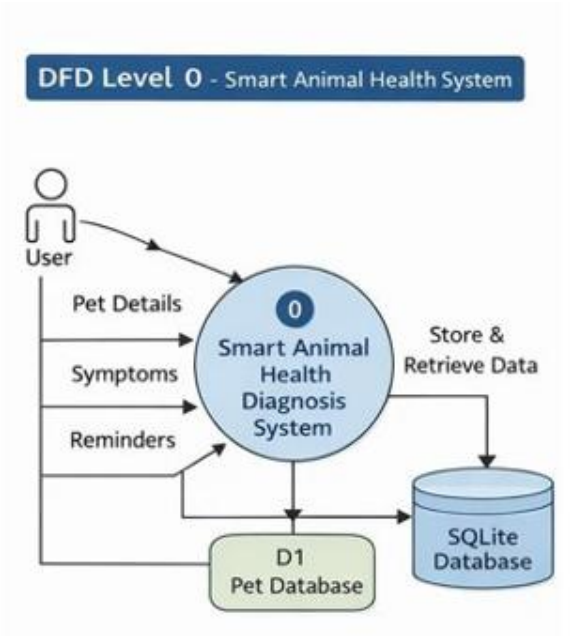


Fig 2. Data Flow Diagram – level 0

3.1 Data Flow Diagram (Level 1)

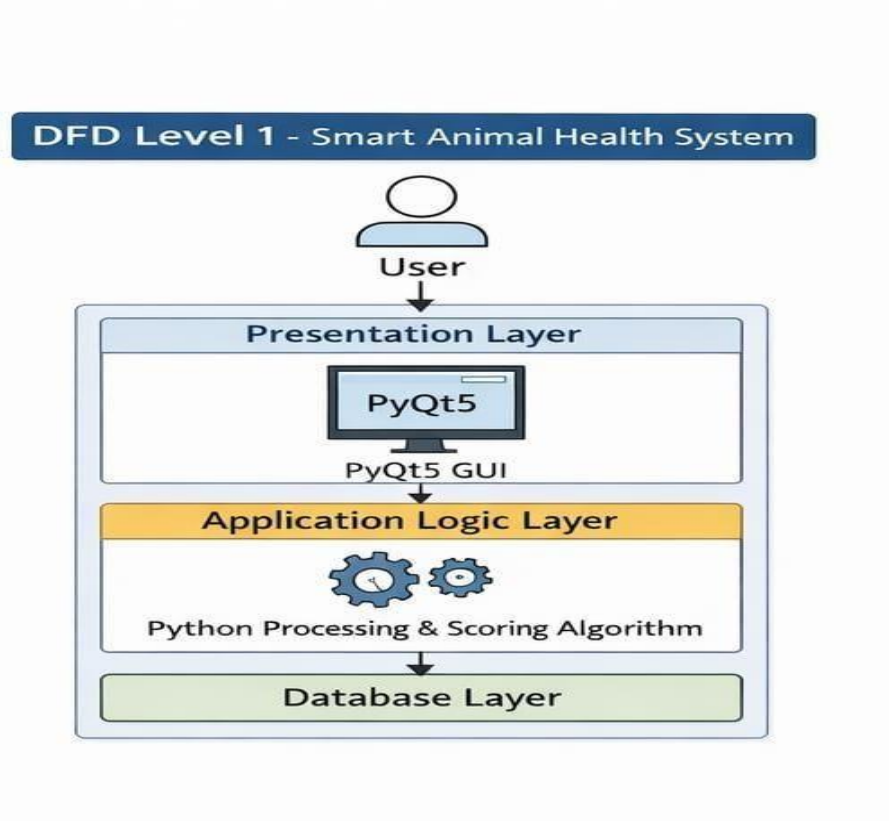


Fig 3. Data Flow Diagram – level 1

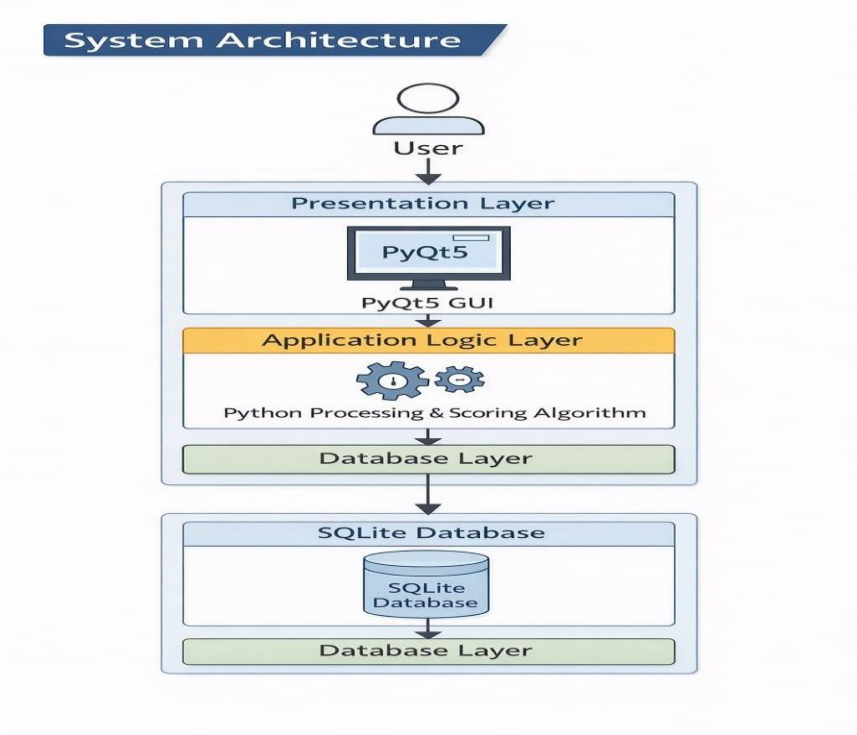
The Level 1 Data Flow Diagram provides a more detailed view of how data flows inside the system modules. Unlike Level 0 DFD, which shows only the overall system interaction, Level 1 explains internal processing such as pet data storage, symptom analysis, reminder scheduling, and growth tracking.

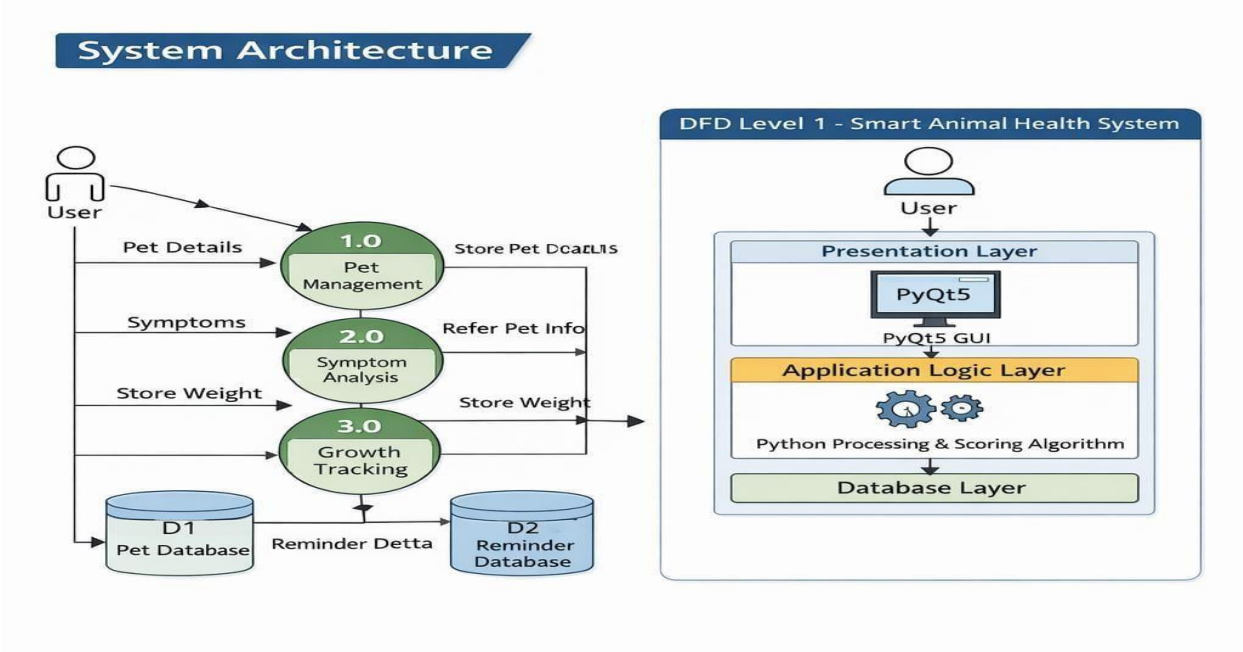
When the user enters pet details, the data is stored in the SQLite database. Similarly, symptom inputs are processed by the health risk analyzer module, which calculates a score and generates a risk level. The growth tracker stores weight data and displays graphical output, while the reminder module manages scheduled notifications. This layered data flow ensures organized processing and efficient data management within the application.

3.3 System Architecture

The system architecture of the proposed application follows a modular desktop-based structure. The user interacts with the graphical interface developed using PyQt5, which acts as the front-end layer of the system. All user inputs such as pet details, symptoms, and growth data are processed through individual modules connected to the main dashboard.

The backend consists of an SQLite database that stores all records locally, ensuring fast and secure offline access. The health risk analyzer processes symptom inputs using rule-based logic, while other modules like reminders, chat assistant, and growth tracker function independently but are integrated through the central dashboard. This architecture improves scalability, maintainability, and overall system performance.





The architecture mainly consists of:

- User Interface Layer (PyQt5)
- Application Logic Layer (Python)
- Database Layer (SQLite)

This layered design makes the system easy to maintain and update in the future.

5. Methodology

The development of this project followed a structured step-by-step approach to ensure clarity, proper functionality, and modular organization. The main goal was to create a simple yet practical desktop application that could assist pet owners in managing health-related information efficiently.

Initially, requirement analysis was carried out to identify the core functionalities needed in a pet healthcare management system. Based on common real-world problems faced by pet owners, features such as pet record management, symptom-based health analysis, reminders, and growth tracking were finalized.

After identifying requirements, system design and module planning were performed. The application was divided into separate modules such as Pet Details, Health Risk Analyzer, Growth Tracker, Reminder System, Chat Assistant, and Veterinary Locator. This modular design approach helped in maintaining clean code structure and easier debugging during development.

The database design phase involved creating structured tables using SQLite to store pet records, reminder data, and growth information. SQLite was chosen because it is lightweight, fast, and suitable for offline desktop applications.

The graphical user interface was developed using PyQt5 in Visual Studio Code through manual coding instead of using PyQt Designer. Designing the layout manually provided better understanding of widget placement, signal-slot connections, and layout management.

For the health diagnosis feature, a rule-based scoring algorithm was implemented. The working of the algorithm is as follows:

1. The user selects symptoms from the interface.
2. Each symptom is assigned a predefined score based on severity.
3. The system calculates the total score.
4. The total score is compared with predefined threshold values.
5. Based on the result, the system displays the risk level as Low, Moderate, or High.

This approach was selected instead of complex machine learning models to maintain transparency, faster execution, and offline compatibility. Since the objective of the project is early awareness rather than medical prediction, the rule-based system is sufficient and practical.

Finally, testing was conducted by entering different symptom combinations and verifying whether the generated risk level matched expected behavior. All modules were tested individually and then integrated to ensure smooth functioning of the entire application.

In addition to functional testing, special attention was given to usability and performance. The application was tested on a standard desktop environment to ensure smooth navigation between modules and quick response time during data processing. Error handling mechanisms were also implemented to prevent crashes due to invalid inputs. This ensured that the system remains stable, user-friendly, and reliable for everyday use by pet owners.

1. Algorithm Table

Step 1: Start the application
Step 2: Input selected symptoms
Step 3: Assign score to each symptom
Step 4: Calculate total score
Step 5: If score< X → Low Risk
Step 6: If score is between X and Y → Moderate Risk
Step 7: If score > Y → High Risk
Step 8: Display result
Step 9: Stop

6. Implementation Details

The implementation of the Smart Pet Health Diagnosis Application was carried out using Python as the core programming language. The application was developed as a desktop-based system to ensure offline usability and simplicity for pet owners.

PyQt5 was used for designing the graphical user interface. All layouts and widgets were created manually using code in Visual Studio Code instead of using PyQt Designer. This helped in understanding layout structure, signal-slot connections, and event handling more clearly.

SQLite was used as the backend database for storing pet details, reminder records, and growth-related data. Since SQLite is lightweight and does not require server configuration, it was suitable for an offline desktop application.

The project was organized into separate modules to maintain clean structure and easy debugging. Each feature such as Pet Details, Health Risk Analyzer, Growth Tracker, Reminder System, Chat Assistant, and Veterinary Locator was implemented in separate files and then integrated into the main dashboard.

The health risk analyzer module uses a predefined scoring system. Based on the symptoms selected by the user, a total score is calculated and compared with threshold values to determine the risk level.

Proper error handling and input validation were added to avoid crashes and incorrect data entry. The application was tested multiple times during development to ensure stable performance.

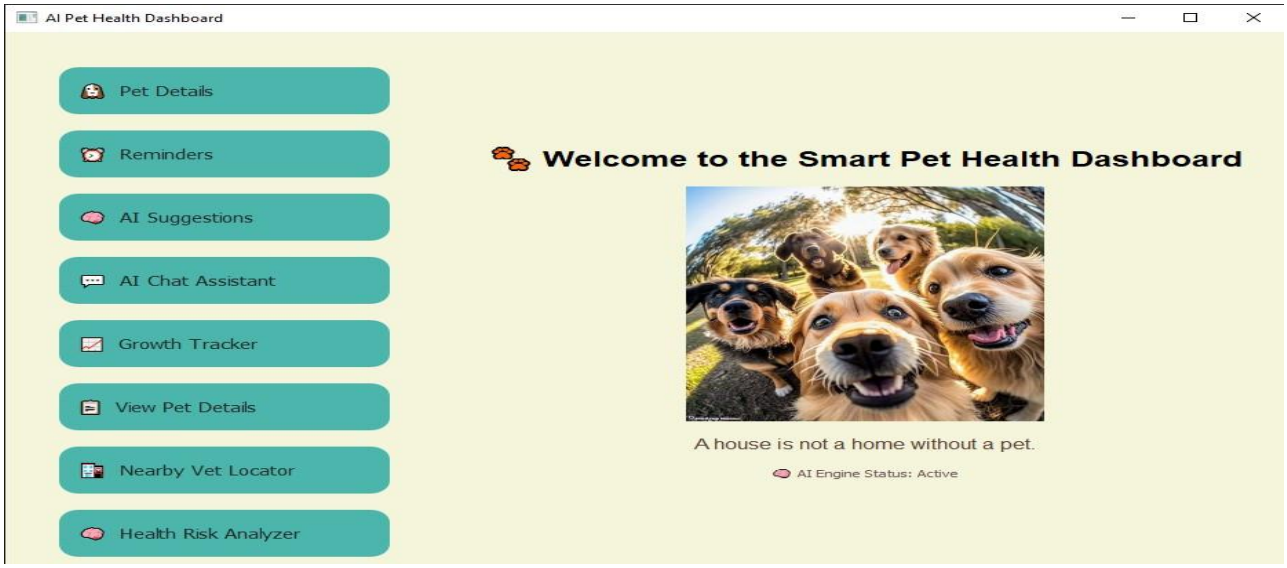
7. Testing and Results

The developed application was tested to ensure that all modules function correctly and provide expected outputs. Testing was carried out by providing different input combinations and observing whether the system behavior matched the designed logic.

Each module was first tested individually and then tested after full integration to verify smooth interaction between components. The application was executed multiple times to check stability and performance in offline mode.

1. Dashboard Module

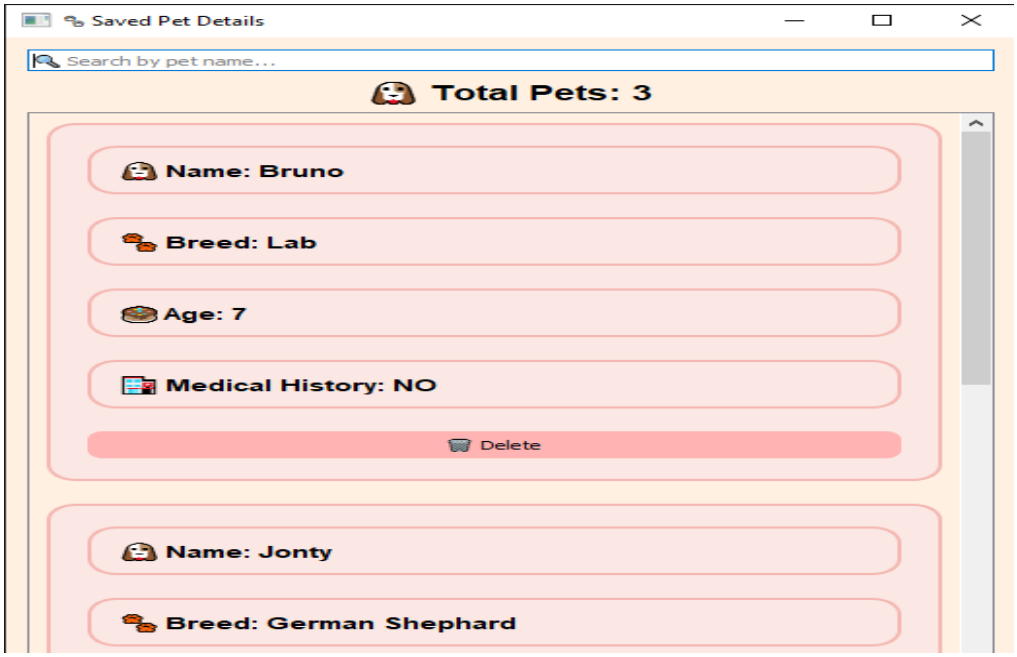
The Dashboard acts as the main control panel of the application. It provides easy navigation to all major modules such as Pet Details, Health Risk Analyzer, Growth Tracker, Reminder System, Chat Assistant, and Veterinary Locator. The purpose of the dashboard is to give users a clear and organized starting point after entering the application. All buttons are arranged in a simple layout so that even non-technical users can access different features without confusion. The dashboard improves usability by keeping all important functionalities in one place and ensuring smooth movement between modules.



2. Pet Details Module

The Pet Details module was tested by adding new pet records, editing existing data, and deleting entries. The system successfully stored all information in the SQLite database and retrieved it correctly when requested.

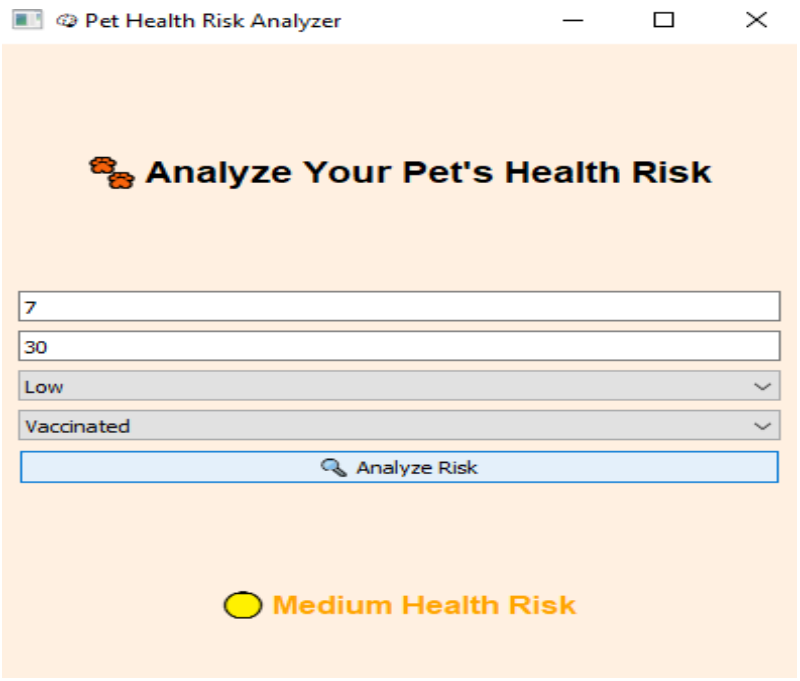
Data consistency was verified by reopening the application and checking whether previously stored records were still available. The records were preserved properly, confirming correct database integration.



3. Health Risk Analyzer Module

The Health Risk Analyzer module was tested using various combinations of symptoms to verify the accuracy of the scoring logic. When mild symptoms such as slight weakness or reduced appetite were selected, the total calculated score remained low and the system displayed a "Low Risk" result.

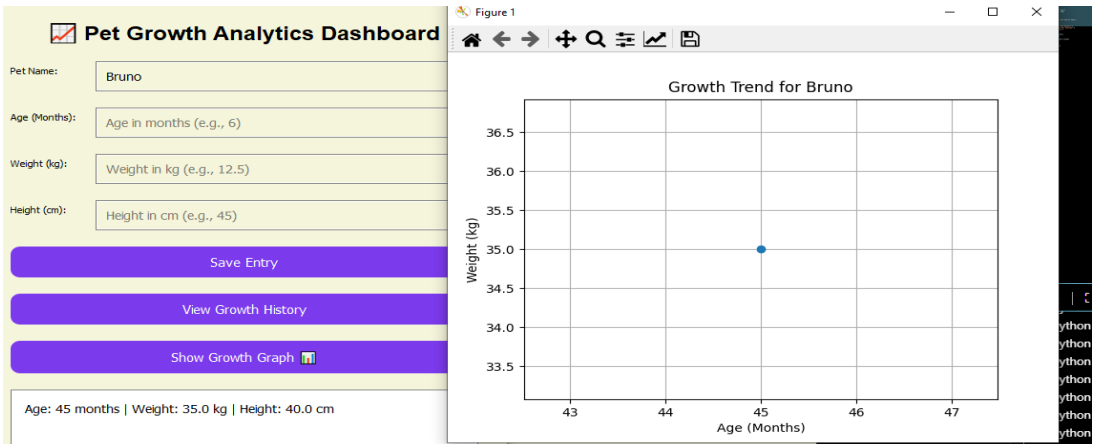
When multiple severe symptoms were selected together, the total score exceeded the threshold value and the system correctly displayed a "High Risk" result. This confirmed that the rule-based scoring logic works as intended.



4. Growth Tracker Module

The Growth Tracker module was tested by entering multiple weight values on different dates. The system stored this data and generated a graphical representation of growth over time.

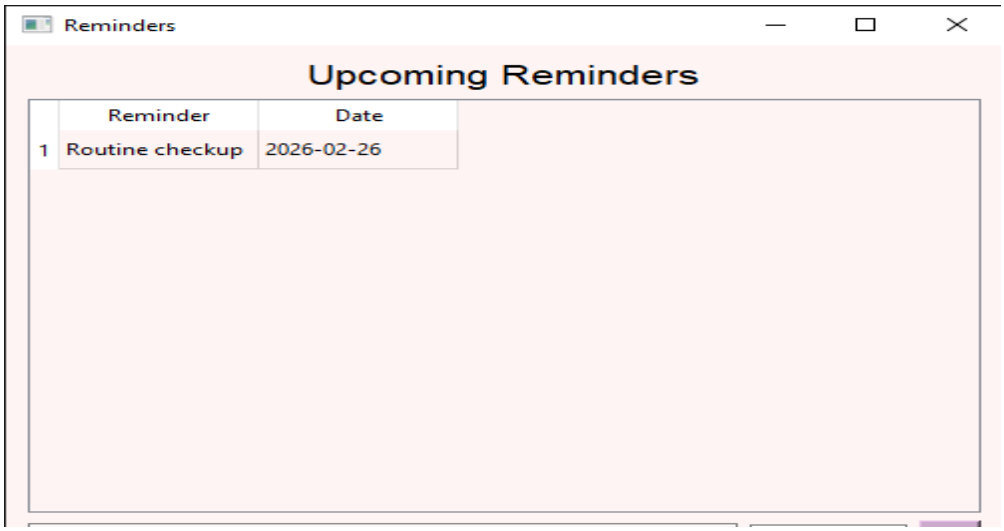
The graph updated correctly whenever new data was entered. This verified that the data processing and graphical visualization logic were function



5. Reminder Module

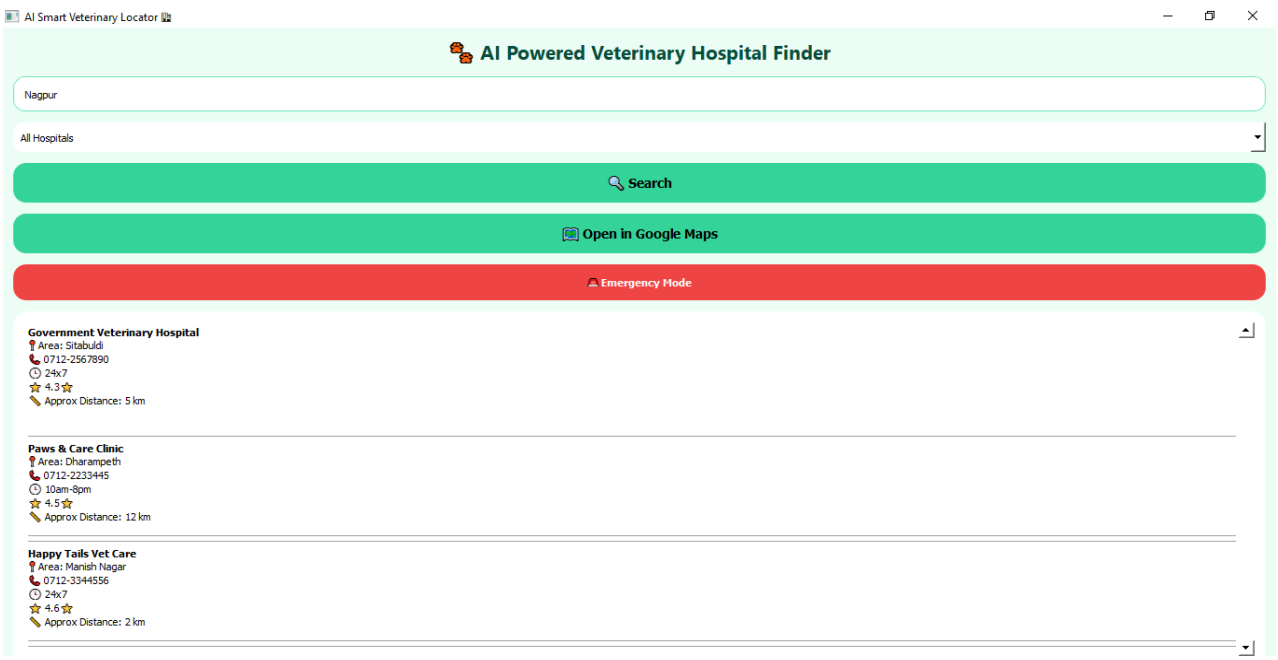
The Reminder module was tested by setting reminders for vaccination and medication. After saving reminder data, the system stored it in the database and displayed confirmation.

The module was checked to ensure that reminder entries could be viewed and managed correctly without errors.



6. Veterinary Locator Module

The Veterinary Locator module allows users to find nearby veterinary clinics when professional help is needed. This feature requires internet access to display location information. It helps pet owners quickly identify veterinary services, especially in cases where the health risk level is moderate or high. This module increases the practical value of the application by supporting real-world assistance.



8. Conclusion

This project successfully developed a simple and practical desktop application for pet health management. The system helps pet owners maintain pet records, track growth, set reminders, and perform basic health risk analysis using a rule-based scoring method. All core functionalities were implemented with the aim of making the application easy to understand and convenient to use.

The application achieves its objective of providing early health awareness without relying on complex machine learning models. Instead of focusing on prediction accuracy at an advanced level, the system prioritizes transparency, simplicity, and offline usability. This makes it suitable for everyday users who may not have technical knowledge.

The modular structure of the application ensures better organization and easier maintenance. Each feature was developed separately and later integrated, which helped in improving clarity and debugging during development. The use of Python, PyQt5, and SQLite proved to be effective for building a stable and lightweight desktop application.

Overall, the project demonstrates how a structured approach and simple logic can be used to create a useful and user-friendly system that addresses real-world problems faced by pet owners.

9. Future Work

The application can be further improved by integrating real-time veterinary consultation features. A cloud-based database can also be added to allow multi-device access.

In the future, machine learning models may be implemented to improve health risk prediction accuracy. Push notifications and mobile application versions can also be developed to increase accessibility.

Additional features such as vaccination tracking and automated diet recommendations can further enhance the usefulness of the system.

References

1. A. Holzinger, “Interactive machine learning for health informatics: When do we need the human-in-the-loop?,” Brain Informatics, vol. 3, no. 2, pp. 119–131, 2016.

2. R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner’s Approach, 8th ed. New York, NY, USA: McGraw-Hill Education, 2015.

3. I. Sommerville, Software Engineering, 10th ed. Boston, MA, USA: Pearson, 2016.

4. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Boston, MA, USA: Addison-Wesley, 1994.

5. W. Stallings, Cryptography and Network Security: Principles and Practice, 7th ed. Boston, MA, USA: Pearson, 2017.

6. M. Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd ed. Boston, MA, USA: Addison-Wesley, 2004.

7. J. D. Kelleher, B. Mac Namee, and A. D’Arcy, Fundamentals of Machine Learning for Predictive Data Analytics. Cambridge, MA, USA: MIT Press, 2015.

8. T. Mitchell, Machine Learning. New York, NY, USA: McGraw-Hill, 1997.

9. D. Flanagan, Python in a Nutshell, 3rd ed. Sebastopol, CA, USA: O’Reilly Media, 2013.

10. M. Summerfield, Rapid GUI Programming with Python and Qt. Upper Saddle River, NJ, USA: Prentice Hall, 2007.

11. A. Chaube, "ACO-Enhanced Siamese Networks for Robust Feature Matching in Copy-Move Image Forgery Detection," 2024 International Conference on Artificial Intelligence and Quantum

Computation-Based Sensor Application (ICAIQSA), Nagpur, India, 2024, pp. 1-6, doi: 10.1109/ICAIQSA64000.2024.10882433.

12. Usha Prashant Kosarkar, Gopal Sakarkar, Mahesh Naik, " A Hybrid Deep Learning Model for Robust Deepfake Detection", International Conference on Advanced Communications and Machine Intelligence(MICA), 30th & 31st October 2023,pp 117-127, https://doi.org/10.1007/978-981-97-6222-4_9