**Virtual Pet Monitoring Application**

# Overview

The Virtual Pet Monitoring Application is a standalone Python-based software project designed to

simulate a virtual pet that users can monitor and interact with. The application utilizes the Tkinter

library for the graphical user interface (GUI) and SQLite for local storage of pet data.

# Features

1. Health Monitoring: Users can monitor the virtual pet's health, which is initially set to 100.

The health value is updated based on user interactions.

2. Happiness Tracking: The application includes a happiness metric for the virtual pet. Users can

engage with the pet to increase its happiness level, enhancing the overall well-being of the

virtual companion.

3. Feed the Pet: Users can feed the virtual pet, which increases its health by 10 points. This

action simulates caring for the pet's nutritional needs.

4. Play with the Pet: Interaction through play boosts the pet's happiness by 10 points. This

feature encourages users to engage with the virtual pet for its mental well-being.

5. Local Storage: The pet's data, including health, happiness, and the timestamp of the last

activity, is stored locally using SQLite. This enables the pet's state to persist even when

the application is closed and reopened.

# Implementation Details

* The project is developed in Python, chosen for its simplicity and versatility.
* Tkinter is used for creating the graphical user interface, providing a straightforward

and native look for the application.

* SQLite is employed as a lightweight database for storing the virtual pet's data. The database file

(virtual\_pet\_data.db) is created in the same directory as the Python script.

* The virtual pet class (VirtualPet) encapsulates the pet's attributes and behavior, including

methods for updating health, happiness, and saving/loading data to/from the database.

* The application's main class (VirtualPetApp) sets up the GUI, handles user interactions,

and updates the display based on the virtual pet's state.

# How to Run the Application

1. Prerequisites:

* Ensure that you have Python installed on your system.

2. Install Required Libraries:

* Open a terminal and run the following commands:
* bash
* pip install tk
* pip install sqlite3

3. Run the Application:

* Save the provided Python script (virtual\_pet.py) on your local machine.
* Open a terminal in the script's directory.
* Run the script using the command:
* bash
* python virtual\_pet.py

4. Interact with the Virtual Pet:

* The GUI will display the virtual pet's health and happiness levels.
* Use the "Feed" and "Play" buttons to interact with the virtual pet and observe the changes in its state.

#Program

import tkinter as tk

import sqlite3

import time

conn = sqlite3.connect('virtual\_pet\_data.db')

cursor = conn.cursor()

cursor.execute('''

CREATE TABLE IF NOT EXISTS pet\_data (

id INTEGER PRIMARY KEY AUTOINCREMENT,

health INTEGER,

happiness INTEGER,

last\_activity\_time TEXT

)

''')

conn.commit()

class VirtualPet:

def \_init\_(self):

self.health = 100

self.happiness = 100

self.last\_activity\_time = time.strftime('%Y-%m-%d %H:%M:%S')

def update\_health(self, value):

self.health += value

self.last\_activity\_time = time.strftime('%Y-%m-%d %H:%M:%S')

self.save\_to\_database()

def update\_happiness(self, value):

self.happiness += value

self.last\_activity\_time = time.strftime('%Y-%m-%d %H:%M:%S')

self.save\_to\_database()

def save\_to\_database(self):

cursor.execute('''

INSERT INTO pet\_data (health, happiness, last\_activity\_time)

VALUES (?, ?, ?)

''', (self.health, self.happiness, self.last\_activity\_time))

conn.commit()

def load\_from\_database(self):

cursor.execute('SELECT \* FROM pet\_data ORDER BY id DESC LIMIT 1')

row = cursor.fetchone()

if row:

self.health = row[1]

self.happiness = row[2]

self.last\_activity\_time = row[3]

class VirtualPetApp:

def \_init\_(self, master):

self.master = master

master.title("Virtual Pet Monitor")

self.pet = VirtualPet()

self.pet.load\_from\_database()

self.health\_label = tk.Label(master, text=f"Health: {self.pet.health}")

self.health\_label.pack()

self.happiness\_label = tk.Label(master, text=f"Happiness: {self.pet.happiness}")

self.happiness\_label.pack()

self.feed\_button = tk.Button(master, text="Feed", command=self.feed\_pet)

self.feed\_button.pack()

self.play\_button = tk.Button(master, text="Play", command=self.play\_with\_pet)

self.play\_button.pack()

def feed\_pet(self):

self.pet.update\_health(10)

self.update\_labels()

def play\_with\_pet(self):

self.pet.update\_happiness(10)

self.update\_labels()

def update\_labels(self):

self.health\_label.config(text=f"Health: {self.pet.health}")

self.happiness\_label.config(text=f"Happiness: {self.pet.happiness}")

if \_name\_ == "\_main\_":

root = tk.Tk()

app = VirtualPetApp(root)

root.mainloop()

conn.close()

# Conclusion

The Virtual Pet Monitoring Application serves as a simple and enjoyable project for users interested in creating

a standalone software application using Python. It provides a foundation that can be extended with additional

features, enhanced UI elements, and even hardware integrations for a more immersive experience. The use of Tkinter

and SQLite makes it accessible to beginners while allowing for scalability and customization.