PetPal
Project Description
PetPal is a beginner-friendly web application that helps pet owners manage the daily care and health tracking of their pets. The app allows users to create profiles for each pet, track vaccinations and vet visits, set reminders for feeding or grooming, and receive breed-specific care tips powered by OpenRouter. With a simple interface and helpful tools, PetPal supports responsible and personalized pet care.

This project teaches students how to build a practical record-keeping system, integrate AI-generated content, manage reminders, and implement multi-profile data tracking.

Key Features Include

• Create and manage pet profiles with photos
• Track vaccinations, vet visits, medications
• Set reminders for feeding, grooming, and exercise
• Generate breed-specific care tips using OpenRouter
• Record pet weight and visualize growth
• Maintain care task checklist
• Store emergency contacts for each pet

Implementation Requirements

Technology Stack
• Frontend: HTML, CSS, JavaScript, Bootstrap
• Backend: Python, Flask
• Database: PostgreSQL
• LLM API: OpenRouter API
• Development Environment: VS Code

API Keys Required

• **OPENROUTER_API_KEY** — used for generating care tips, training suggestions, etc.
• Free tier available with usage limits
• Secure storage in environment variables

Core Features

1. Pet Profile Management

Users can add details for each pet, including name, species, breed, age, and photo.

2. Health Record Tracking

Users can record vaccination dates, medication schedules, vet appointments, and update weight records over time.

3. Reminders and Checklist

A simple reminder and checklist system for feeding, grooming, and medical care helps owners stay organized.

4. Care Recommendations

The app uses OpenRouter to provide breed-specific care tips, such as exercise needs or grooming routines.

Pages to Implement

1. **Home Page**
   • Welcome message and features overview
   • Buttons to log in or register
2. **Register Page**
   • Form to create an account with name, email, password
3. **Login Page**
   • Simple login form with session tracking
4. **Dashboard Page**
   • View list of user's pets
   • Add New Pet button
   • Summary of upcoming tasks or reminders
5. **Add/Edit Pet Profile Page**
   • Form with fields for name, breed, species, age, weight
   • Upload a photo
   • Submit/save profile
6. **Pet Detail Page**
   • Display full pet info
   • Buttons for: Health Records, Care Tips, Checklist
7. **Health Record Page**
   • Add/view/edit vaccinations, vet visits, medications
   • Record weight and view simple weight chart
8. **Care Checklist Page**
   • Add care tasks (e.g., grooming)
   • Mark tasks as done
   • View task history
9. **Care Tips Page**
   • Show AI-generated care tips using OpenRouter based on breed
   • Include exercise, grooming, and diet suggestions
10. **Reminder Settings Page**
    • Set recurring reminders for feeding, walks, medication
    • View upcoming reminder schedule

11. **Logout Route**
    • Clear session and return to homepage

Implementation Steps

1. **Environment Setup**
   • Install Flask and dependencies
   • Set up PostgreSQL connection and Git repository
2. **Flask App Structure**
   • Create base routes, templates, error handlers
   • Set up Jinja templates and shared layout
3. **User Authentication**
   • Register, login, logout with session management
4. **Database Integration**
   • Tables: Users, Pets, HealthRecords, Reminders, Tasks
   • Use SQLAlchemy and migrations
5. **Pet Management Features**
   • Add/edit pet profiles
   • Upload and display pet photos
   • List and delete pets
6. **Health and Weight Tracking**
   • Add vaccination/vet/medication records
   • Log and chart pet weight over time
7. **Reminders and Checklist**
   • Add and track care tasks
   • Display checklist per pet
   • Basic reminder setup (no notification system for now)
8. **OpenRouter Integration**
   • Call API with pet breed and context
   • Generate and display care tips (e.g., diet, grooming, behavior)
9. **Frontend UI/UX**
   • Style app using Bootstrap
   • Add icons and pet-friendly visuals
   • Make responsive for mobile use
10. **Testing and Documentation**
    • Test routes and database logic
    • Write setup instructions and code comments
    • Provide sample pet profiles and care logs

Evaluation Criteria

Technical Implementation (50%)
• Python Programming (10%): Modular, readable code
• Web Development (10%): Well-structured and responsive UI
• Flask Implementation (10%): Routes, sessions, template logic

• Database Design (10%): Normalized schema and clean queries
• OpenRouter Integration (10%): Working care tip generation

Functionality (30%)
• Pet Records (10%): Efficient profile and health management
• Reminders/Checklists (10%): Task tracking and organization
• AI Tips (10%): Personalized care guidance

Project Management (20%)
• Documentation (10%): Clear README and usage examples
• Structure (5%): Logical file layout
• Presentation (5%): Smooth demo and explanation