Overview

PetPal website is a pet adoption platform with the following purposes:

- Promote pet adoption: Help more pets find new homes and reduce the number of stray animals.
- Simplify the adoption process: By providing comprehensive pet information and a convenient adoption application process, make the adoption process more efficient and transparent.
- Build trust and safety: Ensure the identity and information security of all users, creating a trustworthy and reassuring adoption environment.

The target users of the website are:

- Pet owners who, for some reasons, can no longer take care of their pets.
- Adopters who wish to have pets as companions.
- Anyone concerned with animal rights.

Installation Guide

System Requirements:

- Operating System: Windows 10 or later, macOS 10.15 or later, Linux (Ubuntu 18.04+)
- Hardware Requirements: Minimum 4GB RAM, 1GHz processor

Install MongoDB

1. **Download MongoDB:**
    - Visit the MongoDB Download Center and select the appropriate version for your operating system.
2. **Install MongoDB:**
    - Follow the instructions for your specific operating system:
        - **Windows:** Run the downloaded `.msi` installer and follow the setup instructions.
        - **macOS:** Use Homebrew:

          sh

          ```
          brew tap mongodb/brew
          brew install mongodb-community
          ```

- **Linux:** Follow the specific instructions for your Linux distribution, usually involving adding the MongoDB repository and installing via the package manager.

3. **Start MongoDB:**
   - **Windows:** Use `mongod.exe` from the installed directory or as a Windows service.
   - **macOS/Linux:** Start MongoDB using:

     sh

     ```
     brew services start mongodb/brew/mongodb-community
     ```

     or

     sh

     ```
     sudo service mongod start
     ```

4. **Verify Installation:**
   - Open a terminal or command prompt and type `mongo` to enter the MongoDB shell. If it starts without errors, MongoDB is installed correctly.

Install Node.js:

- On Windows, run the downloaded installer and follow the instructions.
- On macOS, open the downloaded .pkg file and follow the instructions.
- On Linux, you can use a package manager. For example, for Ubuntu:

  sudo apt update
  sudo apt install nodejs npm

Verify Installation:

- Open a terminal or command prompt and enter the following commands to check if Node.js and npm were installed successfully:

  node -v
  npm -v

- If the version numbers are displayed, the installation was successful.

Install Dependencies:

- Install the necessary npm packages for your project. For example, if your project uses the Express framework, run:

  npm install

Run the Server:

- Run the following command in the terminal to start the Node.js server:

  npm run start

User Guide

User Sign up and Sign in

1. Sign up a New Account:
    - Steps :
        1. Click the "Signup" at the top right corner of the page.
        2. Fill in the required information such as email address and password.
        3. Click "Submit" to proceed to the dashboard.
    - Notes:
        1. The password must be at least 8 characters long.
        2. A valid email address is required.
        3. The first registered account will be designated as the "Super Admin."
2. Sign in :
    - Steps:
        1. Click the "Signin" at the top right corner of the page.
        2. Enter the email address and password used during registration.
        3. Click the "Login" button to access the dashboard.

User Management

1. Manage Users:
    - Steps:
        1. After signing in, go to the dashboard.
        2. Click on the "Manage Users" menu option.
        3. Change activation status and roles.
    - Notes:
        1. Only administrators can see the "Manage Users" section.
        2. You cannot change your own account, so you must have at least two accounts.
        3. You can only change the activation status of admins and users; hence, downgrade roles to admin or user before making changes.

Edit Profile

1. Personal Profile:
    - Steps:
        1. In the dashboard, click on the "Profile" menu option.
        2. Modify the desired information and click "Update Profile."
2. Settings:
    - Steps:
        1. In the dashboard, click on the "Settings" menu option.
        2. Enter a new password and click "Update Settings."

Add and Delete Pets

1. Add a Pet:
    - Steps:
        1. In the dashboard, click on the "Add a pet" menu option.
        2. Modify the desired information and click "Submit."

2. Delete a Pet
    - Steps:
        1. In the dashboard, click on the "My Pets" menu option.
        2. Delete the information you no longer wish to display.

Technical Details

Website Architecture

Directory Structure Overview:

Root Directory (/)

- The top-level directory that contains all files and subdirectories.

bin/

- www: Script to start the server.

middlewares/

- Custom middleware functions:
    - **auth.js**: Middleware for authentication.
    - **deleteOldImage.js**: Middleware to delete old images.
    - **uploadImage.js**: Middleware to handle image uploads.
    - **validators.js**: Middleware for request validation.

models/

- Database models:
    - **pet.js**: Pet model.
    - **user.js**: User model.

public/

- Static assets directory:
  - **images/**: Stores images.
  - **javascripts/**: Stores JavaScript files.
    - **forms/**: Stores form handler files.
      - **addPetFormHandler.js**: Form handler for adding pets.
      - **deletePetHandler.js**: Form handler for deleting pets.
      - **manageUsersHandler.js**: Form handler for managing users.
      - **profileFormHandler.js**: Form handler for profiles.
      - **settingsFormHandler.js**: Form handler for settings.
      - **signinFormHandler.js**: Form handler for signing in.
      - **signupFormHandler.js**: Form handler for signing up.
  - **stylesheets/**: Stores CSS files.

routes/

- Route handlers:
  - **auth.js**: Auth route handler.
  - **index.js**: Root route handler.
  - **pets.js**: Pets route handler.
  - **user.js**: User route handler.

uploads/

- Stores uploaded files:
  - **avatars/**: Stores avatar images.
    - **originals/**: Original avatar images.
    - **thumbnails/**: Thumbnail avatar images.
  - **pets/**: Stores pet images.
    - **originals/**: Original pet images.
    - **thumbnails/**: Thumbnail pet images.

utils/

- Utility functions:
  - **bcryptHelper.js**: Helper functions for bcrypt (password hashing).

views/

- View templates:
  - **_footer.ejs**: Footer partial template.
  - **_header.ejs**: Header partial template.
  - **404.ejs**: 404 error page template.
  - **about.ejs**: About page template.
  - **addPet.ejs**: Add pet page template.
  - **contact.ejs**: Contact page template.
  - **dashboard.ejs**: Dashboard page template.
  - **error.ejs**: Error page template.
  - **index.ejs**: Home page template.
  - **manageUsers.ejs**: Manage users page template.
  - **myPets.ejs**: My pets page template.
  - **pets.ejs**: Pets page template.

    - **profile.ejs**: Profile page template.
    - **settings.ejs**: Settings page template.
    - **signin.ejs**: Sign in page template.
    - **signup.ejs**: Sign up page template.
    - **users.ejs**: Users page template.

Miscellaneous Files

- **.env**: Environment variables file.
- **app.js**: Main application file.
- **package.json**: Application dependencies and settings.
- **package-lock.json**: Ensures consistent dependency versions.
- **README.txt**: Project documentation.

Database Design

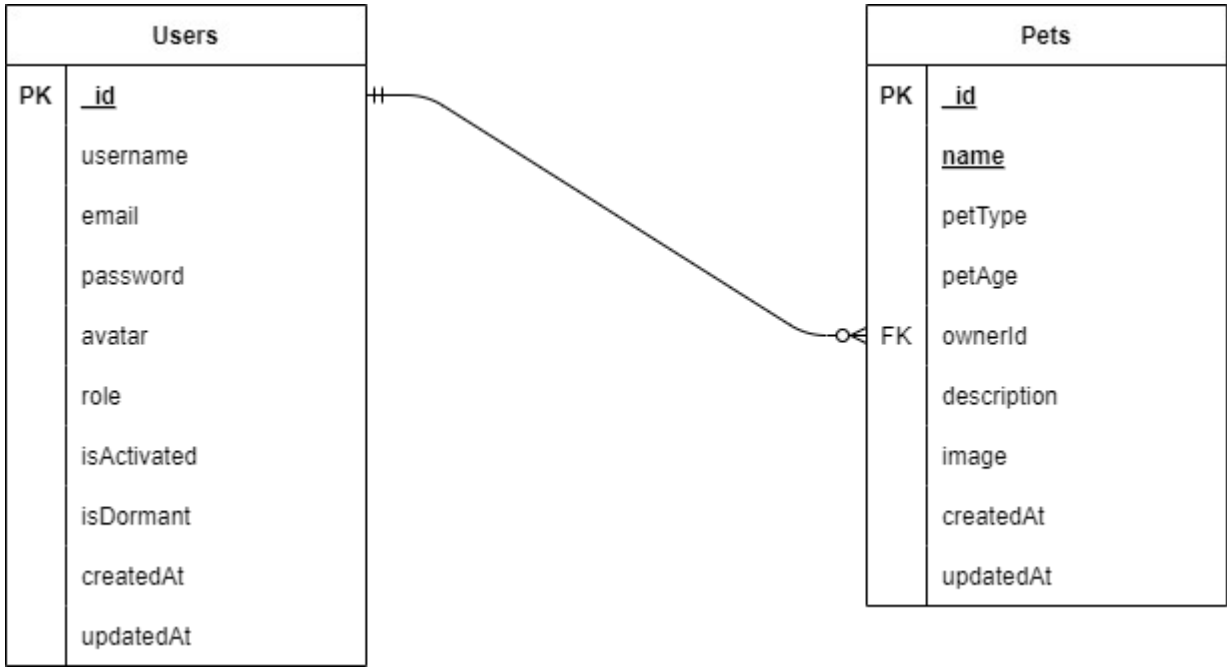
Entities and Relationships

1. User:
    - Attributes: user_id (PK), username, email, password, avatar, role, isActivated, isDormant, createdAt, updatedAt
    - Relationships: One-to-Many with Pet (a user can have multiple pets)

2. Pet:
    - Attributes: pet_id (PK), name, petType, petAge, ownerId (FK referencing user_id), description, image, createdAt, updatedAt
    - Relationships: Many-to-One with User (each pet belongs to a single user)

ER Diagram

A visual representation of the database structure



Test Scripts

Description: These scripts are used to generate and populate the database with test records, making it easier to test the application functionalities.

Scripts:

1. **createUsers.js**
    - **Purpose**: Generates sample user records in the database.
    - **Usage**:

      sh

      ```
      node createUsers.js
      ```

    - **Details**: This script creates a set number of user records with random data, helping to simulate a populated user base.
2. **createPets.js**
    - **Purpose**: Generates sample pet records in the database.
    - **Usage**:

      sh

```
node createPets.js
```

- **Details**: This script creates a set number of pet records associated with existing users, useful for testing pet-related features.