**ELEC5619 Final Report**

Pet connect

2023/10/31

Qinyan Zhong 520110870

Junqi Su 530587839

Jie Yang 530204970

Yaqi Zhou 530601881

# 1. Introduction

## 1.1 Overview

The PetConnect is an online platform based on the Spring Boot framework, which aims to provide a community for pet lovers to exchange, consult and share resources, enabling pet lovers to share their experiences and love. This platform offers a variety of features, including expert consultation, pet diary and forum functions to meet the various needs of pet lovers. The PetConnect aims to connect pet owners and provide them with knowledge, experience and resources to improve the quality of their pets' lives, while also promoting pet rescue and adoption.

## 1.2 Aims

* Provide a comprehensive platform for pet lovers: Let pet lovers get all the information about pet care, health, daily management on the platform, which can Improve the quality of pet life.
* Promote pet rescue and adoption: The platform will update pet rescue and adoption information in real time to encourage people to pay attention to pet rescue
* Expert advice and support: Pet lovers can find pet experts on the platform to seek advice and answer questions, so as to better care for their pets.
* Build an active community: Create an interactive community where pet lovers can share diaries, experiences and stories about their pets to facilitate mutual learning and interaction.
* Spread pet love and responsibility: Through projects to strengthen pet lovers' love and responsibility for pet care

## 1.3 Project Scope

* Project content:The PetConnect Project is a web application based on frontend React, backend Spring Boot that aims to provide an interactive platform for pet lovers to share pet diaries, participate in forum discussions, and more.
* Project functions and features:

1. User authentication: Users can register new accounts or log in to existing accounts to access all features on the platform, while the user's password and identity information will be protected.
2. Pet diary: Users can create pet diaries to record their pets' growth and life.
3. Community Blog: Users can view, create and delete posts.
4. Live Weather Display: Displays real-time weather conditions in the user's area to help users decide whether it is appropriate for dog walking or other pet activities.
5. Event Calendar: The platform provides a calendar of pet-related activities and events that users can preview and plan their trips in advance.
6. AI Pet Assistant: Provides a smart pet assistant to which users can ask pet-related questions and get advice on diet, health and training.
7. Pet adoption: Users can browse information about pet adoption on the platform, and if they are interested in a particular pet, they can click on the details page to adopt.
8. Rescue Suggestion Reading: Users can browse articles about pet rescue
9. Article Search: Users can search for the corresponding article based on the title of the article.

* project implementation

1. Use Spring Boot to build back-end services, including authentication, data storage, and business logic.
2. Use front-end technologies such as React to build user interfaces and front-end presentations of individual functional modules.
3. Integrate third-party weather data apis for real-time weather information.
4. Integrate third-party pet adoption data external apis for real-time pet adoption information.
5. The database uses MySQL to store user data, diaries, posts and other content.

* Project deliverables:

1. Complete source code, including front-end and back-end.
2. readme documentation: Provides users with installation, configuration, and usage instructions.

## 1.4 Project Scenarios

**Best scenario:**

1. Seamless registration and authentication: First make sure the user registration and authentication process is rigorously tested to ensure proper functioning.The second is to provide a user-friendly interface that guides users through the registration and verification steps easily.
2. Effective AI pet assistant: We need to continuously monitor the performance of the AI pet assistant and collect user feedback to make functional improvements.
3. Active user interaction: The platform should first host interesting and engaging community activities, such as contests, challenges and sweepstakes, to encourage users to actively participate in the community, such as pet diaries, forum posts, etc.
4. Pet articles received positive feedback: We need to attract professional authors and pet experts to contribute high-quality pet articles to the platform.

**Worst scenario:**

1. Technical issues are causing registration and login problems：We can set up alternative authentication methods, such as login via SMS verification codes or social media, while Posting notices about issues and solutions on the website so that users understand the issues and get help.
2. AI pet assistants face accuracy and reliability challenges：First, we need to closely monitor the ai pet assistant in order to identify and correct accuracy issues, and second, we can train the AI model for specific areas to improve reliability. Finally, you can also provide alternatives or links to relevant pet resources to make up for the lack of AI.
3. Low user engagement: The platform can implement marketing strategies to increase the visibility and attractiveness of the platform, and can also enable social sharing features to expand the user base.
4. Pet articles are not professional enough: We can create partnerships to attract professional authors and pet experts, while also providing training and support to help authors create quality content.

**Most probable scenario:**

1. Registration and login issues occur occasionally when PetConnect is running, but are resolved in a timely manner.
2. AI pet assistants can provide useful insights, though not always perfect.
3. Users often use community features to actively participate in discussions and adopt pets.
4. Pet articles get moderate contributions and generate moderate interest.

## 1.5 Audiences and Primary Users

* Pet Enthusiasts: This is the core target audience of the project. They love pets, may have pets of their own, or are interested in pet ownership and care. They may use the platform to share pet diaries, exchange experiences, participate in discussions, or get pet care advice.
* Novice Pet Owners: The user may be a first-time pet owner with questions about pet ownership, training and health, and therefore may use the platform for advice, education and guidance.
* Potential Pet Owners: These users may not have a pet yet, but are interested in owning one. They may use the platform to learn about the responsibilities of owning a pet, choose the right kind of pet, and learn about the advantages and challenges of owning a pet.
* Pet rescuer: These users search for information about pet rescues on the platform and decide whether to rescue them.

# 2. Project Architecture

## 2.1 System Architecture

### Ad-hoc Diagram

Upon successful registration and login, users will be directed to the blog interface. Within the blog section, users can peruse posts from other pet enthusiasts, share their own comments, and exchange their pet stories and experiences with the community. Additionally, when users access the notification icon within the application, they will receive timely reminders related to pet diary entries. These prompts serve as a nudge for users to document their pets' daily activities and dietary habits.

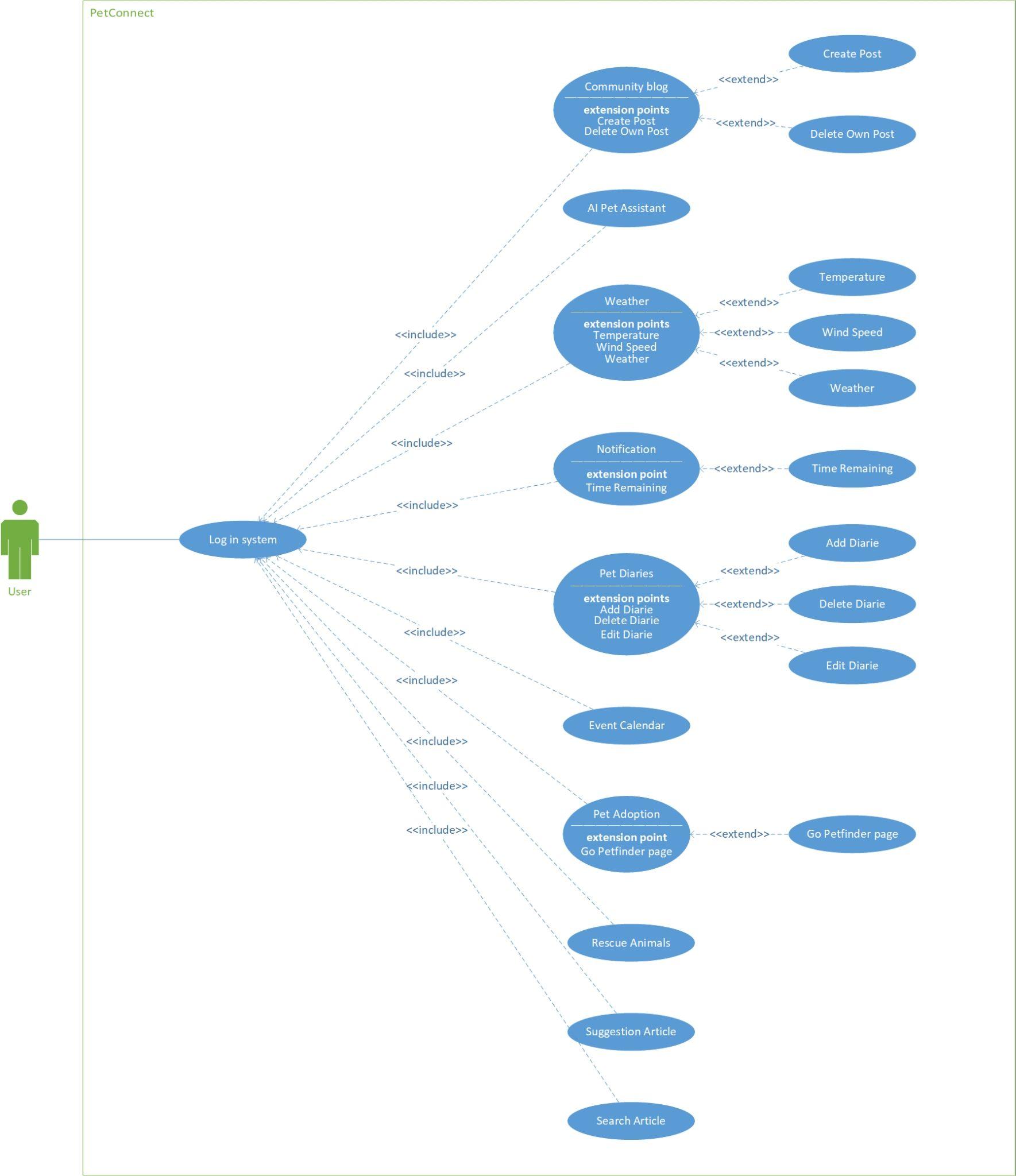
By clicking the "Pet Diary" button, users can access a dedicated page for recording their pets' daily routines and dietary habits. This feature allows users to select specific dates and input detailed information, such as feeding schedules, types of food, and pet activities, facilitating the maintenance of comprehensive records for effective pet management.

Furthermore, the "Adoption Information" section enables users to conveniently browse through a list of pets available for adoption from local shelters. Each pet listing is accompanied by photos and brief descriptions, empowering users to submit adoption applications for the pets they wish to welcome into their homes directly through the application.

When accessing the calendar interface, users can easily identify marked dates corresponding to recorded pet diary entries. Selecting a particular date grants users access to specific entries for that day. By clicking into the rescue section, users can explore articles pertaining to animal rescue efforts, thereby gaining insights into various aspects of animal rescue and welfare.

Moreover, users can check real-time weather conditions and obtain advice for pet-friendly outdoor excursions within the weather interface. By integrating the aforementioned functionalities, the application provides users with a comprehensive and organized platform for managing their pets and engaging with the community.

### Use Case Diagram



### Use case description

Use Case: Log in System

Description: User logs into the system.

Main Flow: The user enters their username and password, the system verifies the user's information, and if the verification is successful, the user logs into the system.

Use Case: Community Blog

Description: Users can view and publish blog articles.

Main Flow: Users browse blogs and have the option to create new blogs or delete their own blogs.

Use Case: AI Pet Assistant

Description: Users ask questions, and the FastGPT provides information and advice related to pet care.

Use Case: Weather

Description: Users can check weather information.

Main Flow: Users check the current temperature, wind speed, and weather conditions.

Use Case: Notification

Description: Users receive notifications.

Main Flow: Users receive notifications show how long it is until the deadline.

Use Case: Pet Diaries

Description: Users can record and manage pet diaries.

Main Flow: Users add, delete, and edit pet diaries.

Use Case: Event Calendar

Description: Users can view highlight dates in the calendar.

Use Case: Pet Adoption

Description: Users can view pet adoption information.

Main Flow: Users can navigate to the Petfinder information page.

Use Case: Rescue Animals

Description: Users can browse information about rescue animals.

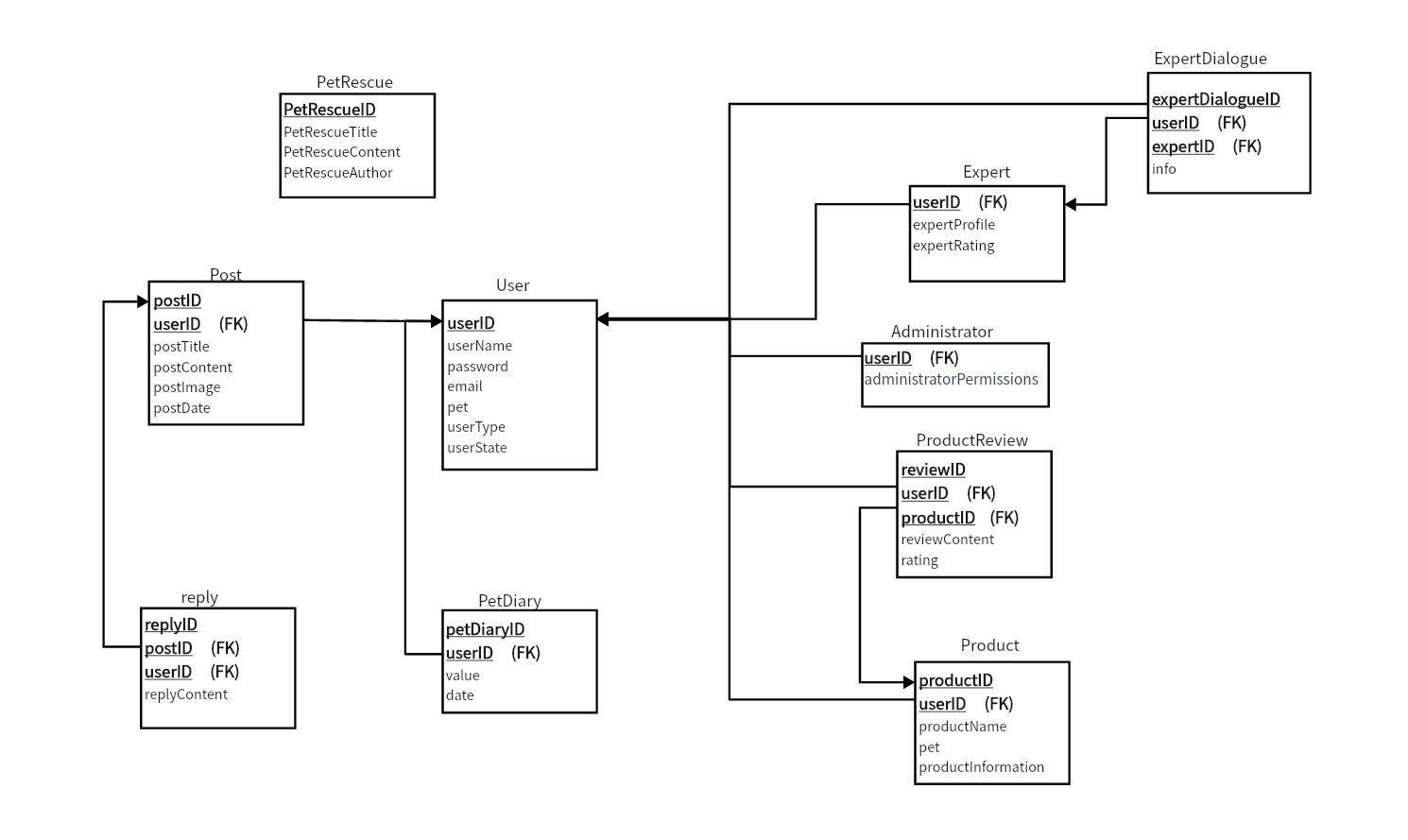
Use Case: Suggestion Article

Description: Users browse articles title from the right sidebar..

Use Case: Search Article

Description: Users can search for specific articles in the navbar.

## 2.2 System data model (ERD)



Compared with the original ERD design, some unnecessary data are removed, such as all timestamp attributes, event table, and adoption table. It also adds data that was not considered, such as the petdiary table.

# 3. Functionalities

## 3.1 Project functionalities

1. Login Function:

- Users should enter their username and password for validation.

1. Account Creation:

- Users should be able to create accounts with unique usernames and passwords and also should input information including owner data (name, email, pet, password).

1. Authentication and Error Handling:

- Authenticate user input based on account information stored in the database. If incorrect, indicate which information is erroneous.

1. Logout Function:

- Allow users to log out and terminate their session after completing actions.

1. AI Pet Assistant:

- Offer personalized AI-based pet training and dietary guidance using FastGPT, tailored to the provided information about the pet and owner. This service aims to address common pet care concerns, improve the pet's physical and mental health, resolve everyday pet-related challenges, and foster stronger owner-pet relationships.

1. Real-time Weather Display:

- The application presents the live temperature, wind speed, and weather status, offering users an immediate understanding of the current weather conditions in their location. By providing clear and concise weather information and tailored suggestions, the application helps users make informed decisions about their pet's outdoor activities, ensuring their comfort and safety in various weather conditions.

1. Pet Diaries:

- Enable users to log their pets' daily activities and dietary habits by providing input fields for recording the time and corresponding details. This feature allows users to maintain a comprehensive record of their pets' routines, including feeding schedules, exercise regimens, and any notable occurrences, fostering a better understanding of their pets' well-being and behavior patterns

1. Community Communication:

- Create a user-friendly blog for pet owners to communicate and share experiences and knowledge about pets via text-based posts. And users have the capability to delete their own posts, ensuring control over their shared content.

1. Event Calendar:

- Present users with a calendar showcasing scheduled community events, allowing them to preview and plan for upcoming activities. Highlight the specific day in the calendar where the user has recorded a pet diary entry, enabling easy identification and quick access to their documented activities and observations on that particular day. This integration enhances user engagement, ensuring a seamless and organized experience for managing both community events and personal pet-related data.

1. Notification Functionality:

- Based on owner-entered time in Pet Diaries, provide timely reminders for tasks such as daily activities.

1. Adoption Information:

- This functionality involves embedding the external API directly into the frontend. It allows the website to seamlessly display pet images and names sourced from the petfinder, enriching the user experience and fostering engagement with the adoption listings. By integrating this feature, users can readily explore and connect with potential pets for adoption, facilitating a smooth and interactive adoption process.

1. Animal Rescue:

- Provide information about stray animal rescue events and educational content, focusing on promoting animal welfare and charitable efforts.

1. Rescue Suggestion Reading:

- Implement a rescue suggestion reading section that showcases a list of animal rescue-related article titles in the right-hand menu. Users can click on a specific title to navigate directly to the corresponding article, encouraging them to become more aware of and engaged with animal rescue efforts. This feature aims to raise awareness about animal welfare and promote a sense of responsibility and compassion towards animals in need.

1. Article Search：

- Implement a feature that enables users to search for articles by their titles. Once a user inputs the title in the search bar, the system retrieves relevant articles, allowing users to navigate directly to the desired article. This functionality enhances user convenience, making it easier for users to find specific articles of interest.

## 3.2 Satisfied non-functional requirements

1. Login Authentication：

Implement login authentication to restrict access to the main page only for logged-in users. Unauthenticated users should only be able to access the login and registration pages. Any attempt to access the main page should automatically redirect to the login page. This feature ensures secure access control, protecting sensitive information and maintaining the integrity of user data and interactions.

2. Security:

Deploying the database on Alibaba Cloud enhances security, safeguarding the system from potential vulnerabilities and external threats.

3. Responsive Design:

Implementing a responsive design allows the website to seamlessly adapt to various screen sizes, including those of mobile devices and tablets, ensuring an optimal user experience across multiple platforms.

4. Slider Verification:

a. Security Requirement: The slider verification component serves the purpose of implementing a captcha verification to protect the website from automated attacks. This feature ensures the integrity of the website's data and functionality against malicious activities.

b. User Experience Requirement: By integrating user-friendly slider verification, the component contributes to an enhanced user experience. The intuitive and smooth verification process reduces unnecessary hassle for the users.

5. Maintainability:

a. Modular Code: Employing a modular architecture with React ensures that the code is easily maintainable and scalable.

b. Documentation and Comments: Providing clear code documentation and comments facilitates better understanding and modification of the code by team members.

6. Testing:

Unit Testing: Writing unit tests to validate the correctness of components and functionalities.

7. User Experience:

a. Interface Design: Implementing an appealing user interface design enhances the overall user experience.

b. Navigation: Ensuring a straightforward website navigation system enables users to easily locate the desired information.

c. Performance Optimization: Providing a seamless user experience, including smooth scrolling, fast page loading, and instantaneous responsiveness.

d. Redirection: After successful login, redirect users to their personal profile/dashboard.

8. Fault Tolerance:

Error Handling: Offering user-friendly error messages and handling mechanisms to address user input errors and system failures effectively

# 4. APIs and Spring

## 4.1 External APIs

In our PetConnect project, we have incorporated two external APIs to enhance the platform's functionality and provide a comprehensive experience for pet lovers.

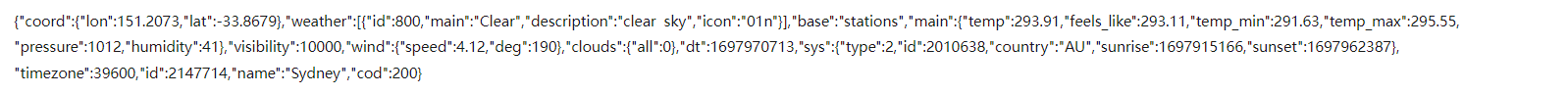
Front-end API is the Petfinder API, we use the Petfinder JS SDK, a JavaScript wrapper designed specifically for the Petfinder API. which grants us access to the vast Petfinder database, including the name of various animals, their images, and the link to their petfinder page. This information is then skillfully presented on our platform's user interface, enabling users to browse and explore the listings. With the Petfinder API, we're able to provide our users with essential features and resources that align with the mission of PetConnect. And this feature empowers our users to seamlessly transition to the Petfinder platform for more in-depth information and potential adoption inquiries.



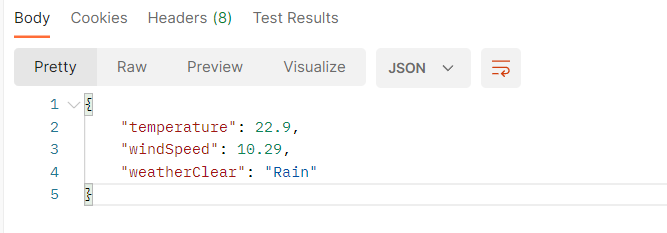
Openweather：

OpenWeather is an external API service that provides global weather data, allowing users to obtain real-time and forecast weather information for a specific location through API requests, including temperature, humidity, wind speed, air pressure, and more.

Integrating the OpenWeather API enables us to provide users with real-time weather data for specific locations, which is very important for pet activity recommendations, as different weather conditions may affect your pet's activity. For example, pets may need proper protection in cold or inclement weather, while pets may be able to enjoy outdoor activities in warm and sunny weather.



These are the original data we obtained from openweather. After analyzing the relationship between pet activity and weather, we finally selected temperature, windSpeed and weatherClear, where the original value of temperature is in Kelvin. To make it easier for the user to understand, we've converted the temperature unit to Celsius, and here's what jason gets back：



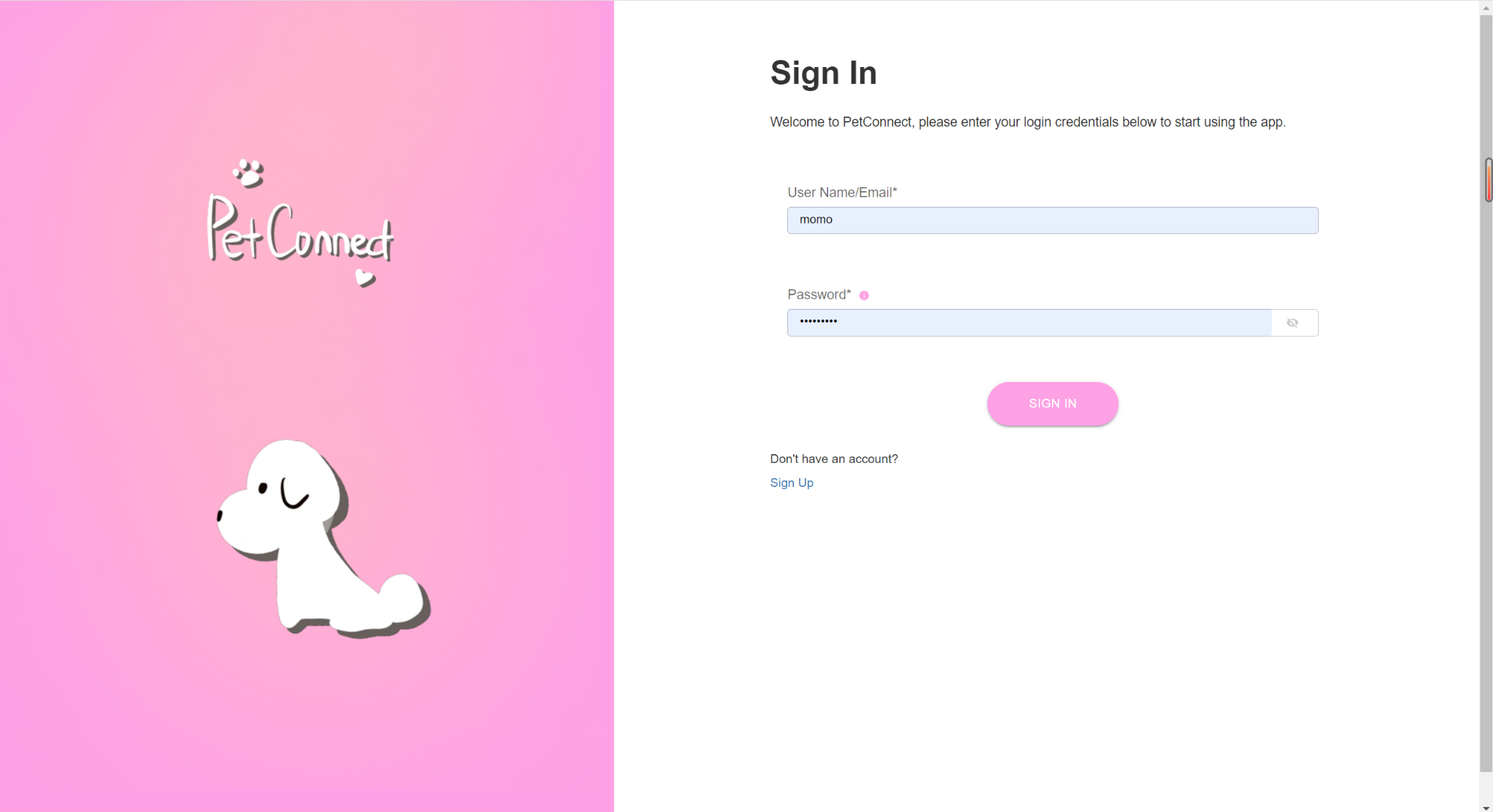
## 4.2 Spring

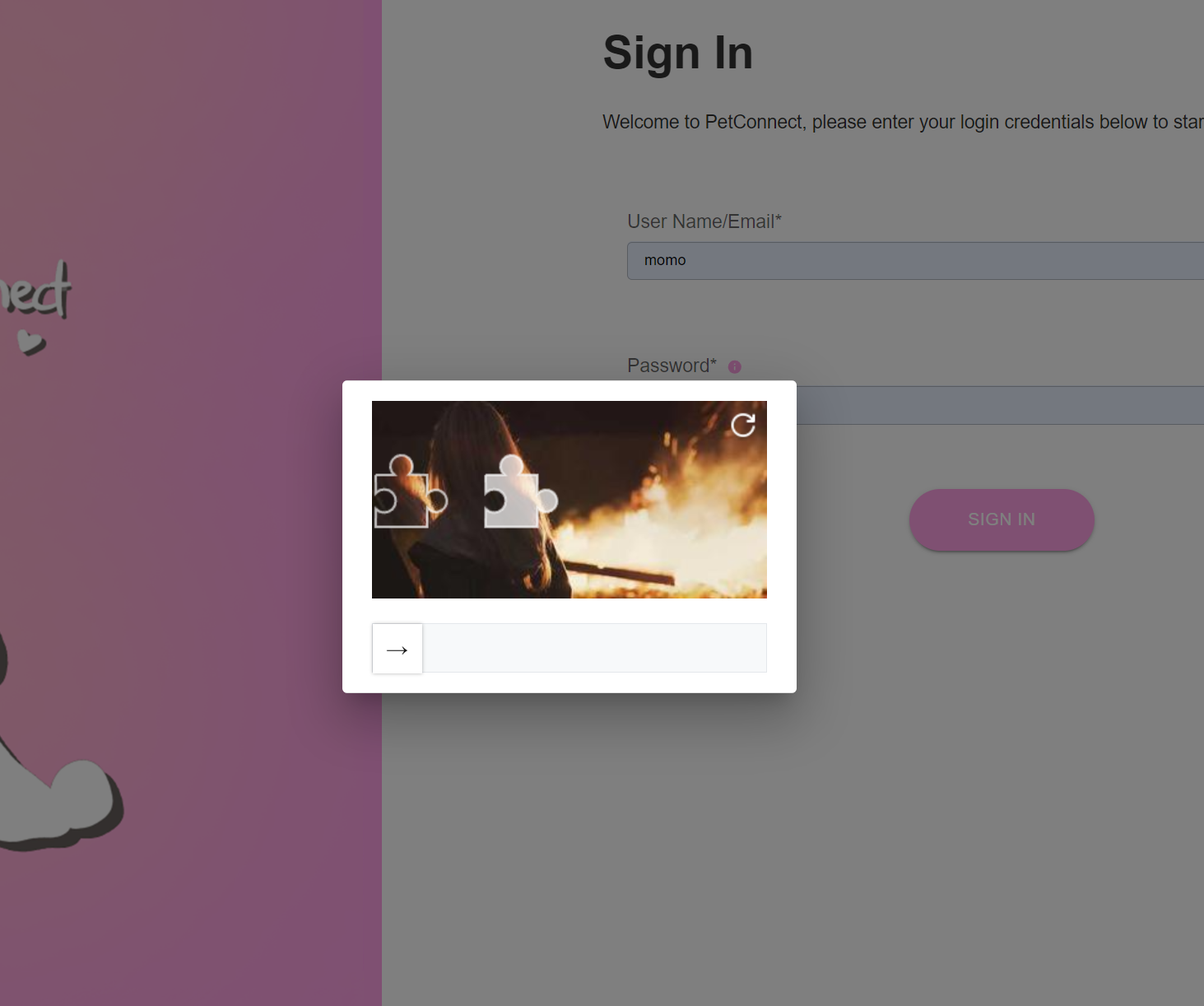
* getuserIdbyuserNameController：This controller handles the user's request to get the user ID based on the user name, passes the request to the UserService to perform the actual lookup operation, and returns the corresponding result.
* logoutController：This controller is used to handle user requests to exit the login operation. When a user wants to log out, they can send an HTTP POST request to the /log\_out path, which the controller passes to the UserService to perform the actual logout, and then returns the corresponding result.
* logStateCheckController：logStateCheckController Controller Used to check the login status of users. The user can trigger this controller by sending an HTTP POST request to the /log\_state\_check path, which then passes the request to the UserService to check the user's login status and return the appropriate status code.
* logStateCheckController： Controller is used to check the login status of users. The user can trigger this controller by sending an HTTP POST request to the /log\_state\_check path, which then passes the request to the UserService to check the user's login status and return the appropriate status code.
* PetDiaryController： The PetdiaryController is used to add, edit, delete, and get a pet diary
* PetProductController: The petProductController is used to query pet products. Users can search for related pet products based on pet information and obtain all pet products in the system. This is important for displaying and managing pet products in Pet Forum projects, which users can do with different HTTP requests.
* PetRescueArticleController: This controller is responsible for updating and displaying pet rescue articles
* PostController: The controller is used to perform Create, Read, Update, and Delete (CRUD) operations for posts, allowing users to get all posts, get posts based on the post ID, create new posts, update existing posts, and delete posts.
* ProductReviewController: The controller is used to add, edit, delete, and get product reviews, allowing users to add new reviews, edit existing reviews, delete reviews, and get related reviews based on the product ID.
* ReceiveExpertDialogueController: controller's main purpose is to receive experts dialogue and according to the need to implement the related operations, this can help users ask experts, experts provide reply or advice.
* ReplyController Controller: The main purpose is to allow the user to create a reply and get a reply list for a particular post for the comment and reply function of a post in the Pet Forum, so that the user can reply to a particular post and get all the replies for that post through the post identifier.
* ResponseMessage：A message or status information is returned as part of the response so that the client can understand the result or status of the operation.
* SendExpertDialogueController: controller sends a request for processing experts dialogue, returned to the corresponding dialog information or results after processing, are used to interact with experts
* signingController: The controller is used to process user login requests, verify user identities, and return login status or related information.
* signupController: The signupController is used to process user registration requests, verify registration information, create new user accounts, and return registration status or related information.
* WeatherController: This is mainly used to get weather information by calling the OpenWeather API, providing an endpoint to get weather information for a specified city, and then returning the data to the client.

# 5. Presentation Layer

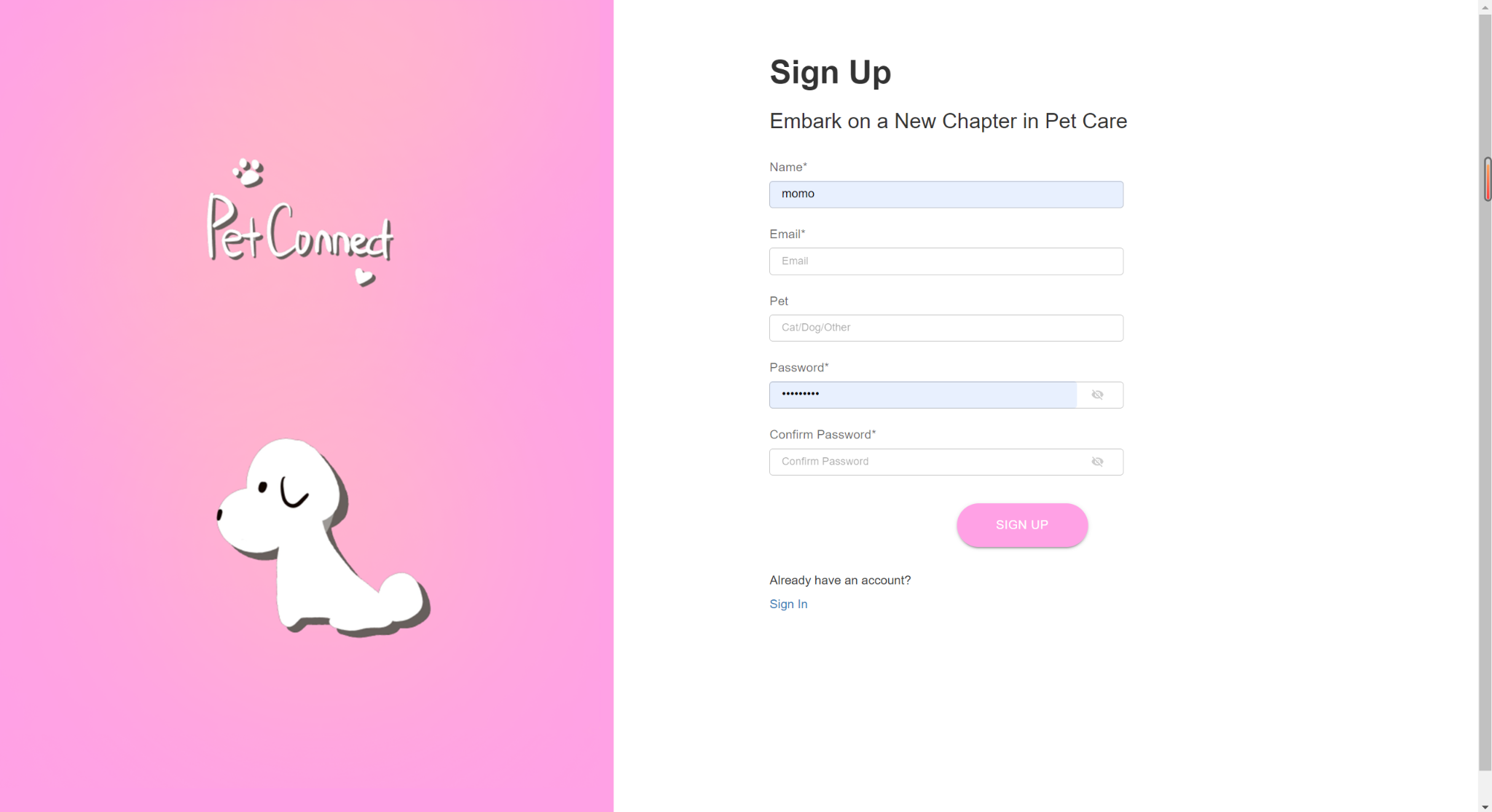
We have 9 pages, including Sign In page, Sign Up page, Home page (Community blog), AI Pet Assistant page, Weather page, Pet Diaries page, Event Calendar page, Pet Adoption page and Rescue Animals page.

Sign In page

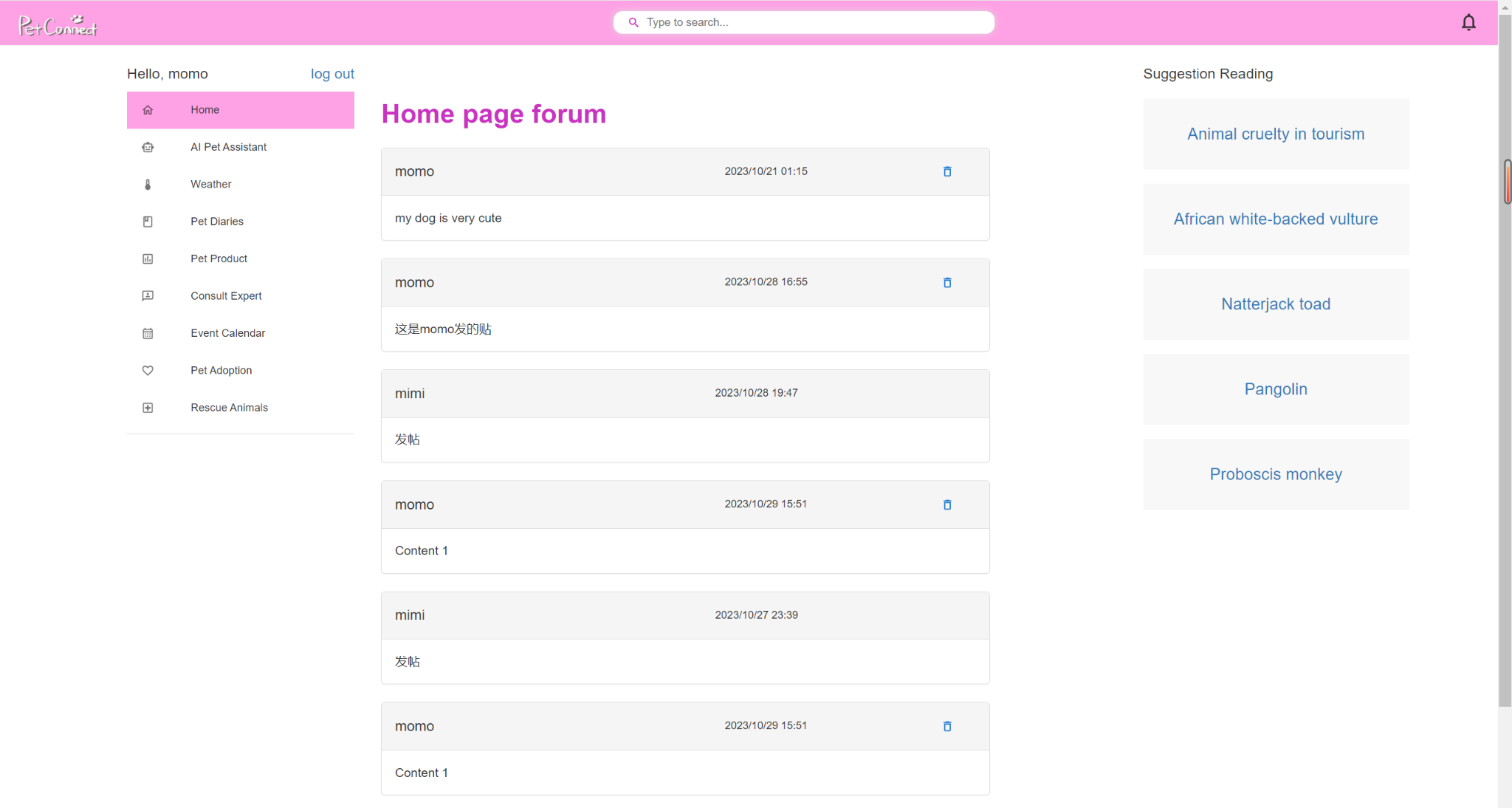




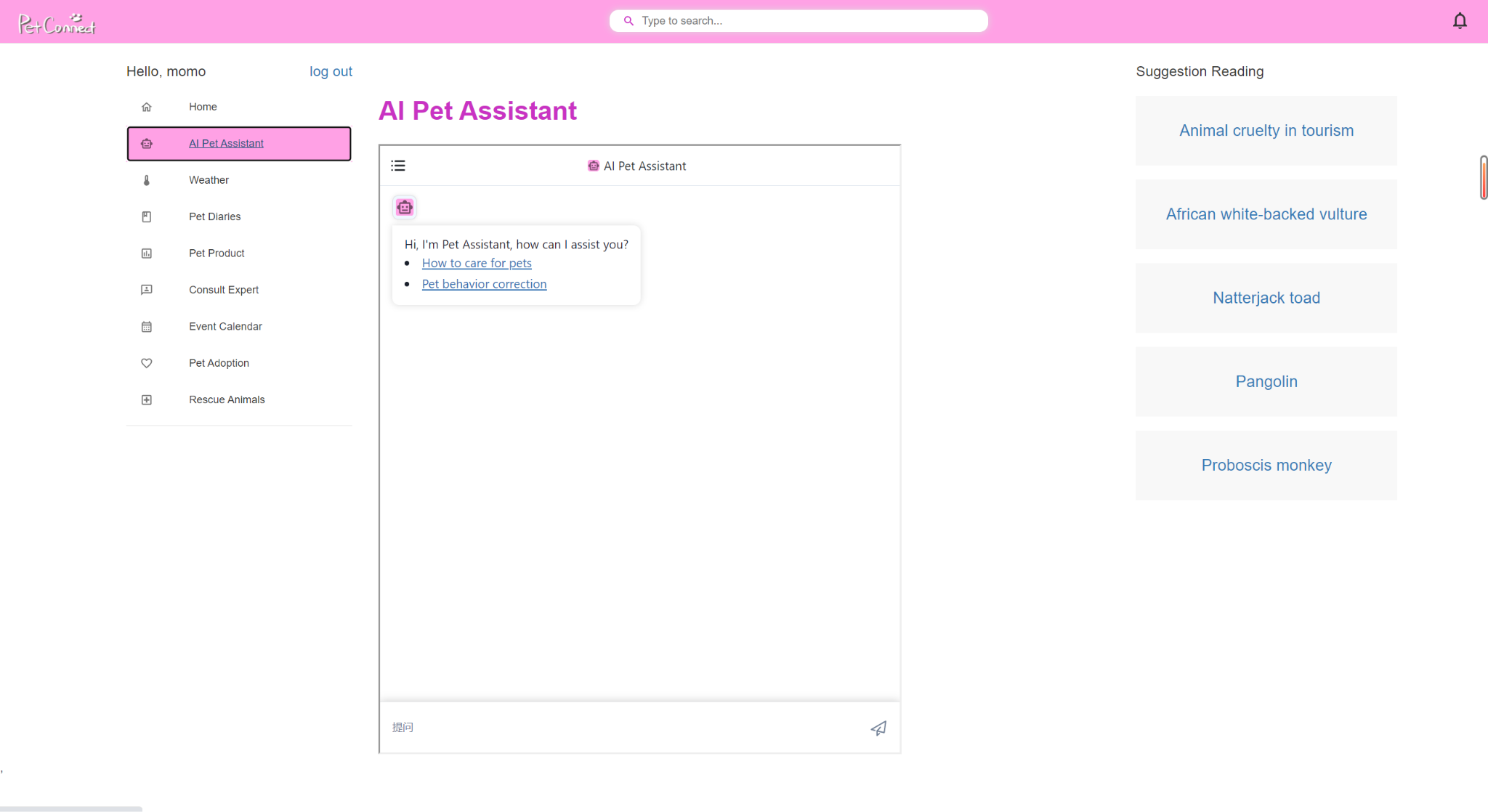
Sign Up page



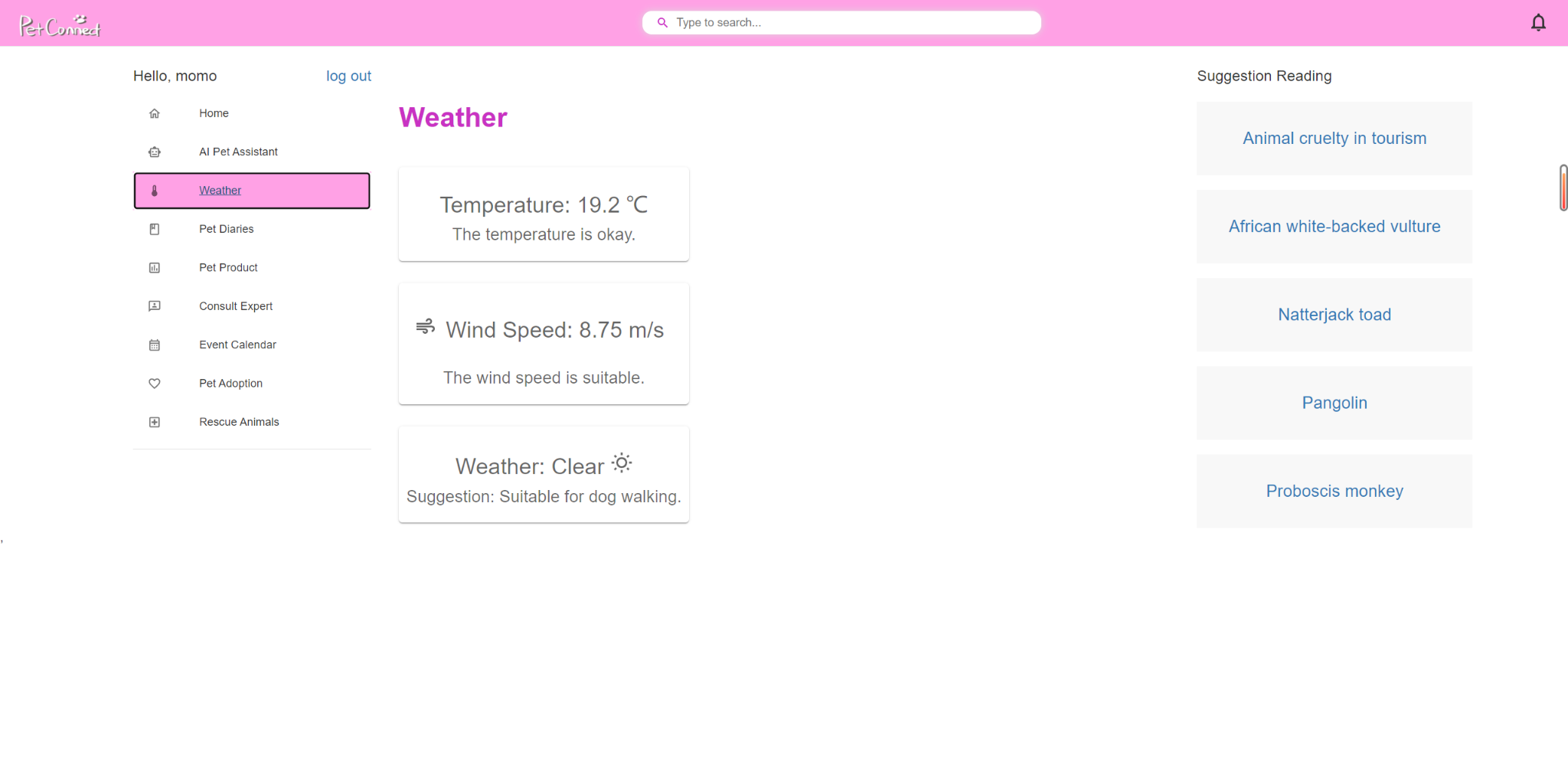
Home page (Community blog)



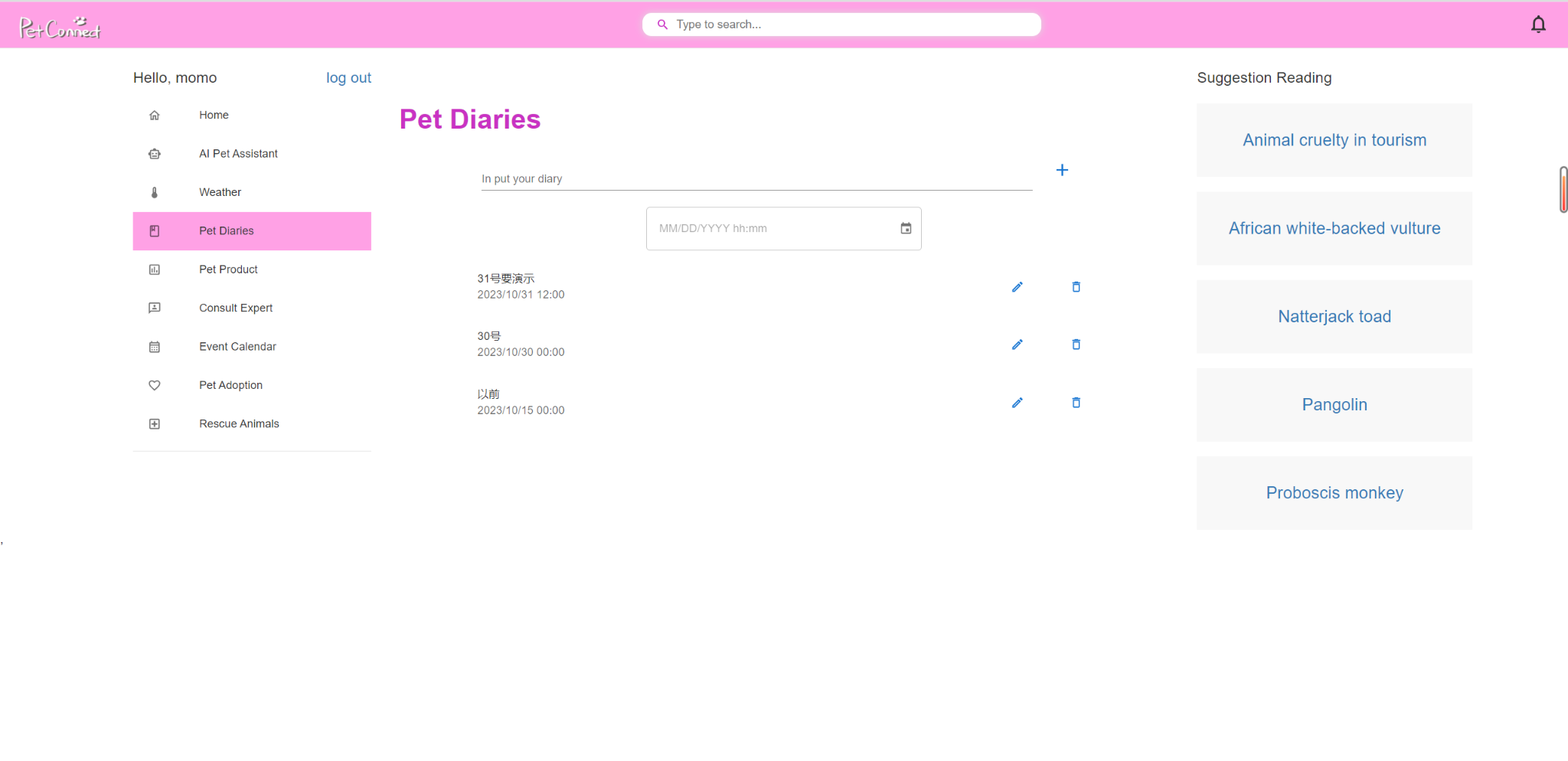
AI Pet Assistant page



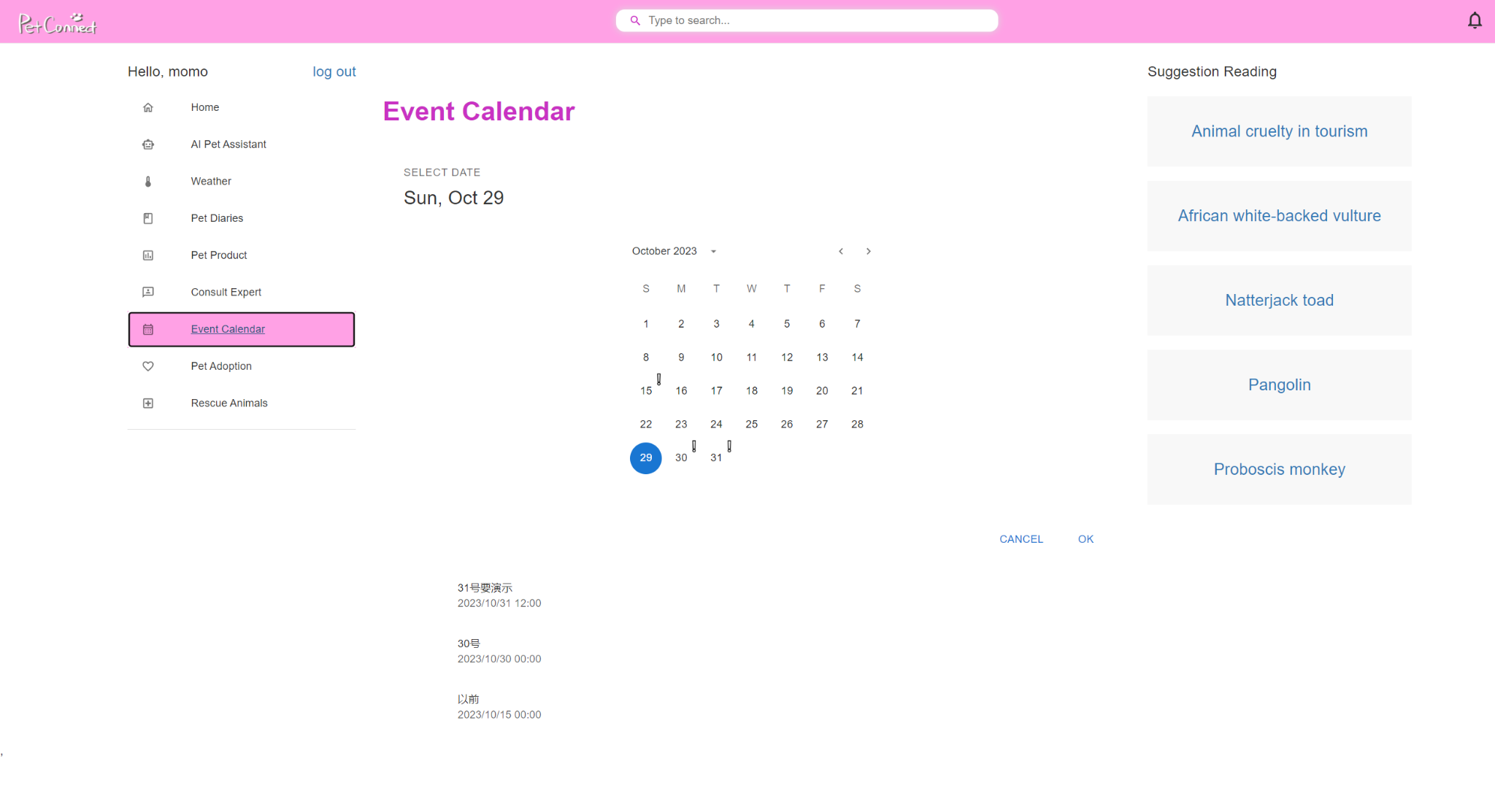
Weather page



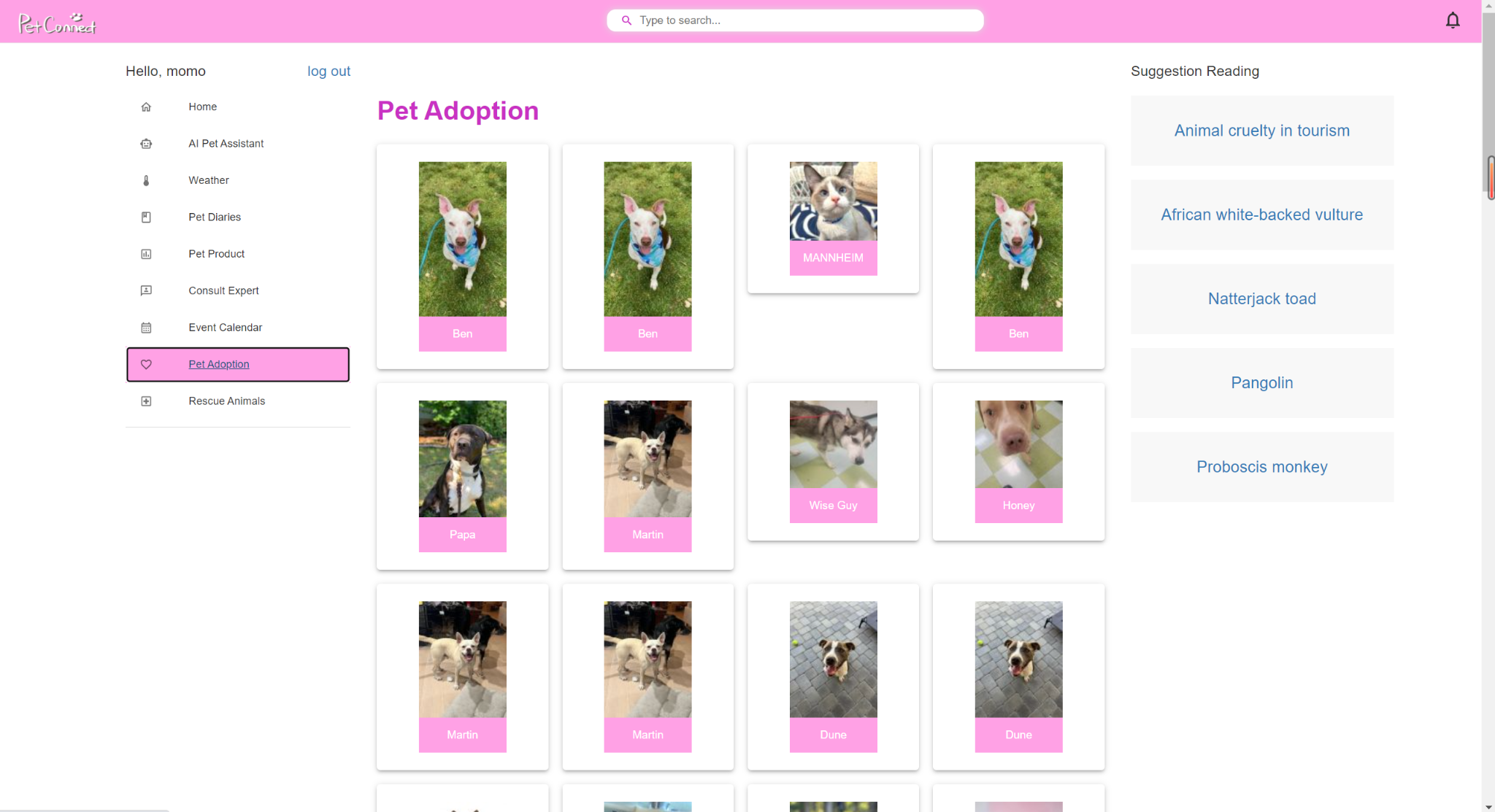
Pet Diaries page



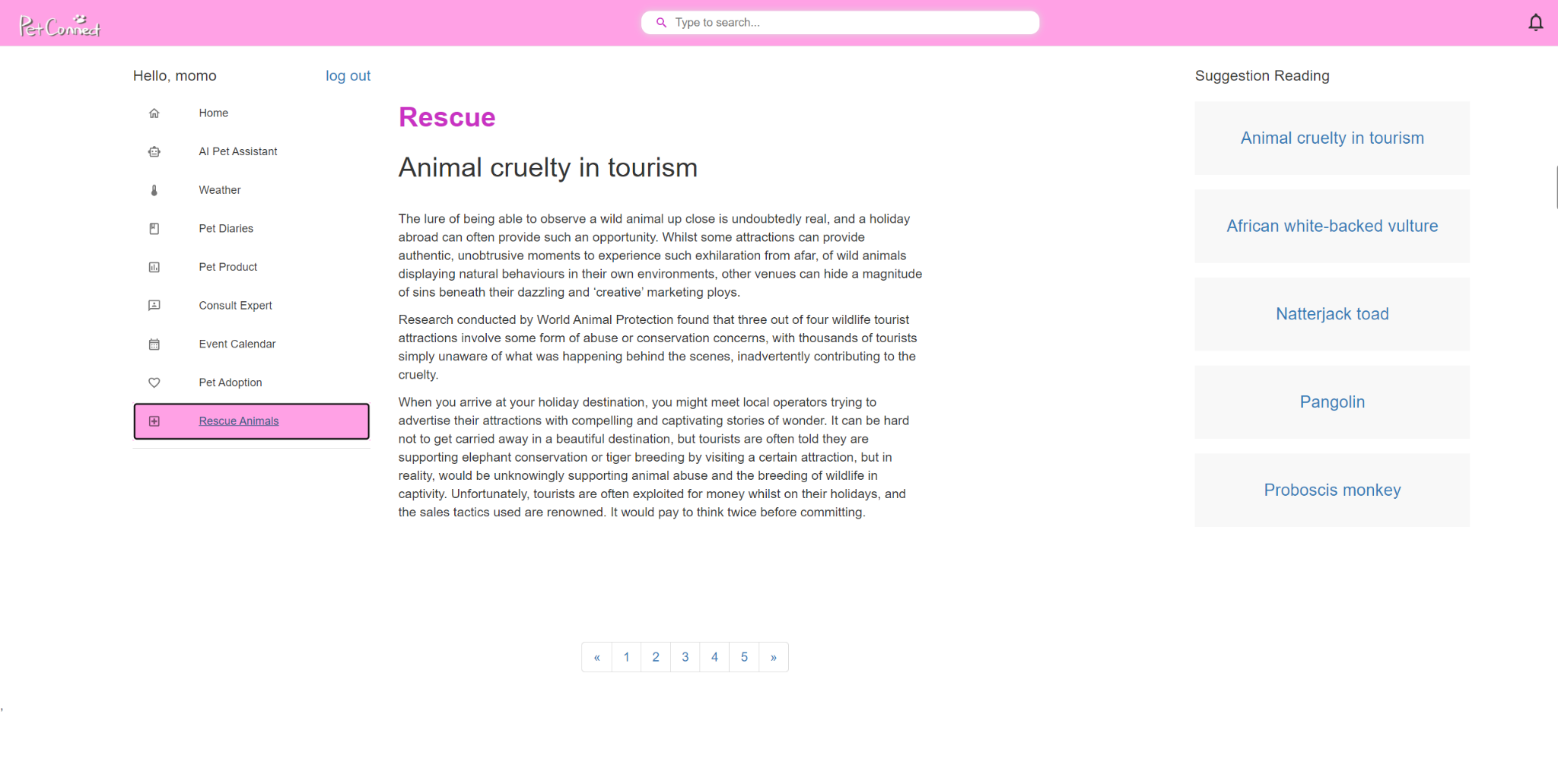
Event Calendar page



Pet Adoption page

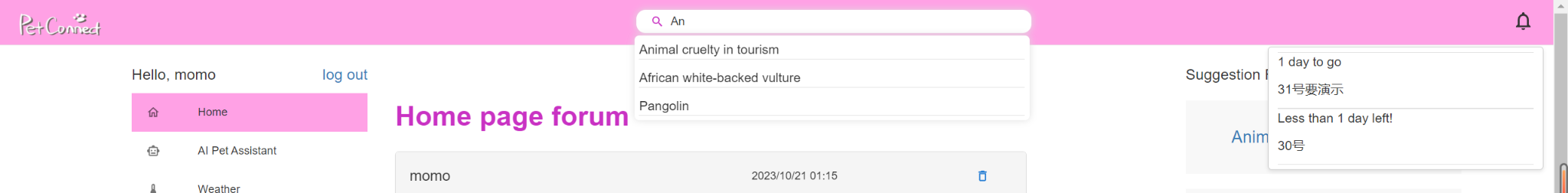


Rescue Animals page

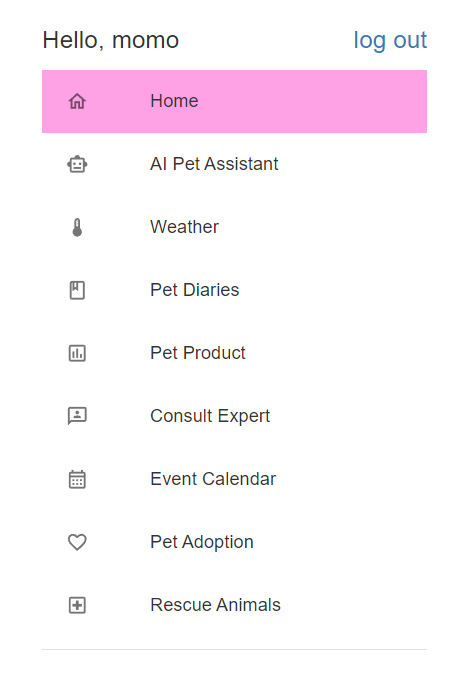


We have three components:

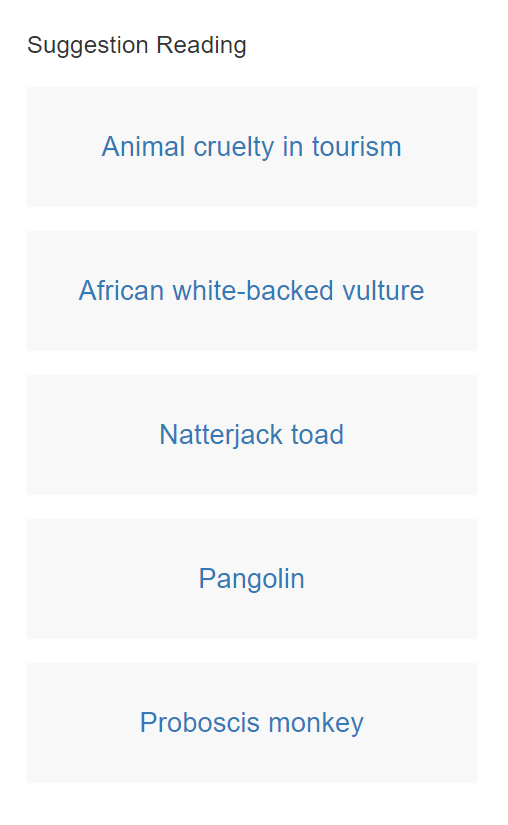
Navigation bar (displaying the website logo, search bar, and notifications)



Left bar (for page navigation)



Right bar (for suggested readings).



We used the following external libraries in the frontend: Bootstrap v3, Material UI v5, and react-slider-vertify.

We employed the Axios JavaScript library for handling AJAX requests.

# 6. Testing

### Front-end Validation Testing：

Front-end Validation Testing is conducted to ensure that the front-end interface and functionality perform as expected. During this type of testing, we meticulously outline various test cases and provide comprehensive explanations of expected outcomes for different testing scenarios. These test cases are executed systematically to perform Validation Testing, ensuring the correctness and reliability of the front-end. This process helps verify if the front-end portion meets requirements and expectations and ensures high-quality performance of the user interface and functionality.

1. Log in

1.1 Ensure login fails when the username or email is empty.

1.2 Ensure login fails when the password is empty.

1.3 Ensure login fails when the username and password do not match.

1.4 Ensure login fails when the slider verification is not passed.

1.5 Verify that users can successfully log in by entering the correct username and password and passing the slider verification.

1.6 Ensure successful login results in redirection to the main page.

1. Registration

2.1 Ensure registration fails when required fields are empty.

2.2 Ensure registration fails when the email format does not meet the requirements.

2.3 Ensure registration fails when the password length does not meet the requirements.

2.4 Ensure registration fails when the entered password and confirmation password do not match.

2.5 Ensure users can successfully register by providing valid information, including username, password, and email.

2.6 Ensure registration fails when attempting to register with an already existing username or email.

2.7 Ensure successful registration results in redirection to the login page.

1. Authentication and Error Handling

3.1 Ensure that users who are not logged in cannot access any pages other than the login and registration pages.

3.2 Ensure that logged-in users can access all pages, including the login and registration pages.

1. Log Out

4.1 When users click "Log Out," ensure they are logged out and redirected to the login page.

1. AI Pet Assistant

5.1 When accessing the AI Pet Assistant page, ensure that users can input text in the chatbox to interact with the AI.

1. Real-time Weather Display

6.1 When accessing the Weather page, ensure that users can view the current temperature, wind speed, and weather conditions.

1. Pet Diaries

7.1 Ensure that diaries cannot be added when the diary content is empty.

7.2 Ensure that diaries cannot be added when the diary time is empty.

7.3 When entering valid diary content and diary time, ensure that users can add diaries.

7.4 When the current user has diaries, ensure that users can see their diary list.

7.5 When the current user has diaries, ensure that users can edit their diaries.

7.6 When the current user has diaries, ensure that users can delete their diaries.

1. Community blog

8.1 When there are posts, ensure that all users can see a list of all posts, including the poster, post content, and posting time.

8.2 When the current user has posted, ensure that users can delete their own posts.

8.3 Ensure that posts cannot be created when the post content is empty.

8.4 When entering valid post content, ensure that posts can be created.

8.5 After a user posts, ensure that the new post is included in the post list.

8.6 After a user deletes a post, ensure that the deleted post is no longer present in the post list.

1. Event Calendar

9.1 When the current user has diaries, ensure that users can see their diary list with corresponding dates highlighted on the calendar.

1. Notification Functionality

10.1 When the current user has diaries, and the current date is not the diary date, ensure that users can see the diary and display how many days are left until the diary date.

10.2 When the current user has diaries, but the current date has passed the diary date, ensure that users cannot see the diary.

10.3 When the current user has no diaries, ensure that the notification bar is empty.

1. Pet Adoption

11.1 When accessing the Pet Adoption page, ensure that users can view information about adoptable pets from the petfinder API, including pet names and images, and clicking on them should navigate to the petfinder pet page.

1. Rescue Animals

12.1 When accessing the Rescue Animals page, ensure that users can view articles and can navigate to different articles using the pagination component.

1. Rescue Suggestion Reading

13.1 After the user logs in, ensure that the right sidebar displays all article titles, and clicking on a title navigates to the corresponding article.

1. Article Search

14.1 When the user's search input matches an article's title, ensure that the corresponding article titles appear, and clicking on a title navigates to the respective article.

14.2 When the user's search input does not match any article titles, ensure that no content appears.

### Back-end unit testing:

The following tables indicate only the attributes that need to be entered, and they differ.

1. User:

A total of five functions were unit tested.

1.1.sign\_up:

Test sample

| Example | userName | password | email | userType | result |
| --- | --- | --- | --- | --- | --- |
| Success | mimimi | 222 | 1234@cc.com | expert | “success” |
| userName null | null | 222 | 1234@cc.com | expert | “No transfer userName” |
| password null | mimimi | null | 1234@cc.com | expert | “No transfer password” |
| email null | mimimi | 222 | null | expert | “No transfer email” |
| userType null | mimimi | 222 | 1234@cc.com | null | “No transfer userType” |
| userName empty |  | 222 | 1234@cc.com | expert | “userName cannot be empty” |
| password empty | mimimi |  | 1234@cc.com | expert | “password cannot be empty” |
| email empty | mimimi | 222 |  | expert | “email cannot be empty” |
| userType empty | mimimi | 222 | 1234@cc.com |  | “userType cannot be empty” |
| userName repeat | mimi | 222 | 1234@cc.com | expert | “userName repeat” |
| email repeat | mimimi | 222 | 156@qq.com | expert | “email repeat” |

All cases were overwritten and passed.

1.2.sign\_in

Test sample

| Example | userName | password | email | result |
| --- | --- | --- | --- | --- |
| Success with userName | mimi | 222 | null | “success” with the user info |
| Success with email | null | 222 | 234@qq.com | “success” with the user info |
| no user | no user | 222 | nouser@qq.com | “username or email does not exist” |
| password error with userName | mimi | 2222 | null | “password error” |
| password error with email | null | 2222 | 234@qq.com | “password error” |

All cases were overwritten and passed.

1.3.log\_state\_check

Test sample

| Example | userId | result |
| --- | --- | --- |
| Success | 3790 | 1 |
| null | null | 0 |
| no userId | 510 | 0 |

All cases were overwritten and passed.

1.4.get\_userId\_by\_userName

| Example | userName | result |
| --- | --- | --- |
| Success | mimi | 3790 |
| null | null | “Cannot find the userName” |
| no userName | nouser | “Cannot find the userName” |

All cases were overwritten and passed.

1.5.log\_out\_state\_update

| Example | userId | result |
| --- | --- | --- |
| Success | 3790 | “success“” |
| null | null | “error” |
| no userId | 510 | “error” |

All cases were overwritten and passed.

1.6.Result with coverage:



The functionality of all functions is well tested

2.ExpertDialogue:

A total of two functions were unit tested.

2.1.receivedialogue

Because the empty and null cases are set in the same if condition, or we cannot set int or Integer data to "" during the test, the case of “” is not listed here.

| Example | userId | expertId | info | result |
| --- | --- | --- | --- | --- |
| Success | 3790 | 10 | test | success |
| userId null | null | 10 | test | “No transfer user” |
| expertId null | 3790 | null | test | “No transfer expert” |
| info null | 3790 | 10 | null | “No transfer info” |

All cases were overwritten and passed.

2.2.senddialogue

Again, the case of "" is not listed here.

| Example | expertId | result |
| --- | --- | --- |
| Success | 10 | the dialogue list of the expert 10 |
| expertId null | null | [],which means empty |
| no expert | 100000 | [],which means empty |

2.3.Result with coverage:



The functionality of all functions is well tested

3. PetDiary:

A total of four functions were unit tested.

3.1.add:

Test sample

| Example | petDiaryId | userId | value | reason |
| --- | --- | --- | --- | --- |
| Success | 0 | 11 | null | “successful” |
| userId null | 0 | 0 | "今天去外面" | "Error: UserId is missing" |
| value null | 0 | 11 | “ ” | "Error: Value is missing" |
| userId empty | 0 | 0 | "今天去外面" | "Error: UserId is missing" |
| value empty | 0 | 11 |  | "Error: Value is missing" |

All cases were overwritten and passed.

3.2.edit:

Test sample

| Example | petDiaryId | userId | value | reason |
| --- | --- | --- | --- | --- |
| Success | 1 | 11 | "今天去外面" | “successful” |
| value null | 1 | 11 | "" | "Error: Value is missing" |
| userId null | 1 | 0 | "今天去外面" | "Error: UserId is missing" |
| petDiaryId null | 0 | 11 | "今天去外面" | "Error: PetDiaryId is missing" |
| value empty | 1 | 11 |  | "Error: Value is missing" |
| userId empty | 1 | 0 | "今天去外面" | "Error: UserId is missing" |
| petDiaryId empty | 0 | 11 | "今天去外面" | "Error: PetDiaryId is missing" |

All cases were overwritten and passed.

3.3.delete:

Test sample

| Example | petDiaryId | userId | value | reason |
| --- | --- | --- | --- | --- |
| Success | 1 | 11 | "今天去外面" | “successful” |
| userId null | 1 | 0 | "今天去外面" | "Error: UserId is missing" |
| petDiaryId null | 0 | 11 | "今天去外面" | "Error: PetDiaryId is missing" |
| userId empty | 1 | 0 | "今天去外面" | "Error: UserId is missing" |
| petDiaryId empty | 0 | 11 | "今天去外面" | "Error: PetDiaryId is missing" |

All cases were overwritten and passed.

3.4.getPetDiariesByUserId:

Test sample

| Example | userId | result |
| --- | --- | --- |
| Success | 1 | get diaries list of this user |
| userId null | 0 | "Error: UserId is missing" |

All cases were overwritten and passed.

3.5.Result with coverage:



3.6.Result with running:



The functionality of all functions is well tested.

4.Post

A total of 3 functions were unit tested.

4.1.add:

Test sample

| Example | userName | userId | content | reason |
| --- | --- | --- | --- | --- |
| Success | “mimi” | 3790 | “This is content” | “successful” |
| userName null | null | 3790 | "This is content" | "Error: UserName is missing" |
| UserId null | “mimi” | 0 | "This is content" | "Error: UserId is missing" |
| content null | “mimi” | 3790 | null | "Error: content is missing" |
| content wrong | “xixi” | 310 | "This is content" | "Error: UserID is wrong" |

All cases were overwritten and passed.

4.2.delete:

Test sample

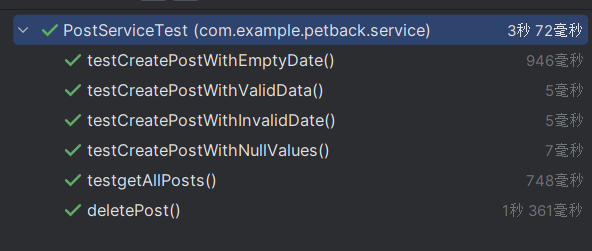
| Example | UserId | PostId | reason |
| --- | --- | --- | --- |
| Success | 3790 | 1267 | “successful” |
| userId wrong | 1 | 1267 | "Error: UserId is wrong" |
| PostId wrong | 3790 | 1 | "Error: PostId is wrong" |
| userId empty | 0 | 1267 | "Error: UserId is missing" |
| PostId empty | 3790 | 11 | "Error: PostId is missing" |

All cases were overwritten and passed.

4.3 Result with coverage:



4.4 Result with running:



The functionality of all functions is well tested.