Black Box Testing

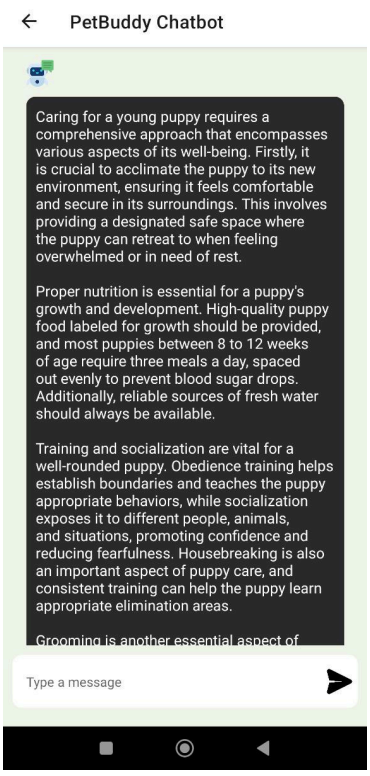
Chatbot

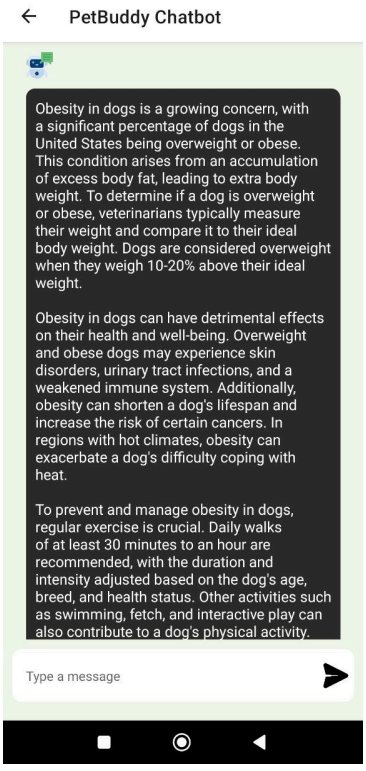
Test Technique 1: Equivalence Partitioning

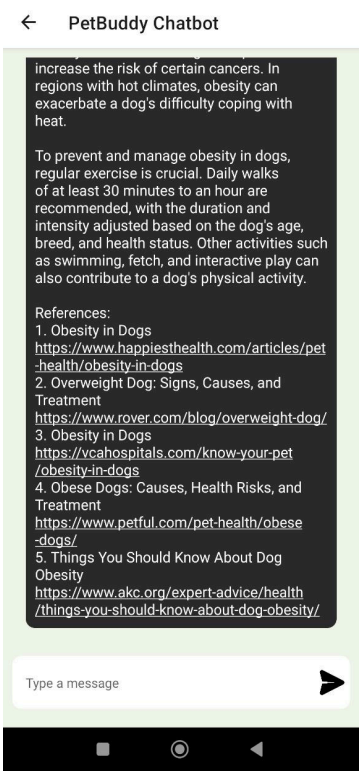
Purpose: testing the chatbot's understanding and response capabilities involves dividing user queries into distinct classes to verify that the chatbot handles each type accurately.

Using **Equivalence Partitioning**, we divide user queries into:

1. Pet Care Queries (Excluding Map Searches)
2. Non-Pet Care Queries
3. Map Location Search Queries

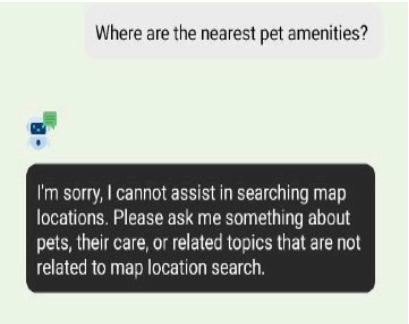
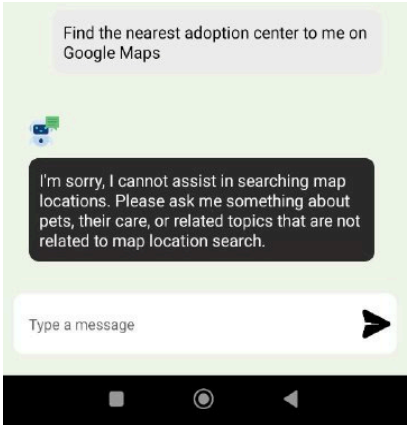
Test ID 1: Pet Care Queries (Excluding Map Searches)				
Steps	**Test Input (Prompt)**	**Expected Output**	**Actual Output**	**Test Result**
1	Tell me about caring for a young puppy	The chatbot correctly identifies and responds to pet care questions that do not involve location-based searches.		Pass

				
2	Tell me about dog obesity	The chatbot correctly identifies and responds to pet care questions that do not involve location-based searches.		Pass

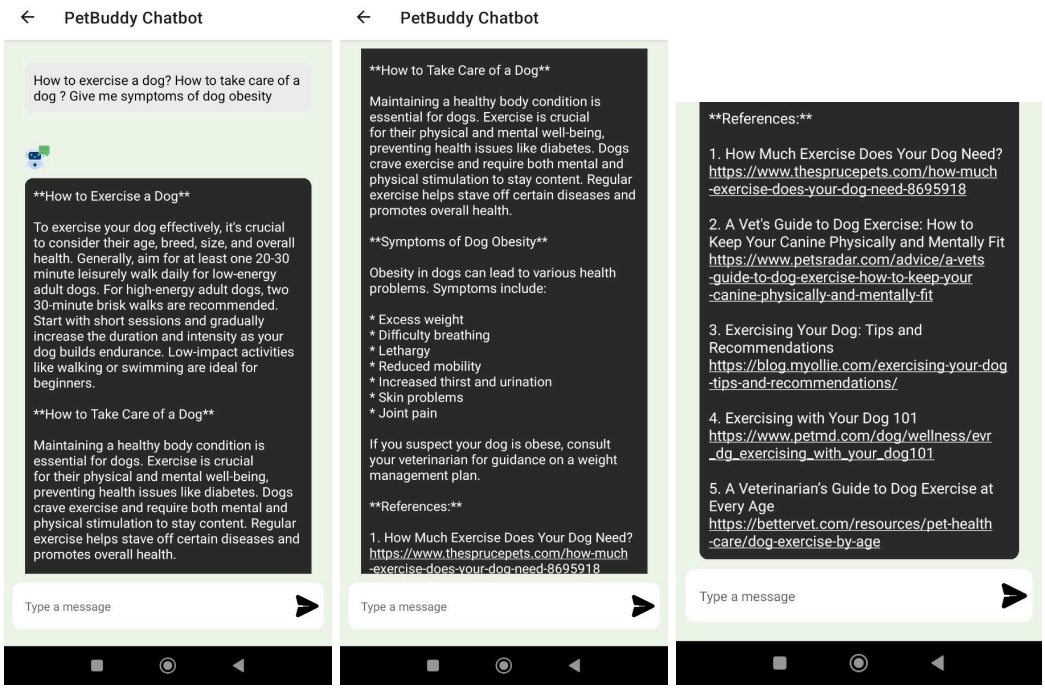
Test ID 2: Non-Pet Care Queries				
Steps	**Test Input (Prompt)**	**Expected Output**	**Actual Output**	**Test Result**
1	What is the capital of France?	“I’m sorry. I can only assist with pet care-related questions. Please ask me something about pets, their care, or related topics.”		Pass

2	How to cook a dog?	“I’m sorry. I can only assist with pet care-related questions. Please ask me something about pets, their care, or related topics.”		Pass

Test ID 3: Map Location Search Queries				
Steps	**Test Input (Prompt)**	**Expected Output**	**Actual Output**	**Test Result**
1	Where are the nearest pet amenities?	"I'm sorry, I cannot assist in searching map locations. Please ask me something about pets, their care, or related topics that are not related to map location search."		Pass
2	Find the nearest adoption center to me on Google Maps	"I'm sorry, I cannot assist in searching map locations. Please ask me something about pets, their care, or related topics that are not related to map location search."		Pass

Test Technique 2: Boundary Value Analysis

Purpose: Verifies behavior when handling input complexity at boundaries.

Test ID 4: Submit a query containing multiple questions	
Test Input (Prompt)	"How to exercise a dog? How to take care of a dog? Give me Symptoms of dog obesity?."
Expected Output	Multiple questions in one query receive relevant, separate responses to each question where possible.
Actual Output	
Test Result	Pass

Test Technique 3: State Transition Testing

Purpose: Ensures smooth transitions and proper input restrictions during backend processing.

Test ID 5: Verifies the chatbot's ability to handle different states during user interaction and backend processing. It ensures smooth state transitions, accurate responses to user inputs, and proper restrictions on additional input during processing. This case also tests the chatbot's error-handling capabilities when the backend fails.				
Steps	**Testing Steps**	**Expected Output**	**Actual Output**	**Test Result**

1	Navigate to the chatbot page and verify the initial response.	"Hi there! 🐾 I'm PetBuddy, your pet care research assistant. How can I help you and your pet today?" displayed as the initial response in the chatbot UI.	"Hi there! 🐾 I'm PetBuddy, your pet care research assistant. How can I help you and your pet today?" displayed as the initial response in the chatbot UI.	Pass
2	Enter a prompt and observe the chatbot's waiting state while processing the backend response.	After submitting a prompt, the input button becomes temporarily disabled during backend response processing.	After submitting a prompt, the input button becomes temporarily disabled during backend response processing.	Pass
3	Attempt to input another prompt during this waiting state.	Subsequent input is only allowed once the response is received	Subsequent input is only allowed once the response is received	Pass
4	Trigger API failure to reach error state.	API failure displays fallback message “Error occurred. Please try again later.”	API failure displays fallback message “Error occurred. Please try again later.”	Pass

Add Pet Form

Test Technique 1: Equivalence Partitioning

Test ID	Test Input	Expected Output	Actual Output	Test Result
1	**Name: ''** Breed: 'Golden Retriever' Weight: 25 Sex: 'M' DOB: valid past date CoatColor: Markings: MedicalCondition:	Alert: "Invalid Input", "All required fields must be filled."	Alert: "Invalid Input", "All required fields must be filled."	Pass
2	**Name: 'Princess Fluffypaws Sparkles of the Magical Puppy Kingdom'** Breed: 'Golden Retriever' Weight: 25 Sex: 'M' DOB: valid past date CoatColor: Markings: MedicalCondition:	Alert: "Invalid Input", "Name must be between 1 and 50 characters."	Alert: "Invalid Input", "Name must be between 1 and 50 characters."	Pass
3	Name: ‘Buddy’ **Breed: ''** Weight: 25 Sex: 'M' DOB: valid past date CoatColor: Markings: MedicalCondition:	Alert: "Invalid Input", "All required fields must be filled."	Alert: "Invalid Input", "All required fields must be filled."	Pass
5	Name: ‘Buddy’ **Breed: 'American Long Haired Shepherd Terrier'** Weight: 25 Sex: 'M' DOB: valid past date CoatColor: Markings: MedicalCondition:	Alert: "Invalid Input", "Breed must be between 1 and 30 characters."	Alert: "Invalid Input", "Breed must be between 1 and 30 characters."	Pass

6	Name: ‘Buddy’ Breed: 'American Long Haired Shepherd Terrier' Weight: 25 Sex: 'M' **DOB: [future date]** CoatColor: Markings: MedicalCondition:	Alert: "Invalid Input", "Date of Birth cannot be in the future."	Alert: "Invalid Input", "Date of Birth cannot be in the future."	Pass
7	Name: 'Buddy' Breed: 'Golden Retriever' **Weight: ‘heavy’** Sex: 'M' DOB: valid past date CoatColor: Markings: MedicalCondition:	"Invalid Input", "Weight must be a positive number up to 100 kg."	"Invalid Input", "Weight must be a positive number up to 100 kg."	Pass
8	Name: 'Buddy' Breed: 'Golden Retriever' **Weight: -5** Sex: 'M' DOB: valid past date CoatColor: Markings: MedicalCondition:	Alert: 'Invalid Input', Weight must be a positive number up to 100 kg.	Alert: 'Invalid Input', Weight must be a positive number up to 100 kg.	Pass
9	Name: 'Buddy' Breed: 'Golden Retriever' **Weight: 105** Sex: 'M' DOB: valid past date CoatColor: Markings: MedicalCondition:	Alert: 'Invalid Input', Weight must be a positive number up to 100 kg.	Alert: 'Invalid Input', Weight must be a positive number up to 100 kg.	Pass
10	Name: 'Buddy' Breed: 'Golden Retriever' Weight: 105 Sex: 'M' DOB: valid past date **CoatColor: ‘Dark Chocolate Brown with Golden Highlights’** Markings: MedicalCondition:	Alert: "Invalid Input", "Coat Color must be less than 30 characters."	Alert: "Invalid Input", "Coat Color must be less than 30 characters."	Pass
11	Name: 'Buddy' Breed: 'Golden Retriever' Weight: 105 Sex: 'M' DOB: valid past date CoatColor: **Markings: [Description**	Alert: "Invalid Input", "Special Markings must be less than 200 characters."	Alert: "Invalid Input", "Special Markings must be less than 200 characters."	Pass

	more than 200 words] MedicalCondition:			
12	Name: 'Buddy' Breed: 'Golden Retriever' Weight: 105 Sex: 'M' DOB: valid past date CoatColor: Markings: **MedicalCondition: [Description more than 200 words]**	Alert: "Invalid Input", "Medical Condition must be less than 200 characters."	Alert: "Invalid Input", "Medical Condition must be less than 200 characters."	Pass
13	Name: 'Buddy' Breed: 'Golden Retriever' Weight: 105 Sex: 'M' DOB: valid past date CoatColor: Markings: MedicalCondition:	Alert: 'Success', 'Pet added successfully!'	Alert: 'Success', 'Pet added successfully!'	Pass

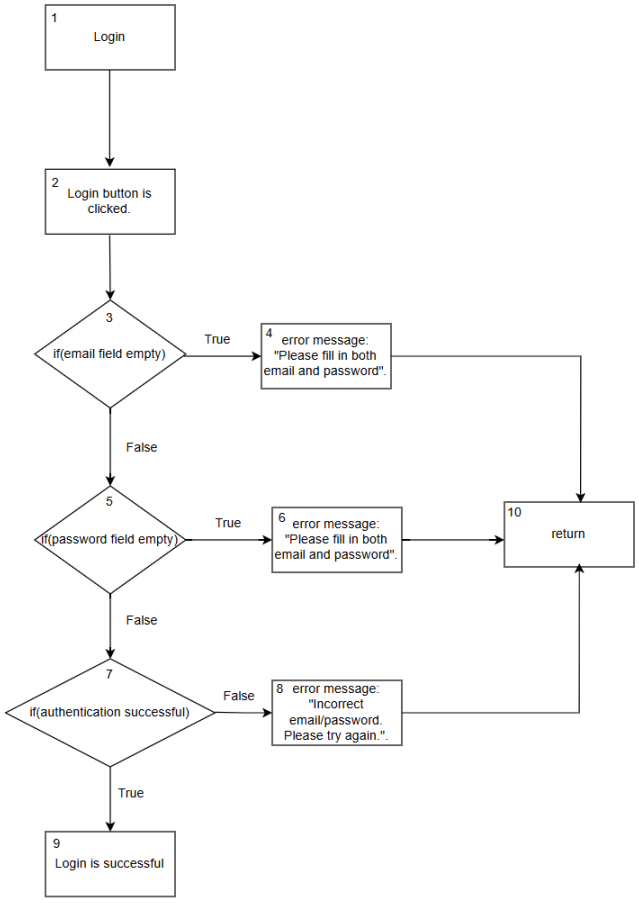
Test Technique 2: Boundary Value Analysis

Test ID	Test Input	Expected Output	Actual Output	Test Result
1	Name: 'Buddy' Breed: 'Golden Retriever' **Weight: 0** Sex: 'M' DOB: valid past date CoatColor: Markings: MedicalCondition:	Alert: Invalid Input", "Name must be between 1 and 50 characters."	Alert: Invalid Input", "Name must be between 1 and 50 characters."	Pass
2	Name: 'Buddy' Breed: 'Golden Retriever' **Weight: 1** Sex: 'M' DOB: valid past date CoatColor: Markings: MedicalCondition:	Alert: 'Success', 'Pet added successfully!'	Alert: 'Success', 'Pet added successfully!'	Pass
3	Name: 'Buddy' Breed: 'Golden Retriever' Weight: 0 Sex: 'M' **DOB: [Tomorrow's date]** CoatColor:	Alert: "Invalid Input", "Date of Birth cannot be in the future."	Alert: "Invalid Input", "Date of Birth cannot be in the future."	Pass

	Markings: MedicalCondition:			
4	Name: 'Buddy' Breed: 'Golden Retriever' Weight: 0 Sex: 'M' **DOB: [Today's date]** CoatColor: Markings: MedicalCondition:	Alert: 'Success', 'Pet added successfully!'	Alert: 'Success', 'Pet added successfully!'	Pass

White Box Testing

Login



Cyclomatic Complexity
|decision points| + 1 = 4

Basic Paths
1. Baseline path: 1,2,3,5,7,9
2. Basic path: 1,2,3,4,10
3. Basic path: 1,2,3,5,6,10
4. Basic path: 1,2,3,5,7,8,10

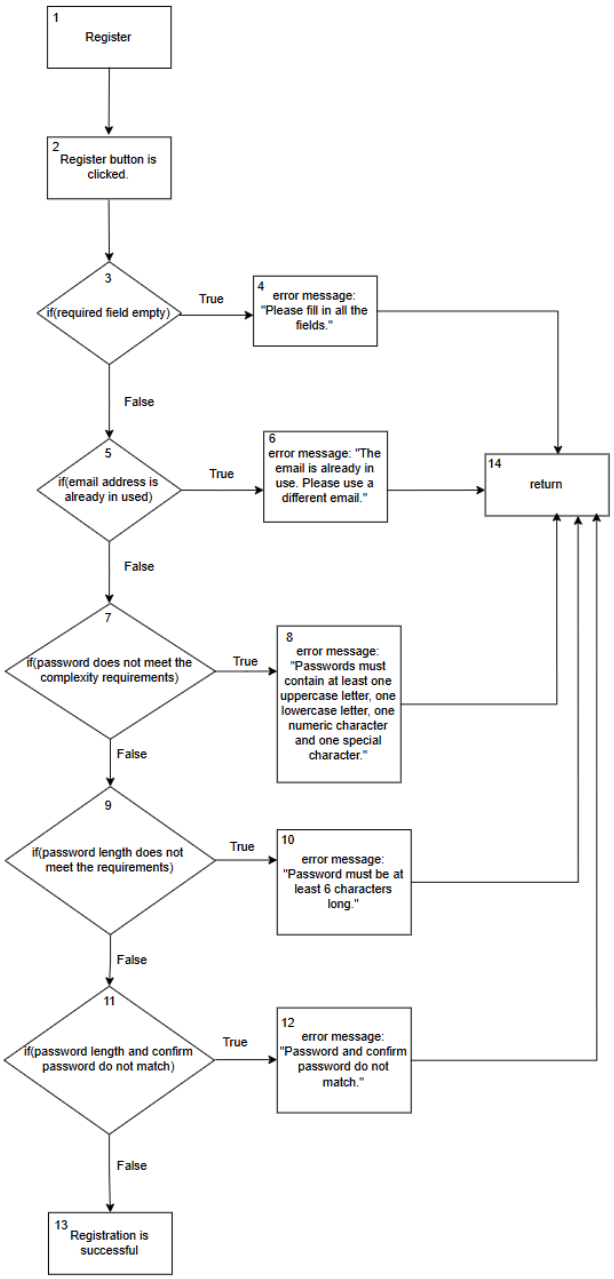
Test Cases
1. email : "ChuaMin@gmail.com " ; password: "123Aac."
2. email : " " ; password: "123Aac."
3. email : "ChuaMin@gmail.com " ; password: " "
4. email : "ChuaMin@gmail.com " ; password: "123AAc."

Real Execution Paths
1. Baseline path: 1,2,3,5,7,9
2. Basic path: 1,2,3,4,10
3. Basic path: 1,2,3,5,6,10
4. Basic path: 1,2,3,5,7,8,10

Test Id	Path	Inputs	Expected Output	Actual output
1	1,2,3,5,7,9	**Email** "ChuaMin@gmail.com " **Password** "123Aac."	System displays a successful login message.	System displays a successful login message.
2	1,2,3,4,10	**Email** " " **Password** "123Aac."	System displays an error message: "Please fill in both email and password". To the user.	System displays an error message: "Please fill in both email and password". To the user.
3	1,2,3,5,6,10	**Email**	System displays an error	System displays an

		"ChuaMin@gmail.com " **Password** " "	message: "Please fill in both email and password". To the user.	error message: "Please fill in both email and password". To the user.
4	1,2,3,5,7,8,10	**Email** "ChuaMin@gmail.com " **Password** "123AAc."	System displays an error message: "Incorrect email/password. Please try again." to the user.	System displays an error message: "Incorrect email/password. Please try again." to the user.

Register



Cyclomatic Complexity
|decision points| + 1 = 6

Basic Paths
1. Baseline path: 1,2,3,5,7,9,11,13
2. Basic path: 1,2,3,4,14
3. Basic path: 1,2,3,5,6,14
4. Basic path: 1,2,3,5,7,8,14
5. Basic path: 1,2,3,5,7,9,10,14
6. Basic path: 1,2,3,5,7,9,11,12,14

Test Cases
1. name: "ChuaShermaine" ; email : "ChuaMin9@gmail.com " ; password: "123Aac.";
confirm password: "123Aac."
2. name: " " ; email : "ChuaMin9@gmail.com " ;
password: "123Aac.";
confirm password: "123Aac."
3. name: " " ; email : "ChuaMin@gmail.com " ;
password: "123Aac.";
confirm password: "123Aac."
4.name: " " ; email : "ChuaMin9@gmail.com " ;
password: "123Aaa";
confirm password: "123Aaa"
5.name: " " ; email : "ChuaMin9@gmail.com " ;
password: "123Aa";
confirm password: "123Aa"
6.name: " " ; email : "ChuaMin9@gmail.com " ;
password: "123Aac.";
confirm password: "123Aaf."

Real Execution Paths
1. Baseline path: 1,2,3,5,7,9,11,13
2. Basic path: 1,2,3,4,14
3. Basic path: 1,2,3,5,6,14
4. Basic path: 1,2,3,5,7,8,14
5. Basic path: 1,2,3,5,7,9,10,14
6. Basic path: 1,2,3,5,7,9,11,12,14

Test Id	Path	Inputs	Expected Output	Actual output
1	1,2,3,5,7,9,11,13	**Name** “ChuaShermaine” **Email** "ChuaMin9@gmail.com " **Password**	System displays a successful registration message.	System displays a successful registration message.

		"123Aac." **Confirm Password** "123Aac."		
2	1,2,3,4,14	**Name** “ ” **Email** "ChuaMin9@gmail.com " **Password** "123Aac." **Confirm Password** "123Aac."	System displays an error message: "Please fill in all the fields.” to the user.	System displays an error message: "Please fill in all the fields.” to the user.
3	1,2,3,5,6,14	**Name** “ChuaShermaine” **Email** "ChuaMin@gmail.com " **Password** "123Aac." **Confirm Password** "123Aac."	System displays an error message: "The email is already in use. Please use a different email.“ to the user.	System displays an error message: "The email is already in use. Please use a different email.” To the user.
4	1,2,3,5,7,8,14	**Name** “ChuaShermaine” **Email** "ChuaMin9@gmail.com " **Password** "123Aaa" **Confirm Password** "123Aaa"	System displays an error message: "Passwords must contain at least one uppercase letter, one lowercase letter, one numeric character and one special character. “ to the user.	System displays an error message: "Passwords must contain at least one uppercase letter, one lowercase letter, one numeric character and one special character.” to the user.
5	1,2,3,5,7,9,10,14	**Name** “ChuaShermaine” **Email** "ChuaMin9@gmail.com " **Password** "123Aa" **Confirm Password** "123Aa"	System displays an error message: "Password must be at least 6 characters long." to the user.	System displays an error message: "Password must be at least 6 characters long." to the user.
6	1,2,3,5,7,9,11,12,14	**Name** “ChuaShermaine” **Email** "ChuaMin9@gmail.com " **Password** "123Aac." **Confirm Password** "123Aaf."	System displays an error message: "Password and confirm password do not match." to the user.	System displays an error message: "Password and confirm password do not match." to the user.

Demo Script

Introduction [Jia Chi]
Good morning everyone, I am Jia Chi. We are from group 10 and my team members are Jamine, Cherng Khai, Jia Qi and Chun Wen.

Today, our team would like to present our product, called PetCare. Before diving into the product, let me introduce the user story of our project. [read slides]. After discussion and research, we decided to propose a mobile application that aims to create a seamless and user-friendly platform for pet owners, adoption centers and animal lovers. The primary goal of PetCare is to strengthen the bond between pet owners and their furry companions by offering a comprehensive suite of tools. Whether it is for adopting new pets, viewing nearby petcare amenities or managing pets records, the application provides an effective solution for all the stakeholders involved.

Key Features [Jia Chi]
These are the key features of the PetCare application. I will explain in detail now. User authentication and personal information include register,login,reset password,view and edit personal account as well as logout. For pet profile and articles there are view,edit,add and delete pet. For articles there are view,add,edit,delete,search and filter articles. Moving on to report missing pets there are display active missing pets, add sighting of selected missing pets, mark missing pet as found as well as add missing pet. As for nearby pet amenities there are search and filter for pet care amenities. Finally, there is also a chatbot in our Pet Care app.

This is the UI for register,login and reset password, user profile, chatbot,home page,pet info, pet adoption for user view as well as pet adoption view for adoption center, nearby petcare amenities,missing pet and article.

I will now handover the time to Cherng Khai.

Tech Stack [Cherng Kai]
Thank you, Jia Chi. Here are the framework and libraries used in the development of our product to ensure a robust and seamless experience. On the frontend, we use Expo for easy deployment on both Android and iOS platforms, with TypeScript and React Native handling the user interface.

For the backend, the core application logic is developed in **Java**, utilizing **Maven** for dependency management. Spring framework, a powerful and flexible framework is adopted as it reduces the amount of repetitive code required using built on features such as @Data annotation for getters and setters. The external apis called by our Java services include GoogleMaps API to locate nearby pet care amenities, and Nomination API to convert a postal code to an address.

Our mobile app integrates Firebase for essential backend services, including real-time data management with Cloud Firestore, image storage with Cloud Storage, and secure user management via Firebase Authentication.

The chatbot feature is developed in our Python backend, where we use FastAPI, a high-performance web framework to build the APIs that connect to our user interface. The chatbot functionality is powered by integrating the Tavily and Gemini APIs using LangChain framework.

Feature Highlight [Cherng Khai]

I will now pass the time to Jia Qi.

Slide: Feature Highlights

Script:

"Our AI-powered PetCare Chatbot combines multiple resources to provide reliable pet care guidance and app navigation assistance.

Here's how it works:

1. For pet care questions, the chatbot uses the Tavily search engine to pull relevant information from online sources. For app navigation, it retrieves information directly from a PDF file with app documentation.
2. All information is processed through Google Gemini, an advanced language model that generates clear, detailed responses.
3. Finally, the AI PetCare Chatbot presents these responses to the user, offering guidance on both pet care and app features.

Together, these components make our chatbot a versatile tool for pet owners, providing both expertise and ease of use."

Good Traceability [Jia Qi]

Thank you, Cherng Khai. To illustrate the traceability of our project features, I will use the "Add Pet" use case as an example. This function allows users to add a new pet via the Add Pet Form, complete with form validation. The key entity classes involved are the Pet and User Models. In this setup, the User object is stored within the Pet object to track pet ownership. An owner can have zero to multiple pets, while each pet belongs to a specific owner. The PetController serves as the bridge between the boundary class and the entity classes. The boundary class for adding a pet is the AddNewPetUI, which consists of the form for adding new pets.

The functionality flow is depicted in a sequence diagram. Upon form submission, the inputs are validated on the UI before data is sent to the backend, creating a loop until the inputs meet the required format. We will demonstrate the actual flow in our live demo later.

Testing

For functionality testing, we employ equivalence and boundary value testing. For equivalence testing, if the name field is left empty, an alert will display on the user's screen, preventing form submission. In boundary value testing, we check three values that are smaller than the valid input, as well as the valid case.

Good SD Practices [Jia Qi]
Our team has adopted several best practices in developing PetCare to ensure reliability, maintainability, and scalability. Firstly, we utilize Git version control for effective collaboration, tracking changes, and managing code across different versions. Robust error handling and logging are implemented through try-catch blocks in both our frontend and backend.

Multi-Layered Architecture [Jia Qi]
Next, we follow modular coding practices using a multi-layered architecture, extending the traditional MVC pattern by adding the Service Layer and Repository Layer. These additional layers enhance the separation of concerns.

For instance, in our Pet Information feature:

- The PetController connects the user interface with backend services to facilitate data transfer and service calls.
- The PetService manages all application logic, ensuring that business rules are enforced properly.
- The PetRepository handles interactions between the backend and the database.
- Lastly, the Model layer defines and organizes class attributes, keeping data structures well-structured.

Single Responsibility Principle [Jia Qi]
Additionally, we adhere to the Single Responsibility Principle, ensuring that each class is responsible for a single functionality. This minimizes the risk of unintended changes. If a class handles multiple responsibilities, modifying one functionality could inadvertently impact unrelated parts. Thus, this principle is crucial for improving code maintainability, making future refactoring and extensions easier.

Data Transfer Object [Jia Qi]
We also utilize Data Transfer Objects (DTOs), allowing us to modify how data is transferred without affecting core business logic. If the data structure changes, we only need to adjust the DTO and its population process, not the underlying entities.

DTOs optimize data transfer by excluding unnecessary fields or combining multiple fields into a single object before sending it to the frontend. This approach is particularly useful for reducing payload sizes in web services or APIs.

Now, I will pass the time to Jasmine.

Live Demo [Jasmine and Chun Wen]
[Jasmine]
Thank you, Jia Qi. Now, I'll guide you through the live demo of our product, PetCare.

First, when a user opens the app, they'll see the login screen. If you're a new user, you can click 'Register' to create a new account by entering your details. For today's demonstration, we'll use a pre-created account that has multiple pets and articles set up, so we can showcase the features.

Once logged in, you'll be greeted by the Home Page. Let's walk through the features one by one. At the top, you'll see the My Pet section, which lists all your added pets. Clicking on a pet's image brings up a pet info modal displaying the pet's details. You can easily update any information by clicking the 'Edit' button, making changes, and then saving. You'll immediately see the updated details reflected. If you wish to remove a pet, click on the pet image and then tap the 'Delete' button.

Next, we have the Pet Articles section, where you can read helpful content about pet care and wellness. An article carousel is displayed which auto-rotates every 10 seconds and displays only the 5 latest articles. Tapping on the "See All" button would redirect the user to the Browse Articles page where users are able to browse through all pet-care related articles posted by other users of PetCare. Additionally, users are able to filter and search articles to further enhance their experience. Tapping on the "Posted" section would bring users to the "Browse Posted Articles" page where they are able to add, edit and delete articles.

Moving on to the Missing Pet feature which allows users to report their pets as missing and share sightings of lost pets in their area. PetCare will prompt the user to enable location access, allowing the app to display their current location on an interactive map. The map will feature markers indicating the locations where pets have been reported lost, providing a visual aid for users to identify nearby cases. By tapping on a specific marker, users are redirected to the missing pet's dedicated page, which includes essential information about the pet and their owner. This information is crucial for connecting potential finders with the rightful owners. A separate map interface showcases sightings of the selected missing pet. Users can navigate through reported sightings by tapping on the "->" arrow, which highlights the next sighting and automatically centers the map on that location. As users move through the sightings, the information displayed will update accordingly.

Pet owners are able to report their pet as missing by tapping on the plus icon where they will be redirected to the Missing Pet Report form. The system will display a list of the user's registered pets. The pet owner will need to select one of their registered pets as missing and pet information such as name and age will be auto-populated in the form. The pet owner will be required to input details such as last seen postal code, date, time, description and also last seen image. Once a report is made, the pet owner waits for other users to report sightings of their missing pet.

When the missing pet has been found, the pet owner toggles the Found toggle switch to mark their pet as found and subsequently remove the pet from view in the missing pet page.

If your pet needs grooming or medical attention, our Nearby Petcare Amenities feature has you covered. Similar to the Missing Pet feature, the system would prompt users for location access if not already granted. Nearby petcare amenities like groomers and vets are displayed as markers on the map interface. Tapping on a listing would open the amenity modal which displays its full details. The approximated distance of amenity from user was also calculated, which is helpful for users to see which amenity is closer. Users are also able to search and filter through the amenities to further refine their search. Users are also able to expand the map to have an expanded view of all the markers in the map by tapping on the expand icon.

Tapping on a listing shows the full detail of an amenity. You can also use the map to see the exact location. Clicking a marker will also display the amenity's details. We also calculated the distance of the amenity from your current location, which is helpful when you are looking for an amenity to visit. PetCare even supports filtering and sorting. By clicking the 'Filter' button, you'll see options to refine your search. For example, let's select [choose an option], and here are the filtered results. If you prefer to sort by rating or distance, just tap the sort icon and select your preference. Let's sort by rating to see the highest-rated locations.

Next, I'll be passing the time over to Chun Wen.

[Chun Wen]
AI PetCare Chatbot is a virtual assistant designed to support pet owners by providing instant, reliable guidance on pet care and app navigation. Users can easily ask questions about their pet's health, diet, behavior, and more, while also receiving guidance on how to use the app's features. PetBuddy combines both pet care support and app assistance in a single interface, creating a seamless experience for users.

If you have a question about your pet, our chatbot is here to help. Our chatbot serves as a virtual assistant designed to offer instant guidance on pet care topics and can even save you time and money on vet consultations if the issue is minor.
Users can type questions or prompts about their pet's health, diet, behavior,and more into the chat interface, and PetBuddy provides tailored reports based on the relevant pet care resources from online sources to help users make informed decisions about their pet's well-being. For non-pet-care queries, PetBuddy gently reminds users that it only handles pet-related topics.

For questions about using the app, our chatbot offers clear, easy-to-follow guidance. Whether it's finding nearby pet amenities, learning how to manage pet records, or exploring adoption options, PetBuddy makes app navigation simple.

For pet lovers who believe one pet is never enough, or for those who want to help unadopted pets, our app offers a simple and convenient Adoption feature. You can browse a wide variety of pets up for adoption with just a few clicks. For each and every pet you see over here are added by different shelters. In order to learn more about pets. You just have to click on the cardview and there will be a pop out modal. To show your interest , you just have to press the book appointment and it will direct you to whatsapp and you can start your conversation from there.

For the adoption center to post their listing . You just have to click on the add icon over here input the details and wait for prompt. To edit the pet information, i,e age or name. Clicked here. In the scenario where the pet have been adopted, you just have to delete them from the page.

Lastly, by clicking on the icon here you are able to view your personal info, you can edit your details such as your profile name, photo, address or phone number..

Future Improvement [Chun Wen]
What a comprehensive application! However, we've identified a few areas for improvement that could be implemented in the future.

Firstly, we aim to reduce page loading times. Since our application has not yet been deployed to the app store, we anticipate that once it's live, optimizing performance will be a top priority. This includes refining our backend queries and caching strategies to ensure a smoother user experience.

We're also planning to introduce a Pet Calendar feature to help pet owners manage and keep track of their pets' schedules. This calendar will allow users to set reminders for important activities, such as vet appointments, vaccination dates, grooming sessions, and even feeding times.

Next, with the PetBuddy chatbot, we can customize it to not only answer pet care questions but also guide users on how to effectively use our app. Here's how it works:
When a user asks a question about using the app, the chatbot performs the following steps:In our backend together with the langchain framework, we can use a PDF loader function to load the pages from the documentation, which are then transformed into embeddings using the **Google Generative AI Embeddings** model.
These embeddings are then stored in a vector store, allowing the chatbot to perform a similarity search to retrieve relevant information related to the user's query. Then, the retrieved information is sent to the **Google Gemini API**, which processes it and generates a response to answer the user's query effectively to guide the user on how to use our app.

Finally, we'll consider expanding integration with more third-party services to offer even richer functionality, such as real-time vet consultations and automated pet supply reordering. These improvements will continue to evolve PetCare into an even more powerful and user-friendly tool for pet owners. Here's the sample code:

With that, we have come to the end of our presentation, thank you.