User manual and Testing plan

WigglePaws

a matching platform for pet owners and pet sitters

Prepared for
Assoc. Prof. Duangdao Wichadakul, Ph. D.

By G.2 Muse Team

6330132621	Natapat Klaewkla	User manual
6330216721	Thanadol Chitthamlerd	Testing plan
6331301371	Kawin Moongsamankul	User manual
6331310121	Thanachote Katanyutapant	User manual
6331312271	Thut Thamrongnawasawat	User manual
6331313021	Tuchtham Sungkameka (Leader)	Overall document organization
6331316021	Thuwa Ngeonsalung	Testing plan
6331323321	Pannatorn Sirasuk	Testing plan
6332026371	Sirawitch Tiyasuttipun	Testing plan

2110336 Software Engineering II, Semester 2 of Academic year 2022
Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University
April 21th, 2023

Table of Contents

Black Box Testing Plan..... 1
Function: Pet Owner Register..... 1
Input Equivalence Classes..... 1
Test Cases..... 3
Function: Searching..... 16
Input Equivalence Classes..... 16
Test Cases..... 16
Function: Booking..... 21
Input Equivalence Classes..... 21
Test Cases..... 22
Function: Chatting..... 26
Input Equivalence Classes..... 26
Test Cases..... 27
Test Coverage Matrix..... 29
White-box Test Plan..... 30
User Manual..... 35
Function: Register..... 35
Role : Pet Owner..... 35
Role : Pet Sitter..... 37
Function: Login..... 40
Role : User..... 40
Function: Profile..... 42
Role: User..... 42
Function: Post..... 46
Role: Pet owner..... 46
Role: Pet sitter..... 47
Function: Searching..... 51
Role: Pet owner..... 51
Function: Booking..... 53
Role : Pet owner..... 53
Function: Chatting..... 54
Role : Pet owner..... 54
Role : Pet sitter..... 57
Function: Review..... 58

Role : Pet owner..... 58
Function: Reporting Problems..... 60
Role : User..... 60
Role : Admin..... 64
Limitations of the Implementation..... 68
Register and Login..... 68
Matching..... 68
Schedule..... 68
Booking..... 68
Review..... 68
Guest..... 68
Chat..... 69
Payment..... 69
Support..... 69
Deployment..... 69
Soft copy of all documents..... 70
SRS Report..... 70
Product Backlog and Sprint1 Backlog..... 70
Sprint 1 Burndown Chart and Sprint 2 Backlog..... 70
Sprint 2 Burndown Chart and Sprint 3 Backlog..... 70
Sprint 3 Burndown chart..... 70
Design document..... 70

Black Box Testing Plan

Function: Pet Owner Register

Input Equivalence Classes

Equivalent Class ID	Input	Validation
1	username is shorter than 5 characters	invalid
2	username exceeds 20 characters	invalid
3	username's length is between 5 and 20 characters	valid
4	username is not alphanumeric	invalid
5	username is alphanumeric	valid
6	username is not in the database	valid
7	username is in the database	invalid
8	password is shorter than 8 characters	invalid
9	password exceeds 30 characters	invalid
10	password is between 8 and 30 characters	valid
11	email exceeds characters	invalid
12	email is in a valid email format (XXX@XXX.XXX without special symbols)	valid
13	email is not in a valid email format (XXX@XXX.XXX without special symbols)	invalid
14	email is in the database	invalid
15	email is not in the database	valid
16	phone number is shorter than 9 digits	invalid
17	phone number exceeds 10 digits	invalid

18	phone number is between 9 and 10	valid
19	phone number only consists of numbers	valid
20	phone number does not only consist of numbers	invalid
21	address is an empty string	valid
22	address exceeds 255 characters	invalid
23	address is between 0 and 255 characters	valid
24	firstname is shorter than 1 character	invalid
25	firstname exceeds 30 characters	invalid
26	firstname is beween 1 and 30 characters	valid
27	firstname is alphabetic	valid
28	firstname is not alphabetic	invalid
29	lastname is shorter than 1 character	invalid
30	lastname exceeds 30 characters	invalid
31	lastname is between 1 and 30 characters	valid
32	lastname is alphabetic	valid
33	lastname is not alphabetic	invalid
34	petType is empty	valid
35	user has more than 10 pet types	invalid
36	user doesn’t have more than 10 pet types	valid

Test Cases

Assume that

- This user exists in the database
    - username: “uOwner123”
    - email: “eOwner123@gmail.com”
- This user does not exist in the database
    - username: “boy123”
    - email: “boy123@hotmail.com”

**** This is the chosen function for automated testing with cypress ****

Test Case ID	TC1-1	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Valid	
Actual Output	Valid	
Classes Covered	3, 5, 6, 10, 12, 15, 18, 19, 23, 26, 27, 31, 32, 36	

Test Case ID	TC1-2	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“Boy”

	lastname	“Testing”
	phone number	“0818118111”
	address	“”
	petType	[“Dog”, “Cat”]
Expected Output	Valid	
Actual Output	Valid	
Classes Covered	3, 5, 6, 10, 12, 15, 18, 19, 21, 26, 27, 31, 32, 36	

Test Case ID	TC1-3	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[]
Expected Output	Valid	
Actual Output	Valid	
Classes Covered	3, 5, 6, 10, 12, 15, 18, 19, 23, 26, 27, 31, 32, 34, 36	

Test Case ID	TC1-4 username is shorter than 5 characters	
Input(s)	username	“A123”
	password	“Boy123_”

	email	“boy123@hotmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Valid	
Classes Covered	1, 5, 6, 10, 12, 15, 18, 19, 23, 26, 27, 31, 32, 36	

Test Case ID	TC1-5 username exceeds 20 characters	
Input(s)	username	“a1b2c3d4e5f6g7h8i91011”
	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Valid	
Classes Covered	2, 5, 6, 10, 12, 15, 18, 19, 23, 26, 27, 31, 32, 36	

Test Case ID	TC1-6 username is not alphanumeric	
Input(s)	username	“boy@”

	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Valid	
Classes Covered	3, 4, 6, 10, 12, 15, 18, 19, 23, 26, 27, 31, 32, 36	

Test Case ID	TC1-7 username is in the database	
Input(s)	username	“uOwner123”
	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Valid	
Classes Covered	3, 5, 7, 10, 12, 15, 18, 19, 23, 26, 27, 31, 32, 36	

Test Case ID	TC1-8 password is shorter than 8 characters

Input(s)	username	“boy123”
	password	“Boy”
	email	“boy123@hotmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Valid	
Classes Covered	3, 5, 6, 8, 12, 15, 18, 19, 23, 26, 27, 31, 32, 36	

Test Case ID	TC1-9 password exceeds 30 characters	
Input(s)	username	“boy123”
	password	“a1b2c3d4e5f6g7h8i910111213126366”
	email	“boy123@hotmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Valid	
Classes Covered	3, 5, 6, 9, 12, 15, 18, 19, 23, 26, 27, 31, 32, 36	

Test Case ID	TC1-10 email exceeds characters	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“a1b2c3d4e5f6g7h8i9a1b2c3d4e5f6g7h8i9a1b2c3d4@hotmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Valid	
Classes Covered	3, 5, 6, 10, 11, 15, 18, 19, 23, 26, 27, 31, 32, 36	

Test Case ID	TC1-11 email is not in a valid email format (XXX@XXX.XXX without special symbols)	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“boy123hotmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	

Actual Output	Invalid
Classes Covered	3, 5, 6, 10, 13, 15, 18, 19, 23, 26, 27, 31, 32, 36

Test Case ID	TC1-12 email is in the database	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“eOwner123@gmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	3, 5, 6, 10, 12, 14, 18, 19, 23, 26, 27, 31, 32, 36	

Test Case ID	TC1-13 phone number is shorter than 9 digits	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“081811”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]

Expected Output	Invalid
Actual Output	Invalid
Classes Covered	3, 5, 6, 10, 12, 15, 16, 19, 23, 26, 27, 31, 32, 36

Test Case ID	TC1-14 phone number exceeds 10 digits	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“08181181111234”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	3, 5, 6, 10, 12, 15, 17, 19, 23, 26, 27, 31, 32, 36	

Test Case ID	TC1-15 phone number does not only consist of numbers	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“081811811A”
	address	“Bangkok”

	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	3, 5, 6, 10, 12, 15, 18, 20, 23, 26, 27, 31, 32, 36	

Test Case ID	TC1-16 address exceeds 255 characters	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	3, 5, 6, 10, 12, 15, 18, 19, 22, 26, 27, 31, 32, 36	

Test Case ID	TC1-17 firstname is shorter than 1 character	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“”
	lastname	“Testing”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	3, 5, 6, 10, 12, 15, 18, 19, 23, 24, 27, 31, 32, 36	

Test Case ID	TC1-18 firstname exceeds 30 characters	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“BoyBoyBoyBoyBoyBoyBoyBoyBoy”
	lastname	“Testing”

	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	3, 5, 6, 10, 12, 15, 18, 19, 23, 25, 27, 31, 32, 36	

Test Case ID	TC1-19 firstname is not alphabetic	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“Boy123”
	lastname	“Testing”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	3, 5, 6, 10, 12, 15, 18, 19, 23, 26, 28, 31, 32, 36	

Test Case ID	TC1-20 lastname is shorter than 1 character	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“Boy”

	lastname	“”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	3, 5, 6, 10, 12, 15, 18, 19, 23, 26, 27, 29, 32, 36	

Test Case ID	TC1-21 lastname exceeds 30 characters	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“Boy”
	lastname	“TestingTestingTestingTesting TestingTesting”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	3, 5, 6, 10, 12, 15, 18, 19, 23, 26, 27, 30, 32, 36	

Test Case ID	TC1-22 lastname is not alphabetic	
Input(s)	username	“boy123”

	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“Boy”
	lastname	“Testing123”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”]
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	3, 5, 6, 10, 12, 15, 18, 19, 23, 26, 27, 31, 33, 36	

Test Case ID	TC1-23 `user has more than 10 pet types`	
Input(s)	username	“boy123”
	password	“Boy123_”
	email	“boy123@hotmail.com”
	firstname	“Boy”
	lastname	“Testing”
	phone number	“0818118111”
	address	“Bangkok”
	petType	[“Dog”, “Cat”, “Goldfish”, “Lizard”, “Snake”, “Panda”, “Bear”, “Goldfish”, “Hamster”, “Wolf”, “Rait”]
Expected Output	Invalid	
Actual Output	Invalid	

Classes Covered	3, 5, 6, 10, 12, 15, 18, 19, 23, 26, 27, 31, 32, 35

Function: Searching

Input Equivalence Classes

Equivalent Class ID	Input	Validation
1	searchName is an empty string	valid
2	searchName is a substring of an existing sitter's name	valid
3	searchName is not empty and is not a substring of any existing sitter's name	valid
4	searchPetTypes is an empty array	valid
5	searchPetTypes is not an empty array	valid
6	searchPriceMin or searchPriceMax is empty	invalid
7	searchPriceMin and searchPriceMax are not empty but searchPriceMin or searchPriceMax is not a float	invalid
8	searchPriceMin, searchPriceMax are floats and (searchPriceMin < 0)	invalid
9	searchPriceMin, searchPriceMax are floats and (searchPriceMin > searchPriceMax)	invalid
10	searchPriceMin, searchPriceMax are floats and (searchPriceMax >= 10000)	invalid
11	searchPriceMin, searchPriceMax are floats and (0 <= searchPriceMin <= searchPriceMax < 10000)	valid

Test Cases

Assumes that all test cases will search for both Pet Hotels and FreelancePetSitter and sort by avgRating.

Test Case ID	TC2-1: Empty searchName, empty searchPetTypes array, valid searchPriceRange	
Input(s)	searchName	“”
	searchPetTypes	[]
	searchPriceMin	250
	searchPriceMax	500
Expected Output	Valid	
Actual Output	Valid	
Classes Covered	1,4,11	

Test Case ID	TC2-2: searchName is a substring of an existing sitter's name, searchPetTypes is a non-empty array, valid searchPriceRange	
Input(s)	searchName	“Tom”
	searchPetTypes	["Cat", "Dog"]
	searchPriceMin	250
	searchPriceMax	500
Expected Output	Valid	
Actual Output	Valid	
Classes Covered	2,5,11	

Test Case ID	TC2-3: searchName is NOT a substring of an existing sitter's name, searchPetTypes is an empty array, valid searchPriceRange	
Input(s)	searchName	“aowienfoiawen”
	searchPetTypes	[]
	searchPriceMin	250
	searchPriceMax	500
Expected Output	Valid	
Actual Output	Valid	
Classes Covered	3,4,11	

Test Case ID	TC2-4: searchPriceMin or searchPriceMax is empty	
Input(s)	searchName	“”
	searchPetTypes	[]
	searchPriceMin	(empty)
	searchPriceMax	500

Expected Output	Invalid
Actual Output	Invalid
Classes Covered	1,4,6

Test Case ID	TC2-5: searchPriceMin or searchPriceMax is not a float	
Input(s)	searchName	""
	searchPetTypes	[]
	searchPriceMin	a
	searchPriceMax	200
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	1,4,7	

Test Case ID	TC2-6: searchPriceMin < 0	
Input(s)	searchName	""
	searchPetTypes	[]
	searchPriceMin	-2
	searchPriceMax	200
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	1,4,8	

Test Case ID	TC2-7: searchPriceMin > searchPriceMax	
Input(s)	searchName	""

	searchPetTypes	[]
	searchPriceMin	2000
	searchPriceMax	500
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	1,4,9	

Test Case ID	TC2-8: searchPriceMax >= 10000	
Input(s)	searchName	""
	searchPetTypes	[]
	searchPriceMin	200
	searchPriceMax	500000
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	1,4,10	

Function: Booking

Input Equivalence Classes

Equivalent Class ID	Input	Validation
1	Start Date is empty	invalid
2	Start Date is before current time	invalid
3	Start Date is after current time	valid
4	End Date is empty	invalid
5	End Date is before Start Date	invalid
6	End Date is after Start Date	valid
7	Not select any pet	invalid
8	Select at least one pet	valid
9	Total Price is empty	invalid
10	Total Price is negative	invalid
11	Total Price is equal or more than 0	valid
12	Note is empty	valid
13	Note is not empty	valid

Test Cases

Assume that

- The user is logged-in as account that registered as Pet Owner
- The user is at booking page of any Pet Sitter
- The user has at least one pet

Test Case ID	TC3-1 `All fields have appropriate input with no note`	
Inputs	Start Date	Current Date + 1 Day
	End Date	Start Date + 1 Day
	Selected Pet	Scooby-Doo
	Total Price	300
	Note	<Empty String>
Expected Output	Valid	
Actual Output	Valid	
Classes Covered	3, 6, 8, 11, 13	

Test Case ID	TC3-2 `All fields have appropriate input with note`	
Inputs	Start Date	Current Date + 1 Day
	End Date	Start Date + 1 Day
	Selected Pet	Scooby-Doo
	Total Price	300
	Note	Please don't feed my dog chocolate
Expected Output	Valid	
Actual Output	Valid	
Classes Covered	3, 6, 8, 11, 12	

Test Case ID	TC3-3 `Start date is empty`	
Inputs	Start Date	*<None>*
	End Date	Current Date + 2 Day
	Selected Pet	Scooby-Doo
	Total Price	300
	Note	Feed my dog Scooby Snacks!
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	1, 6, 8, 11, 13	

Test Case ID	TC3-4 `End date is empty`	
Inputs	Start Date	Current Date + 2 Day
	End Date	*None*
	Selected Pet	Scooby-Doo
	Total Price	300
	Note	Feed my dog Scooby Snacks!
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	3, 4, 8, 11, 13	

Test Case ID	TC3-5 `Start date is before current time`	
Inputs	Start Date	Current Date - 1 Hour
	End Date	Current Date + 2 Day
	Selected Pet	Scooby-Doo
	Total Price	300
	Note	Feed my dog Scooby Snacks!
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	2, 6, 8, 11, 13	

Test Case ID	TC3-6 `End date is before start date`	
Inputs	Start Date	Current Date + 1 Day
	End Date	Current Date - 2 Day
	Selected Pet	Scooby-Doo
	Total Price	300
	Note	Feed my dog Scooby Snacks!
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	3, 5, 8, 11, 12	

Test Case ID	TC3-7 `No pet selected`	
Inputs	Start Date	Current Date + 1 Day
	End Date	Start Date + 1 Day
	Selected Pet	None
	Total Price	300
	Note	<empty string>
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	3, 6, 7, 11, 12	

Test Case ID	TC3-8 `Total price is empty`	
Inputs	Start Date	Current Date + 1 Day
	End Date	Start Date + 1 Day
	Selected Pet	Scooby-Doo
	Total Price	300
	Note	Feed my dog Scooby Snacks!
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	3, 6, 7, 9, 12	

Test Case ID	TC3-9 `Total price is negative`	
Inputs	Start Date	Current Date + 1 Day
	End Date	Start Date + 1 Day
	Selected Pet	Scooby-Doo
	Total Price	-300
	Note	Feed my dog Scooby Snacks!
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	3, 6, 7, 10, 12	

Function: Chatting

Input Equivalence Classes

Equivalent Class ID	Input	Validation
1	Block interlocutor	valid
2	Mute interlocutor	valid
3	Same user role send message to each other	invalid
4	Petsitter chat with Petowner and vice versa	valid
5	Admin chat with User	valid
6	Send message to blocked user	valid
7	Send message to muted user	valid
8	Send not empty string	valid
9	Send empty string	invalid
10	Receive message from blocked user	invalid
11	Receive message from muted user (Notification)	invalid

Test Cases

Assume that

- There are 2 users in the same time, which have a common chatroom
- There are 3 states for each chatroom. normal, muted and blocked

Test Case ID	TC4-1	
Inputs	User role (1)	Petsitter
	User role (2)	Petowner
	Message	<empty string>
	Chatroom's state	Normal
Expected Output	Invalid	
Actual Output	Invalid	
Classes Covered	4,9	

Test Case ID	TC4-2	
Inputs	User role (1)	Petsitter
	User role (2)	Petowner
	Message	"Hello World"
	Chatroom's state	Normal
Expected Output	Valid	
Actual Output	Valid	
Classes Covered	4,8	

Test Case ID	TC4-3	
Inputs	User role (1)	Admin
	User role (2)	Petsitter
	Message	"What is your problem?"
	Chatroom's state	Normal
Expected Output	Valid	
Actual Output	Valid	
Classes Covered	5,8	

Test Case ID	TC4-4	
Inputs	User role (1)	Petsitter
	User role (2)	Petowner
	Message	"Can you hear me?"
	Chatroom's state	Muted
Expected Output	Valid (Sender), Invalid(Receiver's Notification)	
Actual Output	Valid (Sender), Invalid(Receiver's Notification)	
Classes Covered	2,7,8,11	

Test Coverage Matrix

Function ID	Function Name	Test cases ID	Status
1	Pet Owner Register	TC1-1, TC1-2, TC1-3, TC1-4, TC1-5, TC1-6, TC1-7, TC1-8, TC1-9, TC1-10, TC1-11, TC1-12, TC1-13, TC1-14, TC1-15, TC1-16, TC1-17, TC1-18, TC1-19, TC1-20, TC1-21, TC1-22, TC1-23	P, P, P, P, P, P, P, P, P, P, P, P, P, P, P, P, P, P, P, P, P, P, P
2	Seacrhing	TC2-1, TC2-2, TC2-3, TC2-4, TC2-5, TC2-6, TC2-7, TC2-8	P,P,P,P,P,P,P,P
3	Booking	TC3-1, TC3-2, TC3-3, TC3-4, TC3-5, TC3-6, TC3-7, TC3-8, TC3-9	P,P,P,P,P,P,P,P,P
4	Chatting	TC4-1, TC4-2, TC4-3, TC4-4	P,P,P,P

White-box Test Plan

selected function: request (bookingRouter)

We will be using “vitest” which is very similar to JEST.

coverage report HTML of the function:

```
121 1x    request: protectedProcedure
122 1x      .meta({
123 1x        openapi: {
124 1x          method: "POST",
125 1x          path: "/booking/request",
126 1x          summary: "Request a new booking for petOwner",
127 1x          protect: true,
128 1x          tags: ["booking"],
129 1x        },
130 1x      })
131 1x      .input(bookingFields)
132 1x      .mutation(async ({ ctx, input }) => {
133 3x        const userType: UserType = ctx.session.user?.userType ?? null;
134 3x        if (!UserTypeLogic.isPetOwner(userType)) return USER_TYPE_MISMATCH;
135 2x        const petOwnerId = ctx.session.user.id;
136 2x        const uniquePetIdList = [...new Set(input.petIdList)];
137 2x
138 2x        // if (input.startDate < new Date()) {
139 2x        //   return START_DATE_BEFORE_NOW;
140 2x        // }
141 3x        if (input.endDate <= input.startDate) {
142 1x          return END_DATE_BEFORE_START_DATE;
143 1x        }
144 1x
145 1x        return await ctx.prisma.booking.create({
146 1x          data: {
147 1x            petOwnerId: petOwnerId,
148 1x            petSitterId: input.petSitterId,
149 1x            totalPrice: input.totalPrice,
150 1x            startDate: input.startDate,
151 1x            endDate: input.endDate,
152 1x            note: input.note,
153 1x            numberOfPets: uniquePetIdList.length,
154 1x            status: BookingStatus.requested,
155 1x            pet: {
156 1x              connect: uniquePetIdList.map((petId) => ({ petId: petId })),
157 1x            },
158 1x          },
159 1x          select: Return.booking,
160 1x        });
161 1x      }),
```

Note: Green color on the left side indicates that the lines were run by our testcases.

Testcases covered and code are

1. Not Pet Owner: Booking requests must be made by a pet owner. We will check their userType for authorization.

```
// clgxpy7i9000cu52cn9ixcc6g is a pet sitter from seed (invoker)
// clgxpy80o000gu52cw3y3y25g is another pet sitter
test("Not Pet Owner", async () => {
  const ctx = createInnerTRPCContext({
    session: {
      user: { id: "clgxpy7i9000cu52cn9ixcc6g", userType: "PetSitter" },
    },
  });
  const caller = appRouter.createCaller(ctx);

  const fields = {
    petSitterId: "clgxpy80o000gu52cw3y3y25g",
    totalPrice: 696,
    startDate: new Date(Date.now() + dayToMs * 7),
    endDate: new Date(Date.now() + dayToMs * 14),
    petIdList: [],
  };

  expect(await caller.booking.request(fields)).toEqual(USER_TYPE_MISMATCH);
});

```

2. Start time before current time: Start time of a booking request must not be earlier than the present time.

```
41  // clgxpygbm001uu52ct30xg4f6 is their pet
42  // clgxpy80o000gu52cw3y3y25g is another pet sitter
43  test("Start time before current time", async () => {
44    const ctx = createInnerTRPCContext({
45      session: {
46        user: { id: "clgxpyawk000yu52cl9c4j0pe", userType: "PetOwner" },
47      },
48    });
49    const caller = appRouter.createCaller(ctx);
50
51    const fields = {
52      petSitterId: "clgxpy80o000gu52cw3y3y25g",
53      totalPrice: 696,
54      startDate: new Date(Date.now() - dayToMs * 7),
55      endDate: new Date(Date.now() + dayToMs * 14),
56      petIdList: ["clgxpygbm001uu52ct30xg4f6"],
57    };
58
59    await expect(caller.booking.request(fields)).rejects.toBeInstanceOf(
60      TRPCError
61    );
62  });
```

3. End time before start time: End time must be later than start time.

```
64  // clgxpyawk000yu52cl9c4j0pe is a pet owner from seed (invoker)
65  // clgxpygbm001uu52ct30xg4f6 is their pet
66  // clgxpy80o000gu52cw3y3y25g is another pet sitter
67  test("End time before start time", async () => {
68    const ctx = createInnerTRPCContext({
69      session: {
70        user: { id: "clgxpyawk000yu52cl9c4j0pe", userType: "PetOwner" },
71      },
72    });
73    const caller = appRouter.createCaller(ctx);
74
75    const fields = {
76      petSitterId: "clgxpy80o000gu52cw3y3y25g",
77      totalPrice: 696,
78      startDate: new Date(Date.now() + dayToMs * 7),
79      endDate: new Date(Date.now() + dayToMs * 5),
80      petIdList: ["clgxpygbm001uu52ct30xg4f6"],
81    };
82
83    expect(await caller.booking.request(fields)).toEqual(
84      END_DATE_BEFORE_START_DATE
85    );
86  });
```

4. Valid: A valid case, to cover the correct return statement.

```
88  // clgxpyawk000yu52cl9c4j0pe is a pet owner from seed (invoker)
89  // clgxpygbm001uu52ct30xg4f6 is their pet
90  // clgxpy80o000gu52cw3y3y25g is another pet sitter
91  test("Valid", async () => {
92    const ctx = createInnerTRPCContext({
93      session: {
94        user: { id: "clgxpyawk000yu52cl9c4j0pe", userType: "PetOwner" },
95      },
96    });
97    const caller = appRouter.createCaller(ctx);
98
99    const fields = {
100     petSitterId: "clgxpy80o000gu52cw3y3y25g",
101     totalPrice: 696,
102     startDate: new Date(Date.now() + dayToMs * 7),
103     endDate: new Date(Date.now() + dayToMs * 14),
104     petIdList: ["clgxpygbm001uu52ct30xg4f6"],
105   };
106
107   const res = await caller.booking.request(fields);
108   expect(res).toBeInstanceOf(Object);
109   expect(res).not.toEqual({});
110 });
```

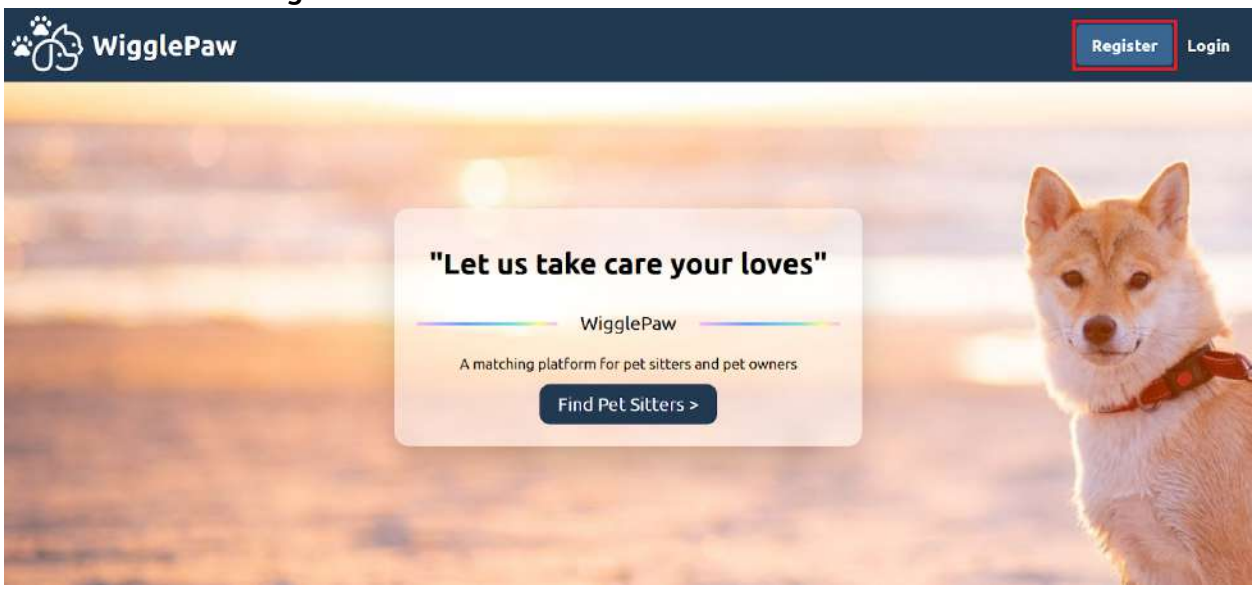
User Manual

Function: Register

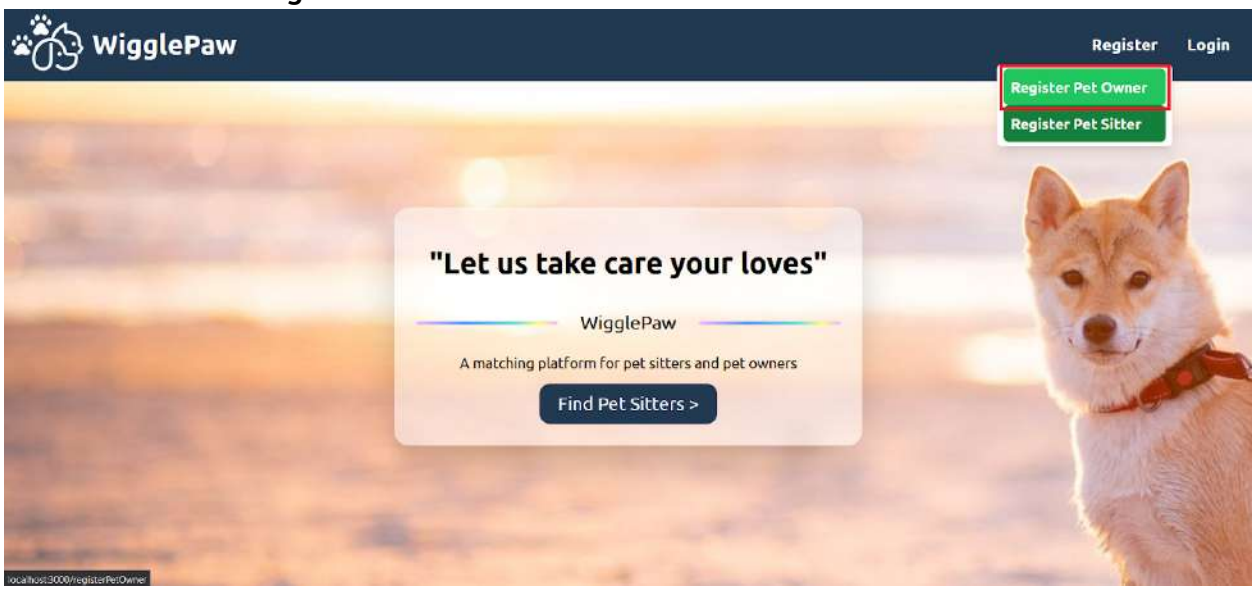
Role : Pet Owner

For use that want to register as a **Pet Owner** to have their pets looked after by Pet Sitter.

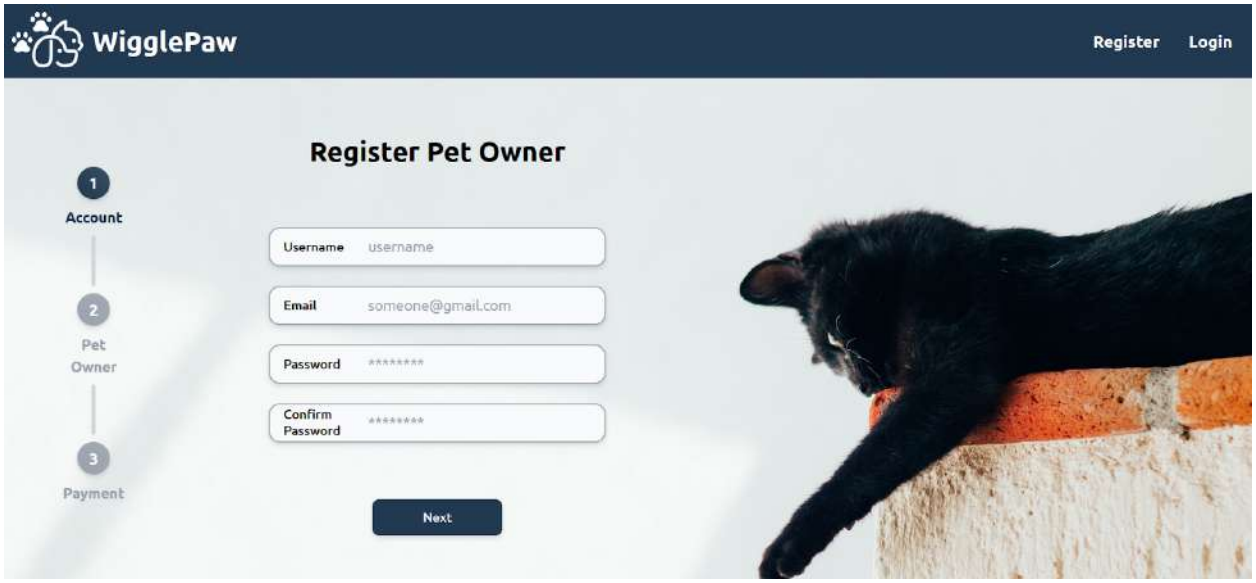
1. Click on the **Register** button on the header



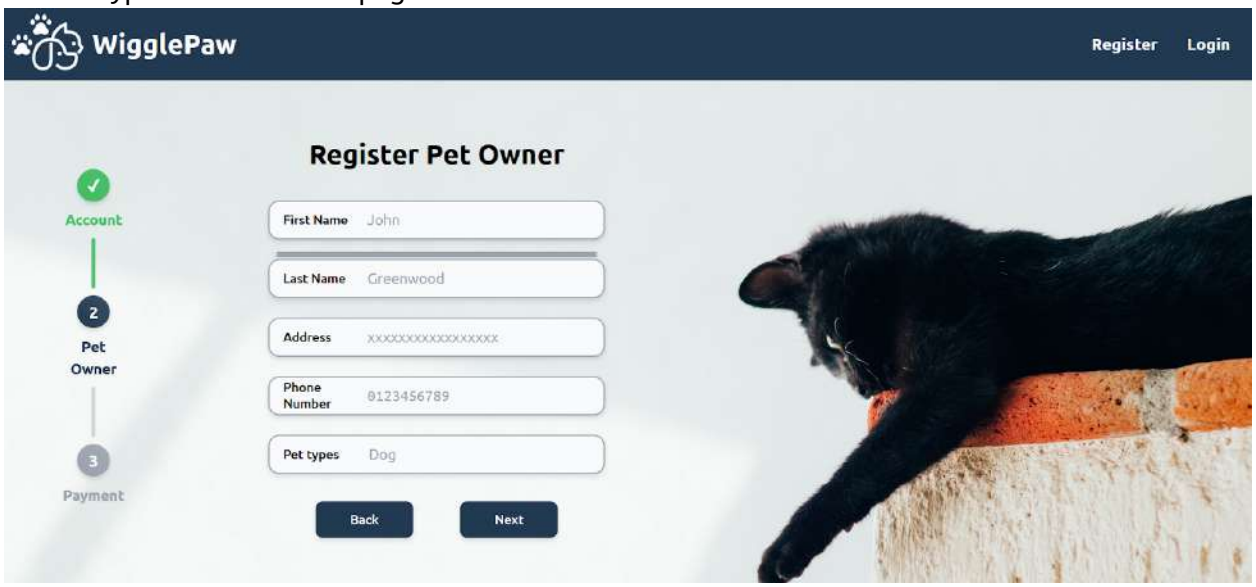
2. Click on the **Register Pet Owner** button



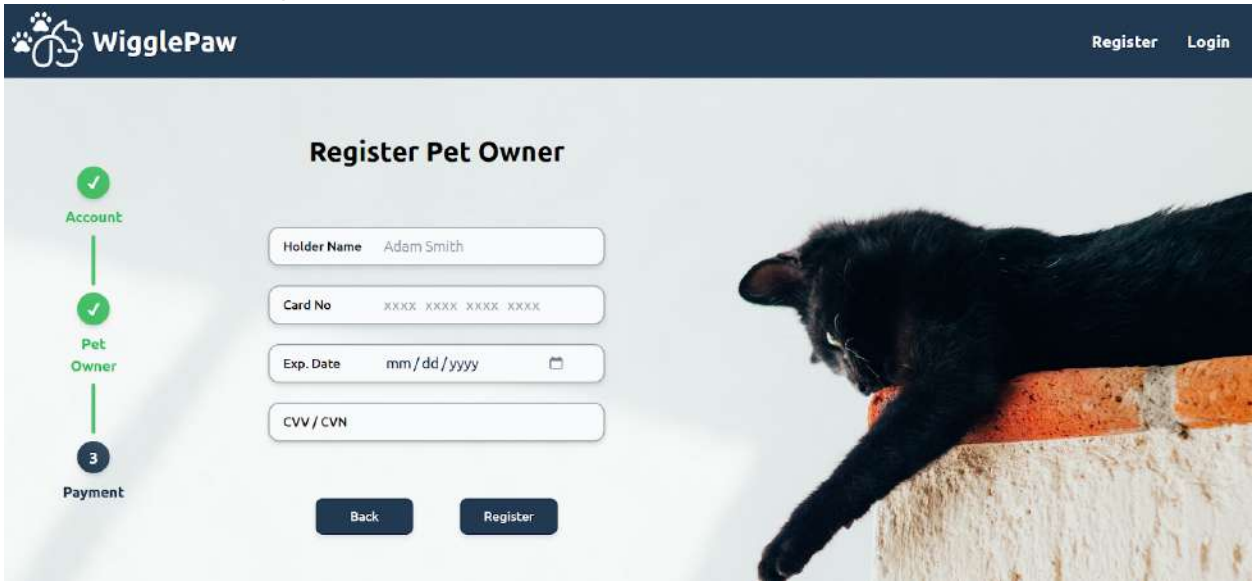
3. Field in all account informations (Username, Email, Password) on the first page then click on the **Next** button



4. Field in all personal informations (First Name, Last Name, Address, Phone Number, Pet Type) on the second page then click on the **Next** button



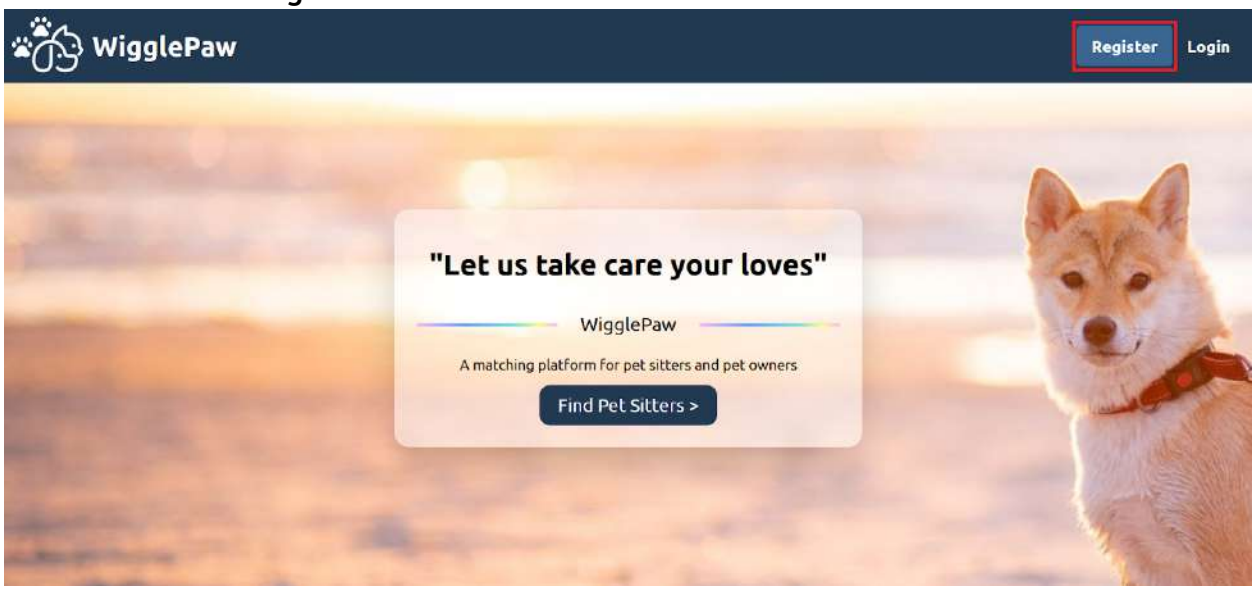
5. Field in all payment informations on the last page then click on the **Register** button on the bottom of the page to create Pet Owner Account



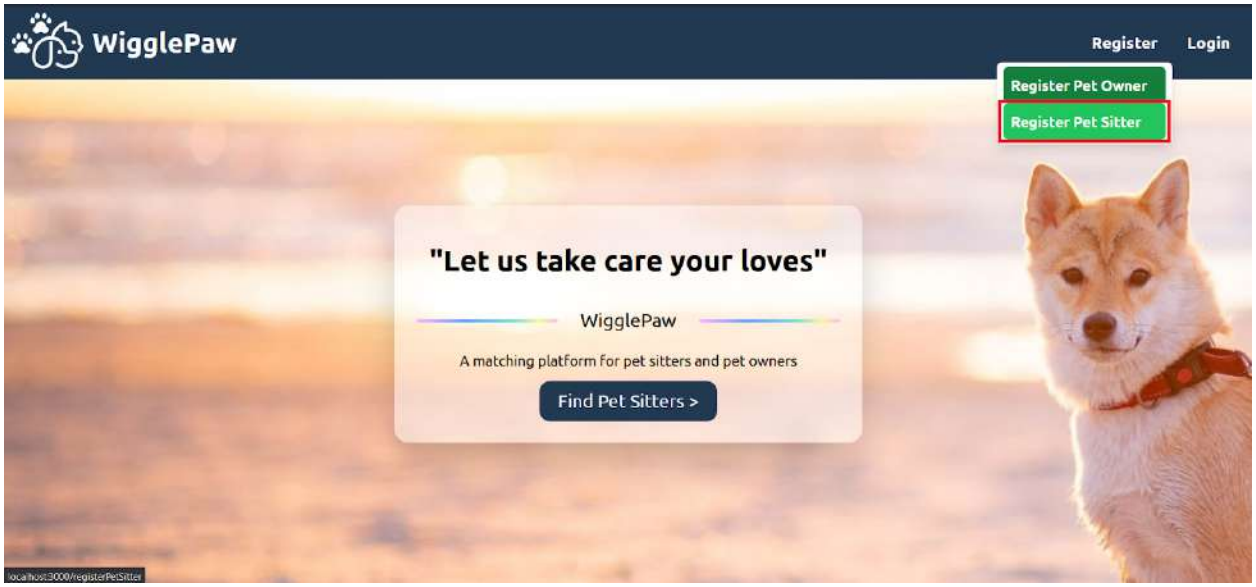
Role : Pet Sitter

For use that want to register as a **Pet Sitter** either Freelance or Pet Hotel to look after pets of Pet Owner.

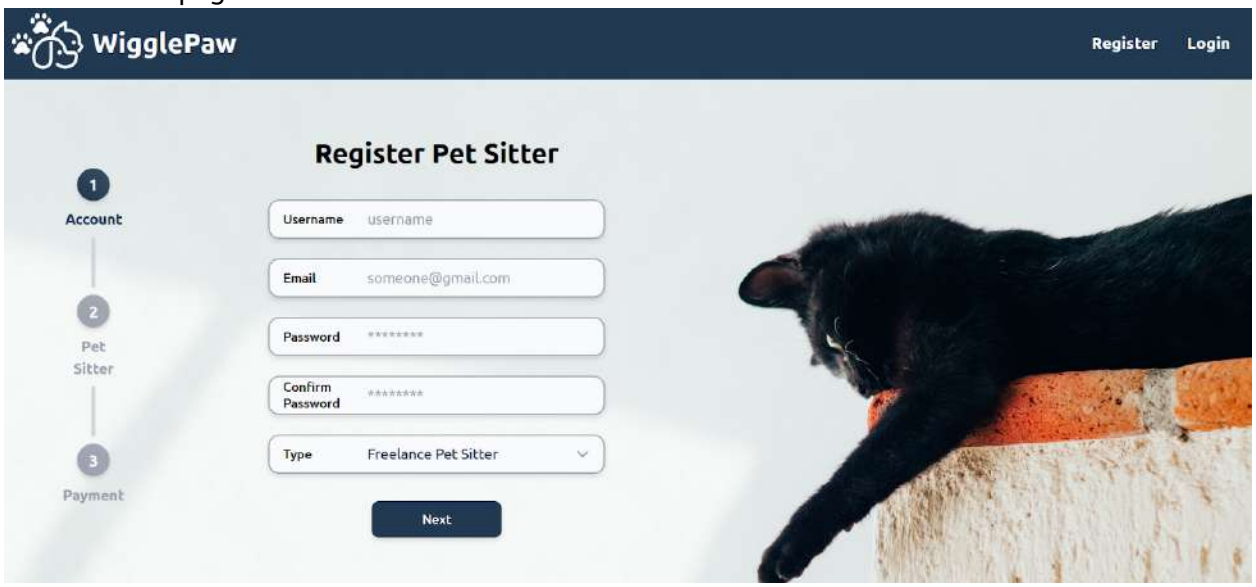
1. Click on the **Register** button on the header



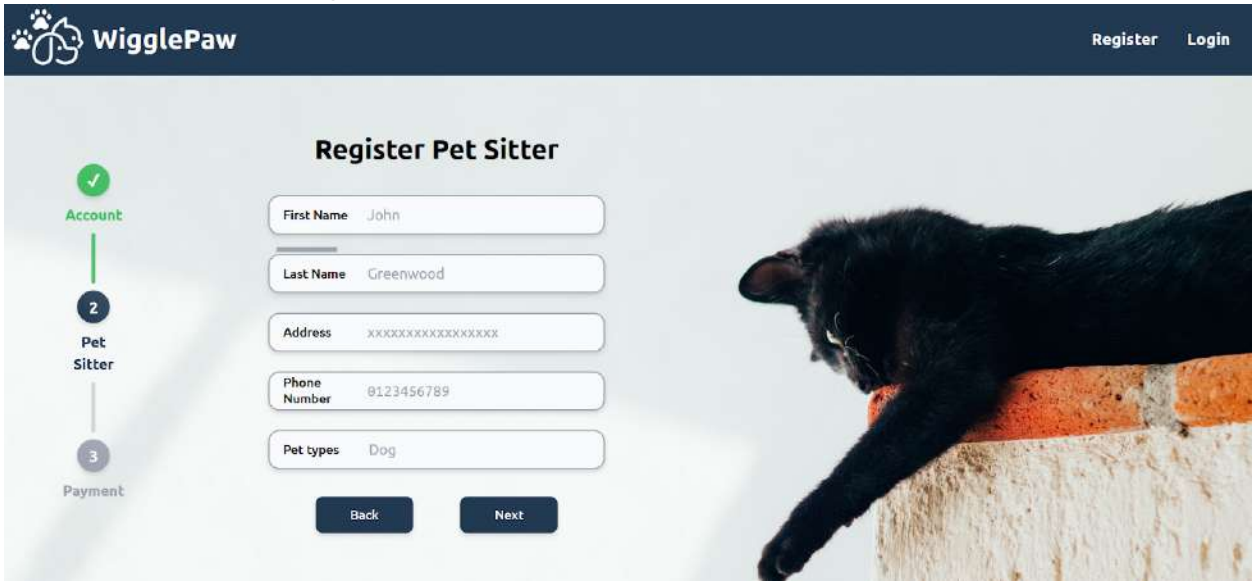
2. Click on the **Register Pet Sitter** button



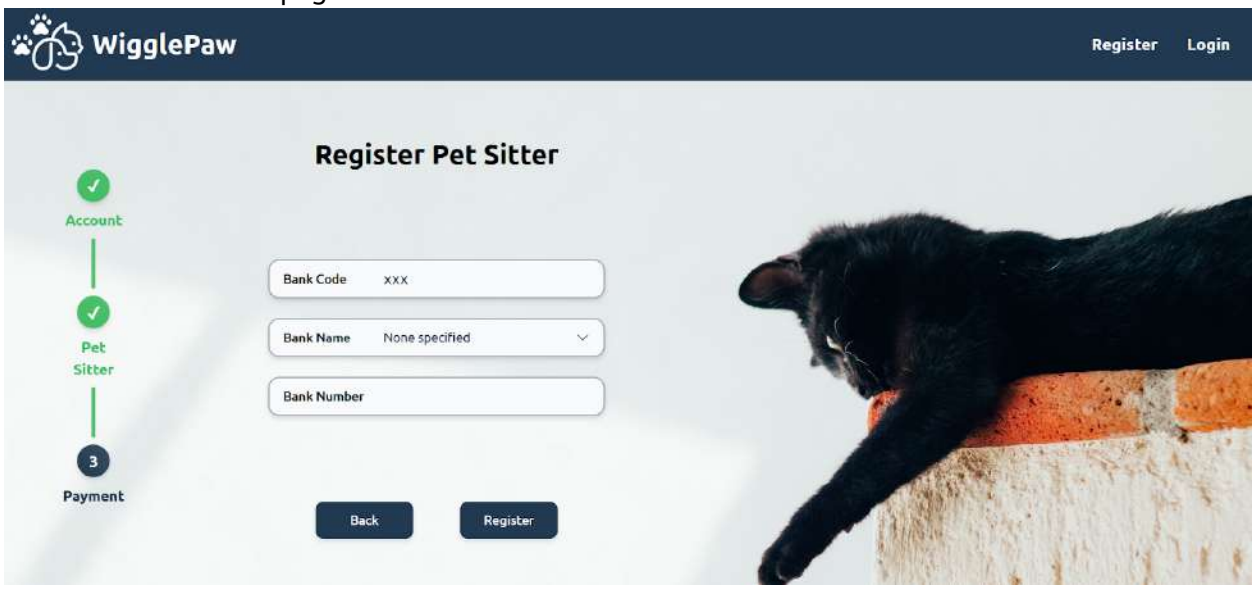
3. Field in all account informations (User, Email, Password) on the first page and choose the role you want either Freelance or Pet Hotel then click on the **Next** button on the bottom of the page



4. Field in all pet sitter informations on the second page then click on the **Next** button on the bottom of the page



5. Field in all payment informations on the last page then click on the **Submit** button on the bottom of the page to create Pet Sitter Account

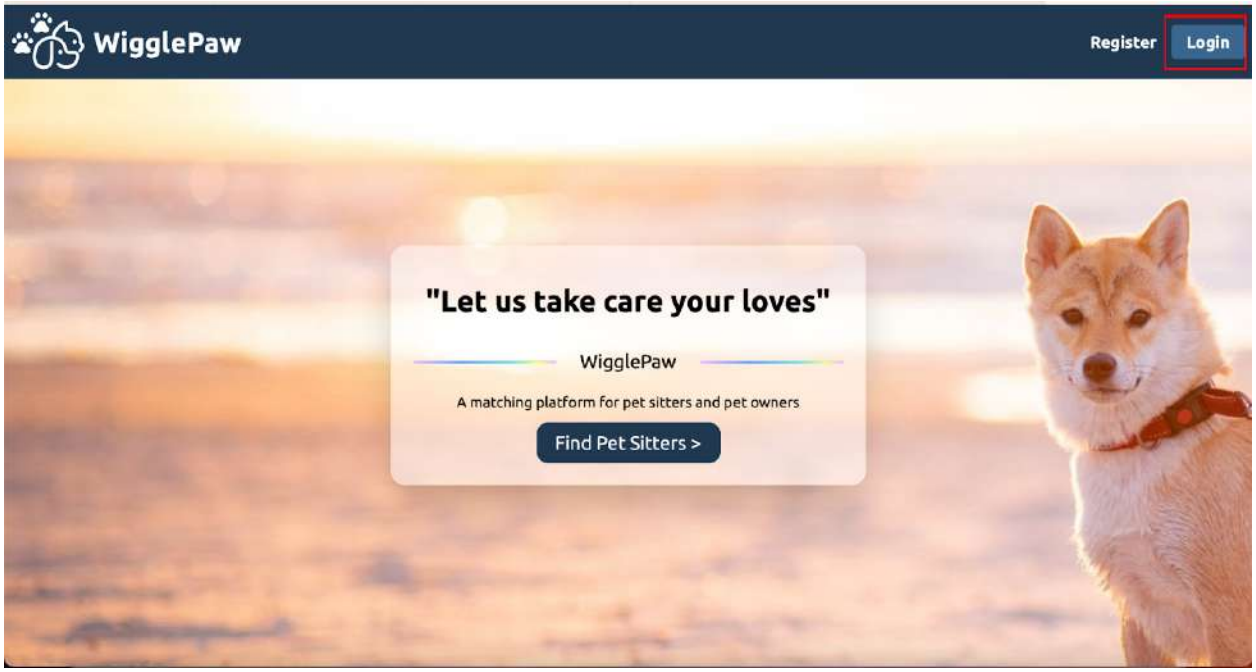


Function: Login

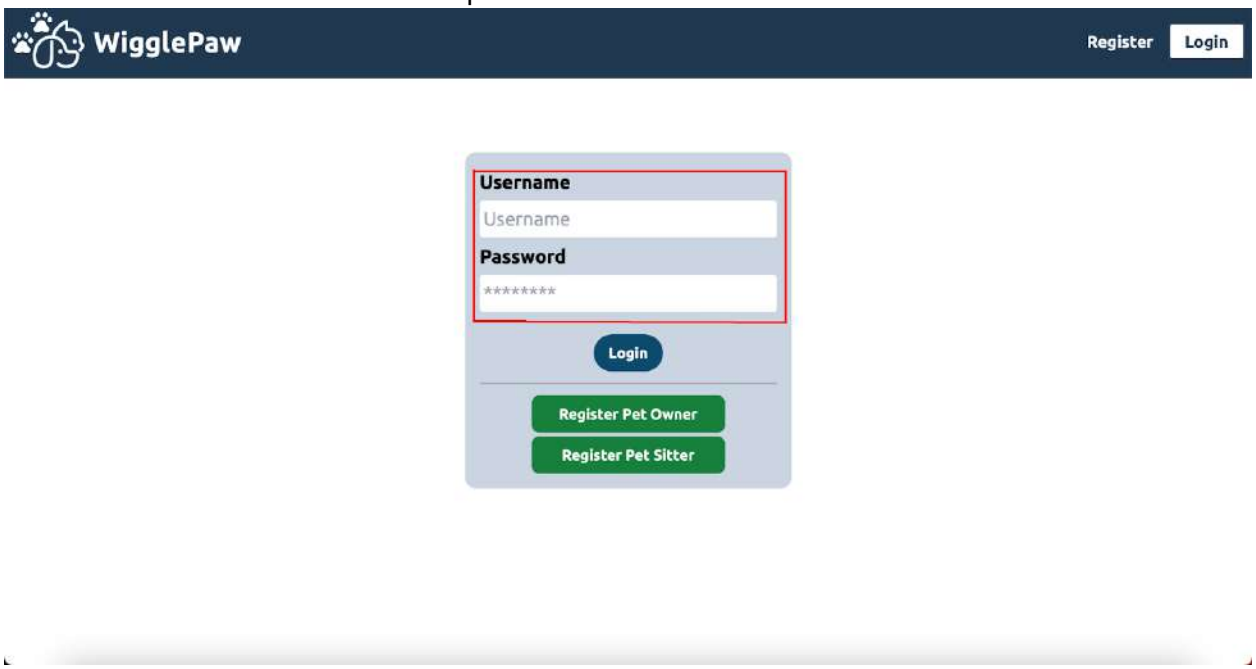
Role : User

Instructions

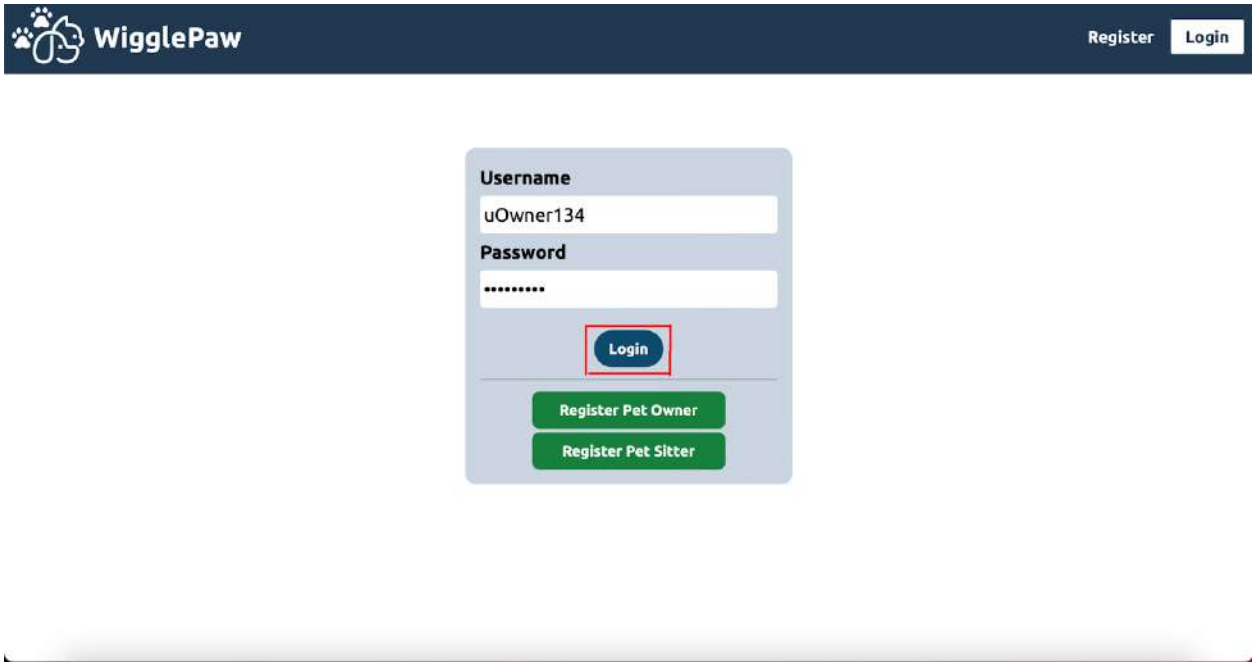
1. User clicks the “**Login**” button on the header



2. User fills their username and password



3. User clicks on the **submit** button to login


WigglePaw
Register
Login
Username
uOwner134
Password
Login
Register Pet Owner
Register Pet Sitter

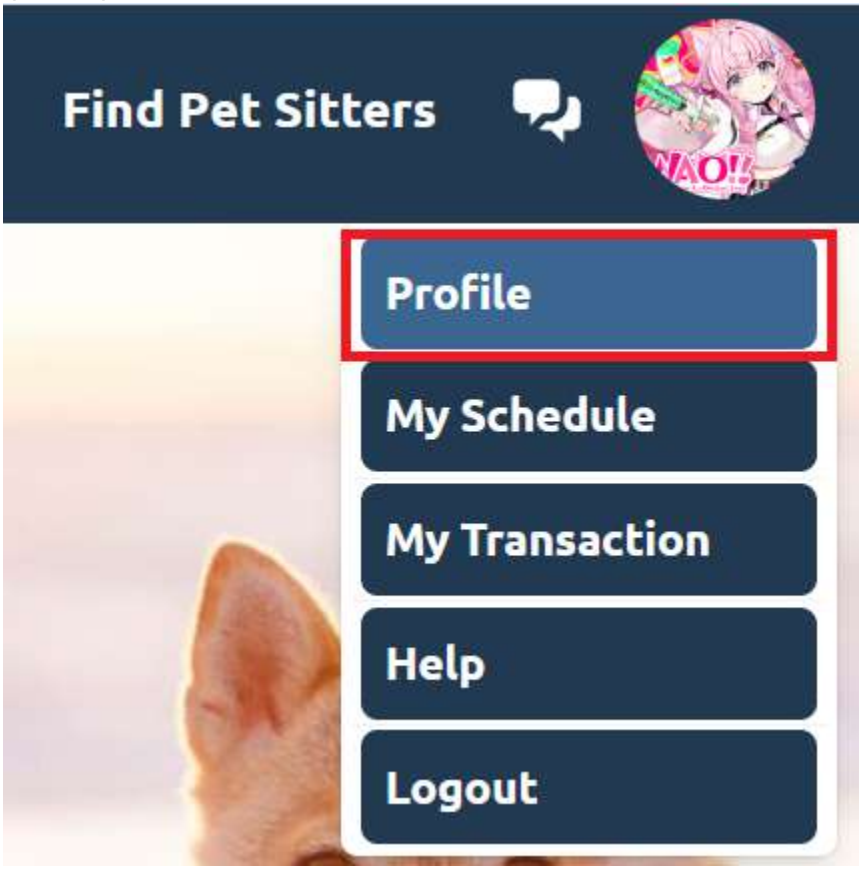
Function: Profile

Role: User

Instructions

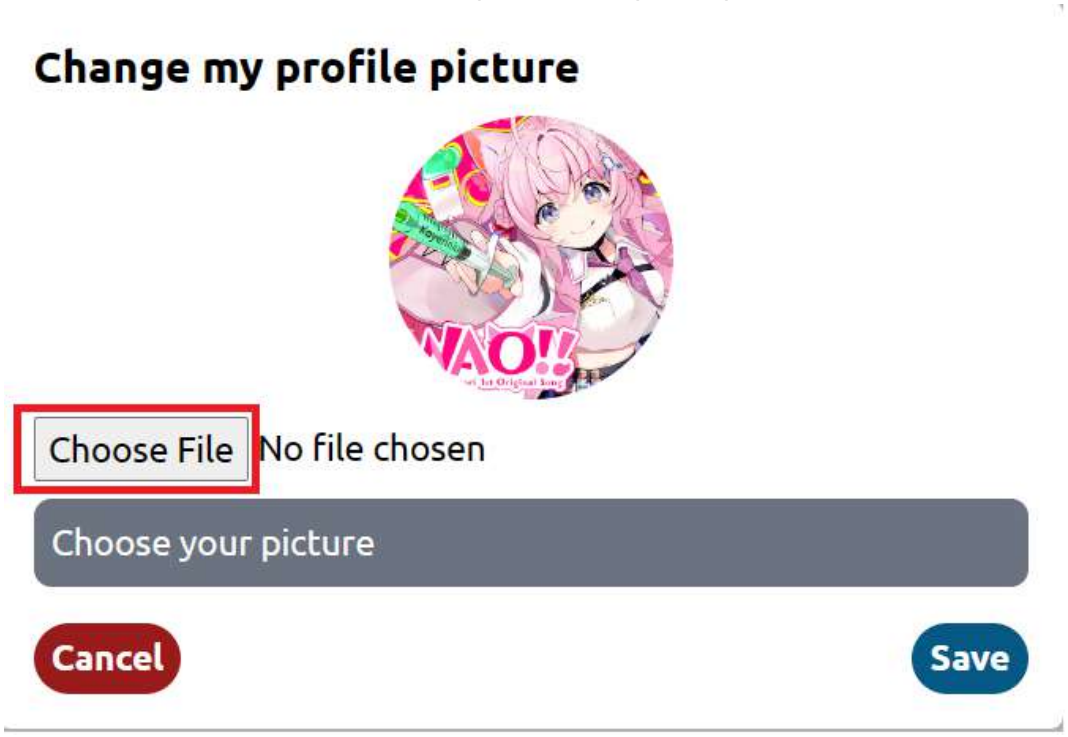
To Edit Profile

1. After logging in, click the **profile picture** on **top right corner**. Then, select the **profile** tab to enter profile page.

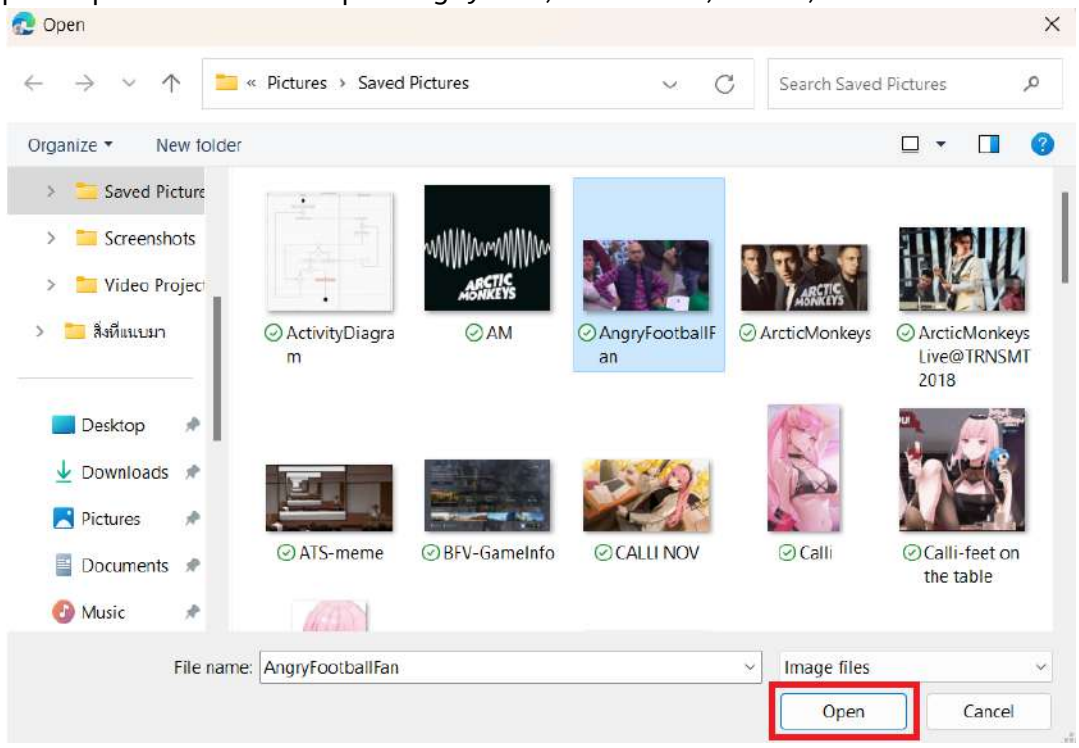


2. To change profile picture, user click the profile picture in the profile page.

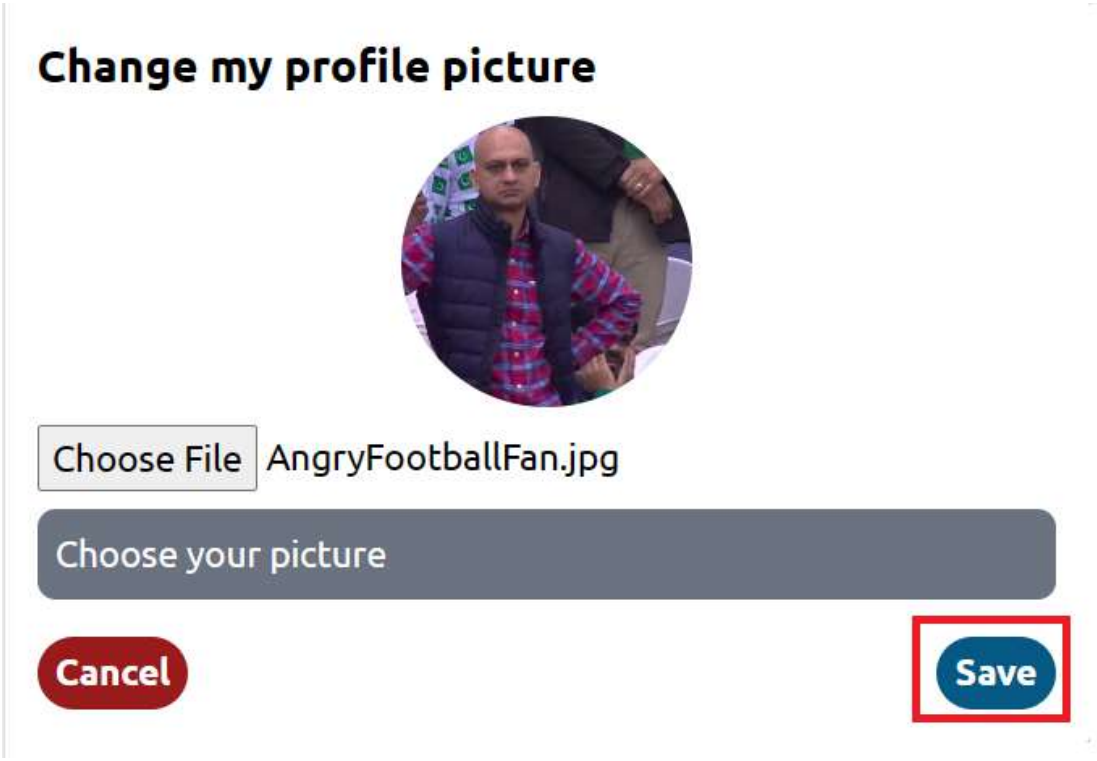
3. User clicks “**Choose File**” button to upload a new profile picture.



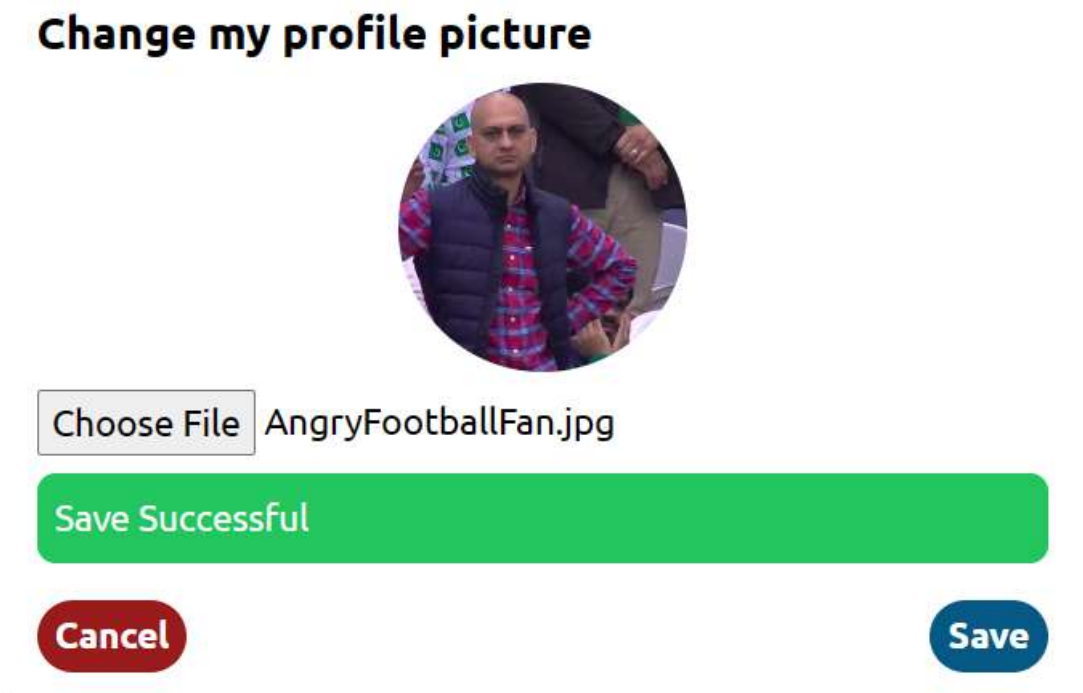
4. User selects the desired picture to upload and click open button to upload picture. (This part depends on the user operating system; WindowsOS, MacOS, etc.)



5. User clicks **“save”** button to save the uploaded picture as profile picture.



6. User will get a success notification after saving profile picture successfully.



7. To edit profile information, user click **“Edit my profile”** button.

8. User changes desired fields by clicking at the text box below the field such that the cursor pops up.

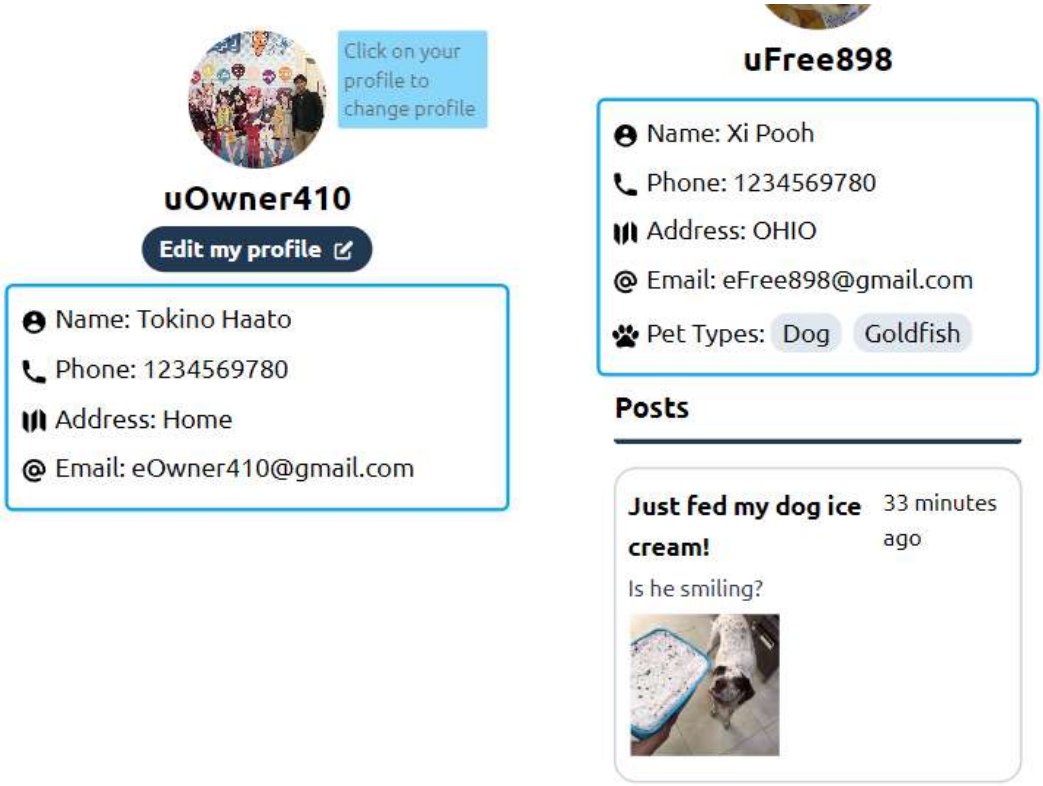
9. After finish changing profile information, user clicks **“Save profile”** button to save.

10. User will now see their new edited profile information.

Function: Post

Role: Pet owner

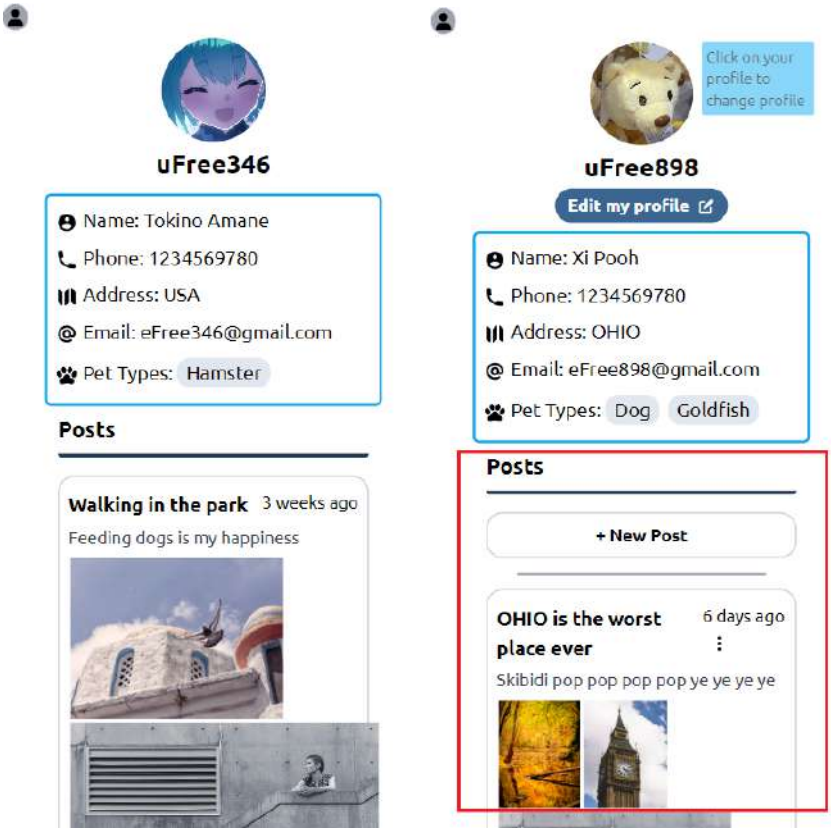
1. A pet owner does not have the ability to view or create their own post, but they can view posts of other pet sitters.



2. A pet owner can view post images in their full size by clicking on the image. They can also view other images of the same post by clicking on arrows.

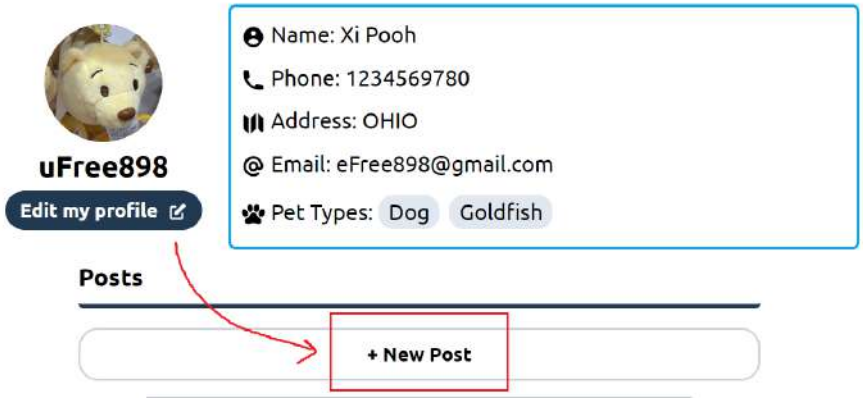
Role: Pet sitter

1. A pet sitter can only view posts in other pet sitter's profile pages, while they can both view and edit posts in their own profile page.

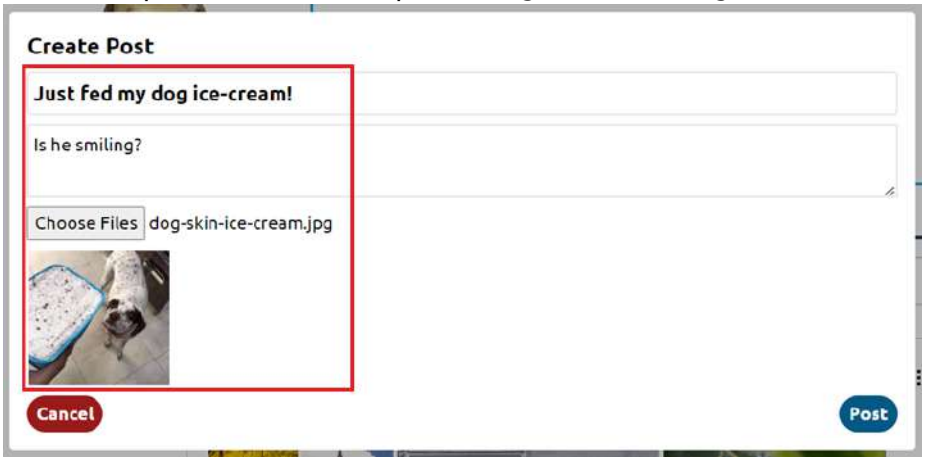


2. A pet sitter can view post images in their full size by clicking on the image. They can also view other images of the same post by clicking on arrows.

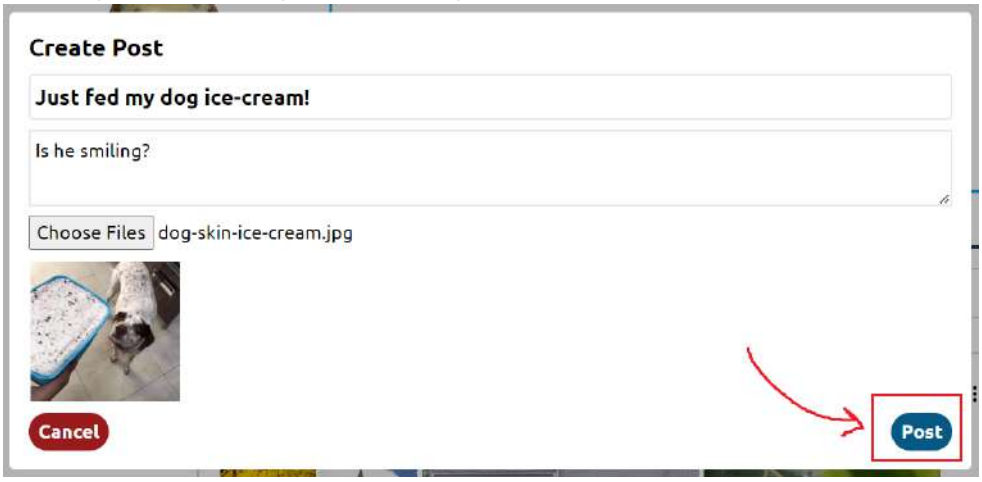
3. A pet sitter can create a new post by clicking **'New Post'**.



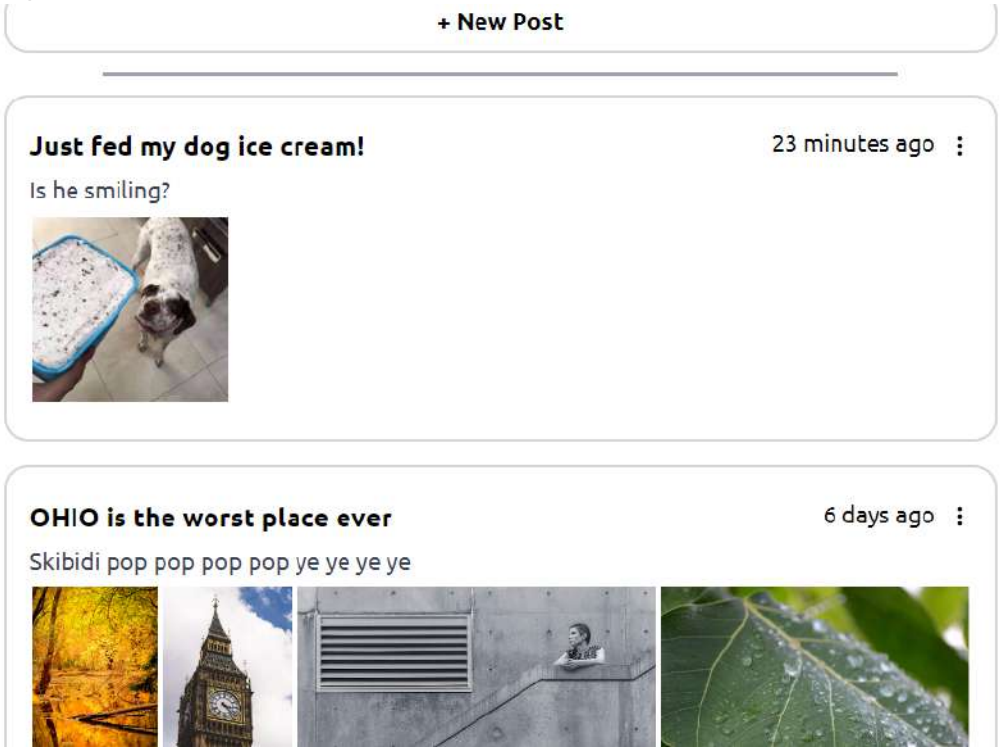
4. A pet sitter can edit post contents and upload images in the dialog.



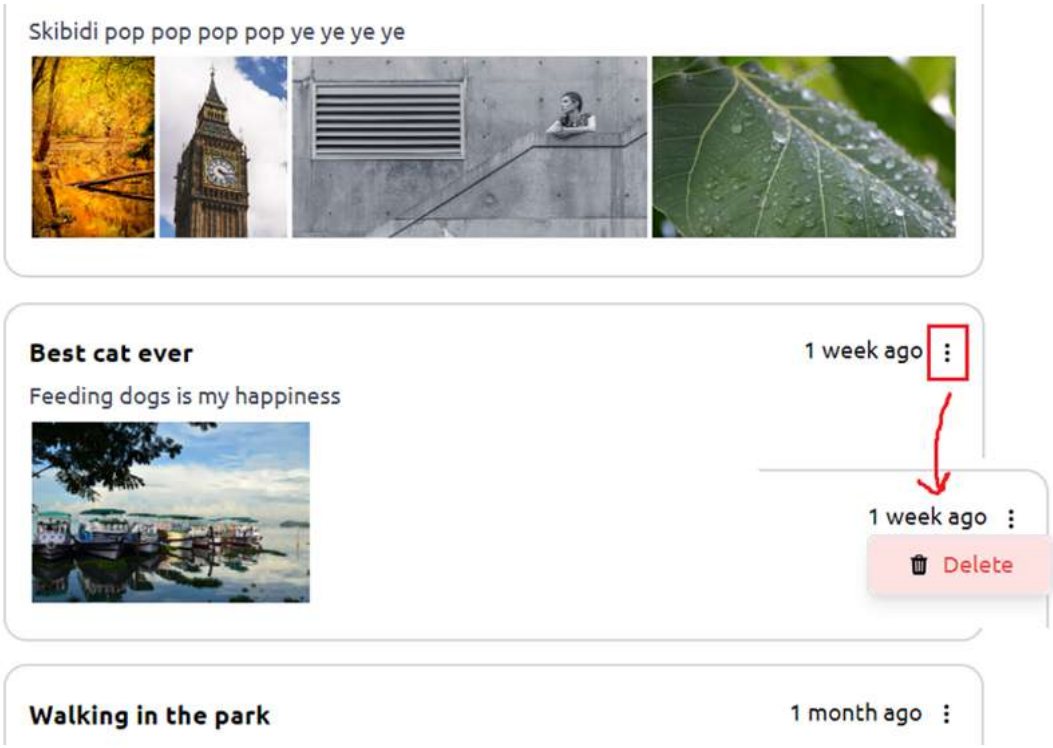
5. After filling in post details, a pet sitter can post by clicking **'Post'**



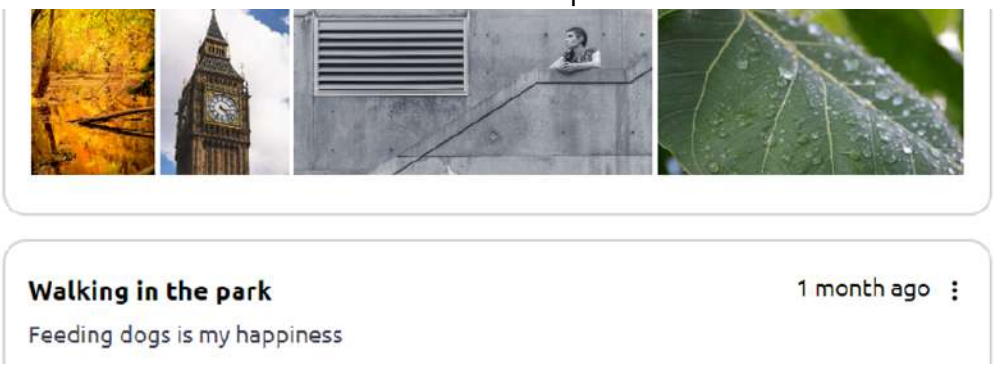
6. Once the post is successfully created, the pet sitter receives a popup and the post becomes visible to all users.



7. A pet sitter can also delete their post by clicking on the kebab menu (three dots) and then **'Delete'**



8. The pet sitter and other users can now see that the post is deleted.

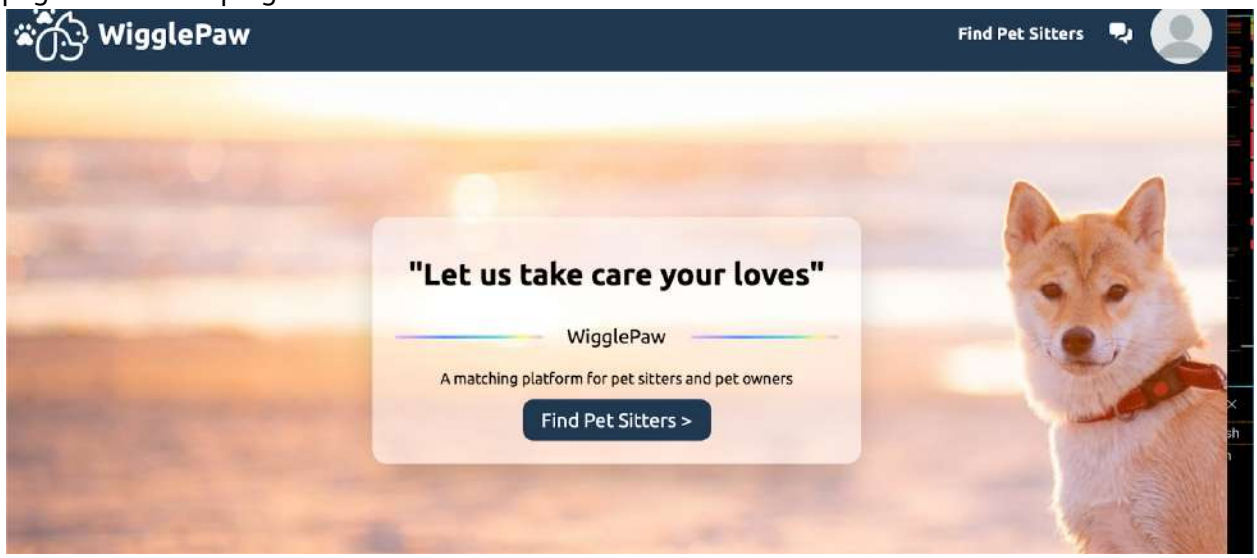


Function: Searching

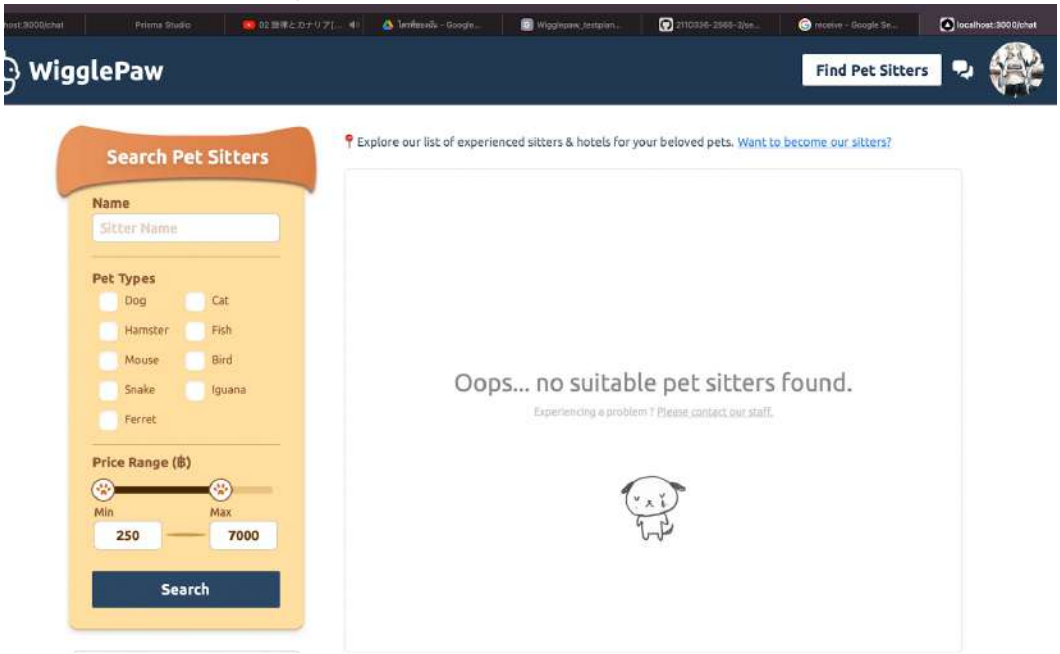
Role: Pet owner

Instructions

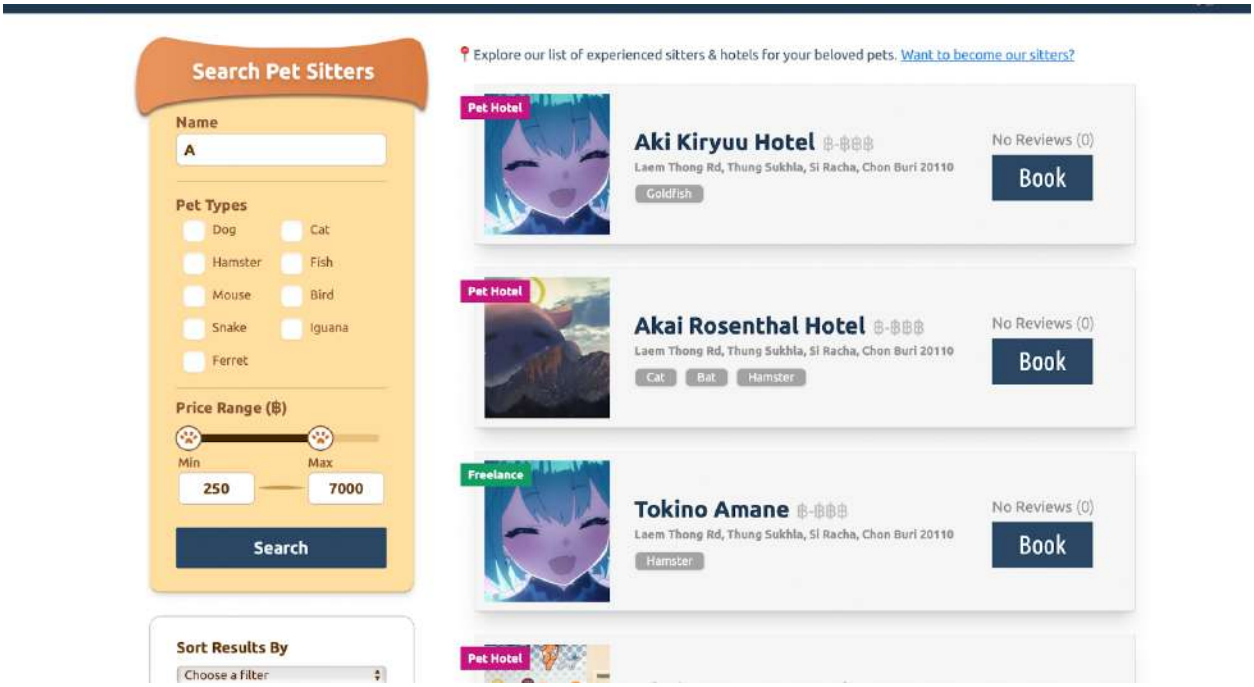
1. User goes to matching page by clicking on the “**Find Pet Sitters**” button from the landing page or on the top right of the screen



2. User fills in fields and filters (Note that they are all optional). Click the “Search” button to show matching pet sitters.



3. Users can find all pet sitters who passed the criteria on the right side. User can scroll down to see more.

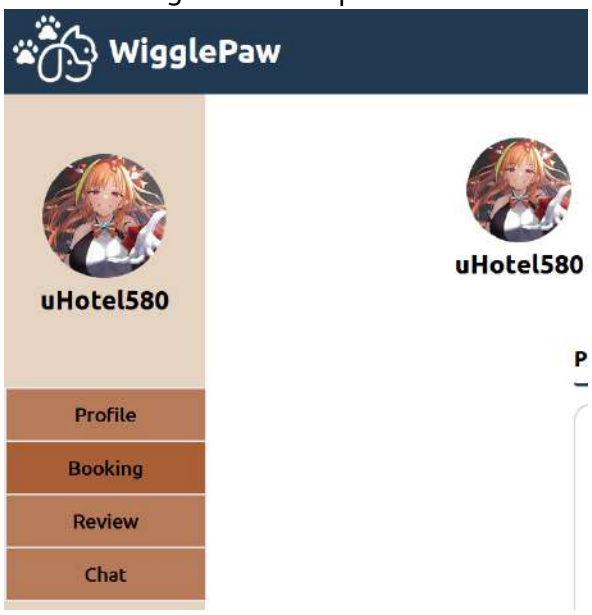


Function: Booking

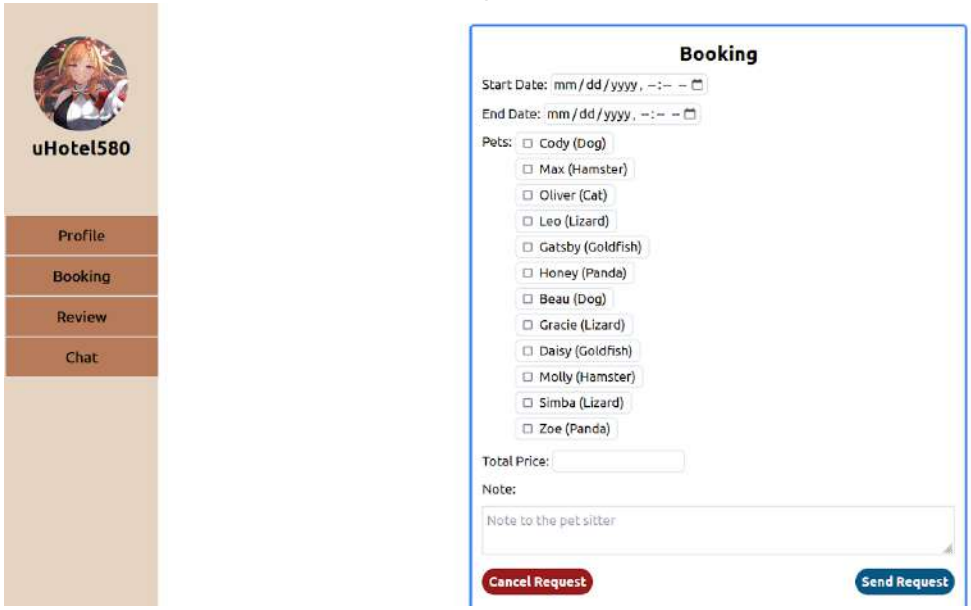
Role : Pet owner

Instructions

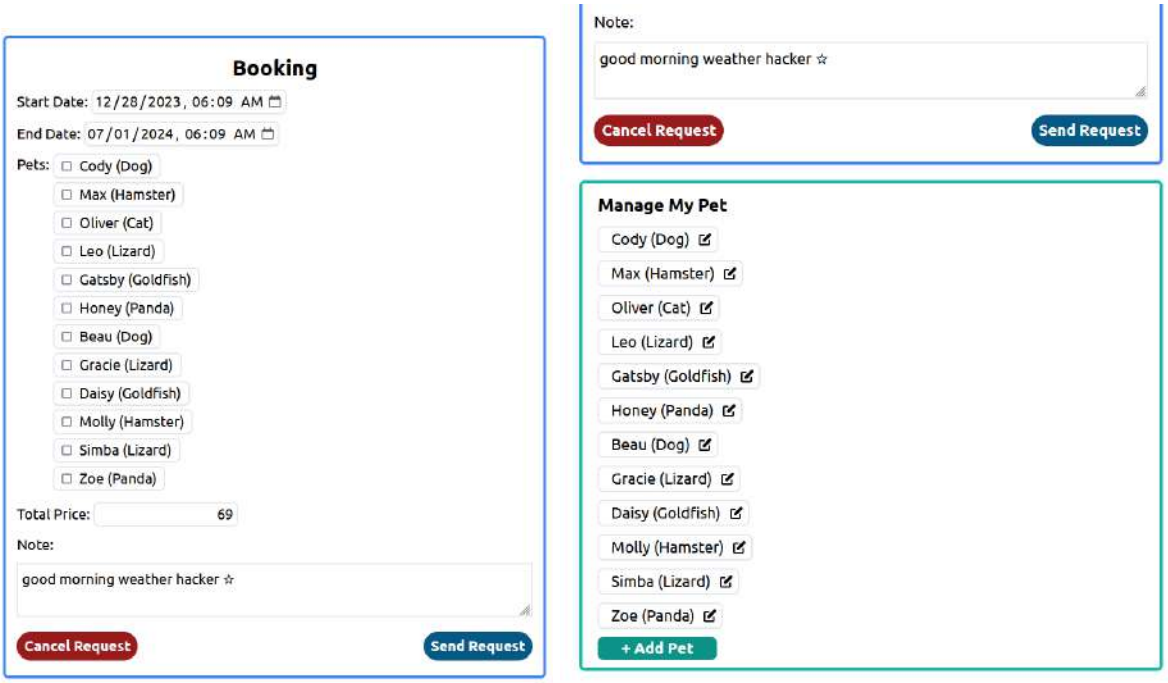
1. User can start booking by clicking on the “**Booking**” button in a pet sitter’s profile page after choosing the desired pet sitter from the searching page.



2. User will be redirected to the booking page.



3. After filling form, user can click “Send Request” button to request for booking. User can scroll down to manage their pets as well.

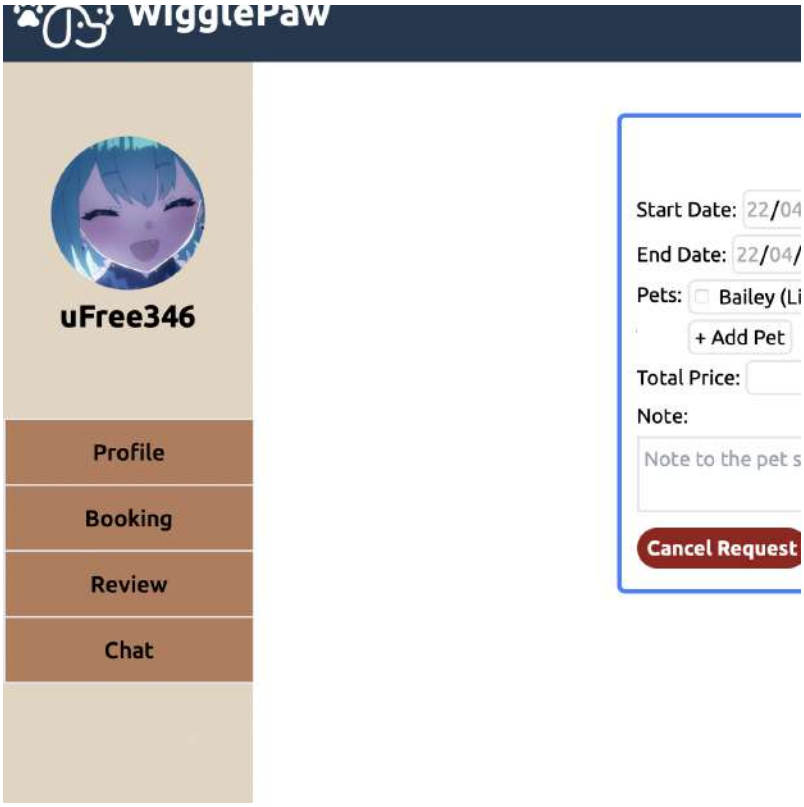


Function: Chatting

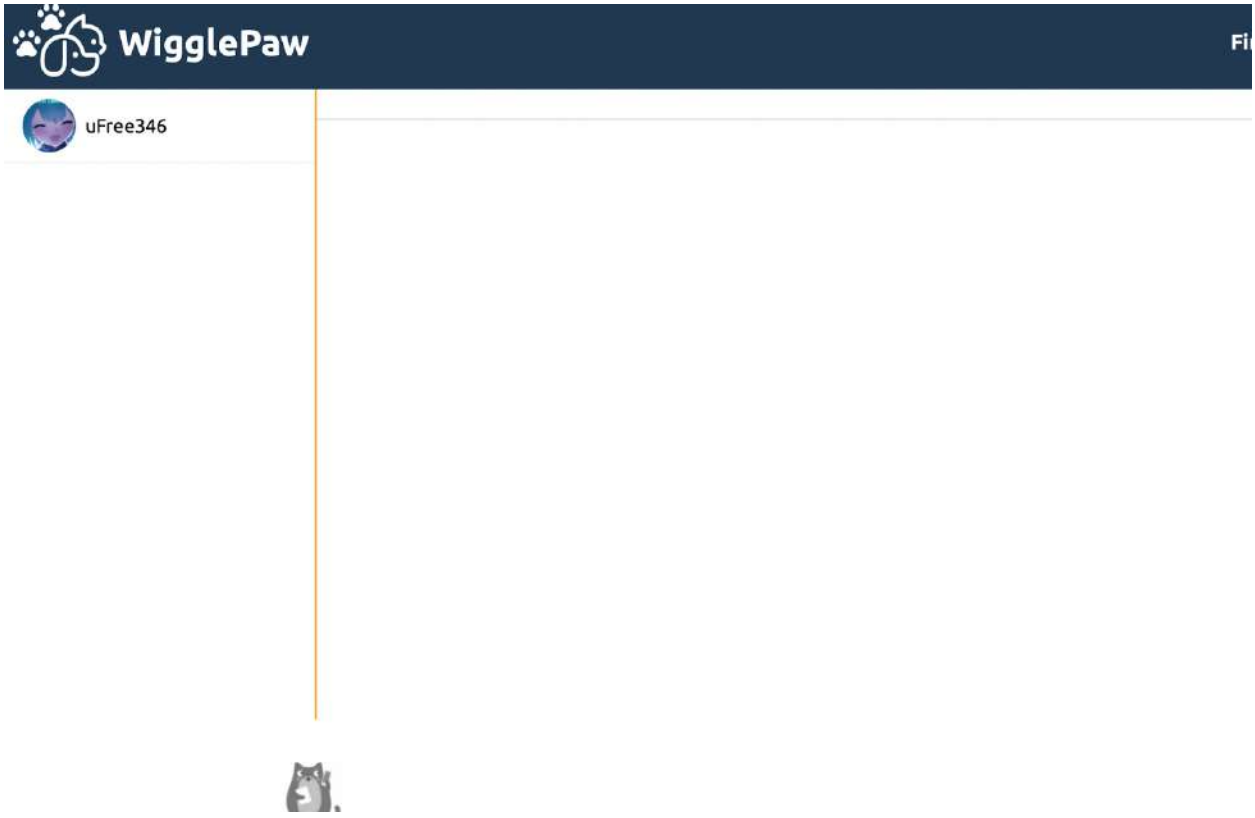
Role : Pet owner

Instructions

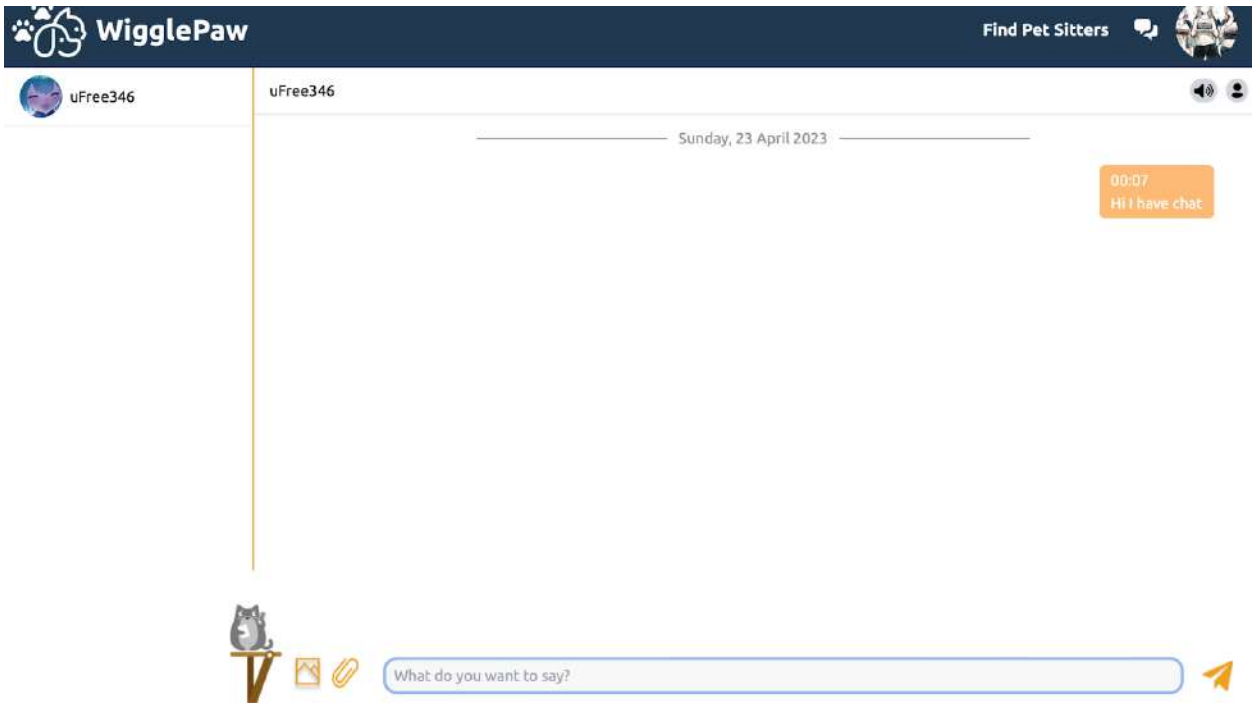
4. User can start chatting by clicking on the “**Chat**” button in a pet sitter’s profile page after choosing the desired pet sitter from the searching page.



5. User will be redirected to the chatting page where they can click on the desired pet sitter profile located on the left side to start chat or continue chat.



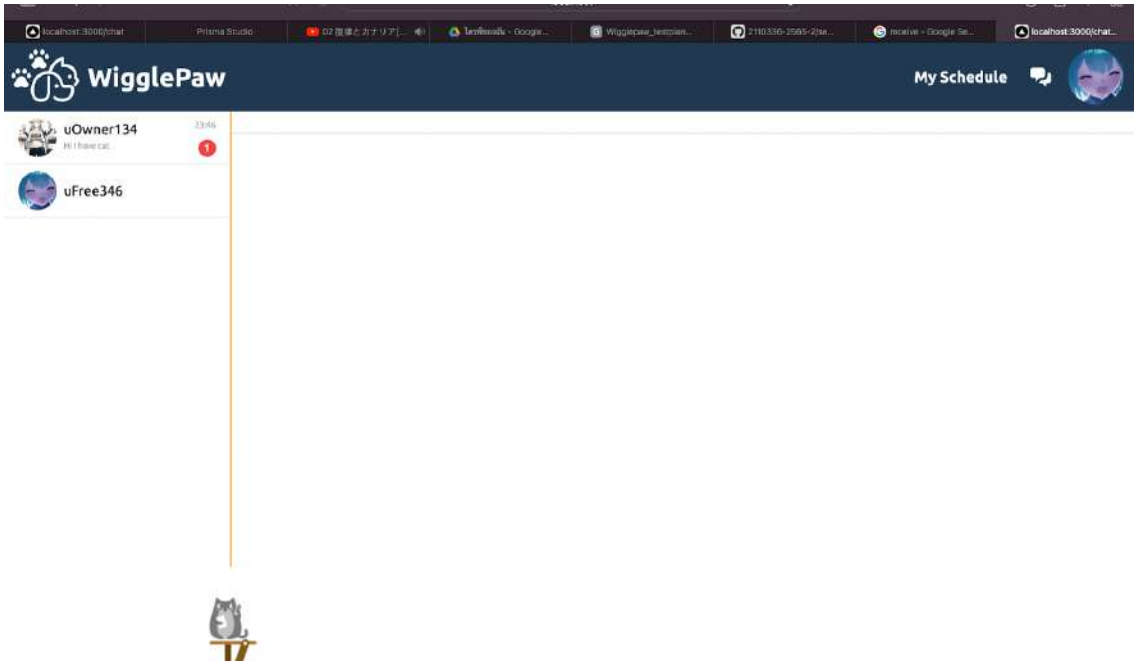
6. After choosing who to chat with, message input box will appear, and user can send message to that specific pet sitter



Role : Pet sitter

Instruction

1. User can access chat by clicking on the **chat icon** on the top right of the app, all the existing chatrooms will be displayed here, including all new incoming chat message from new pet owner marked with notifications (red circle with a number)



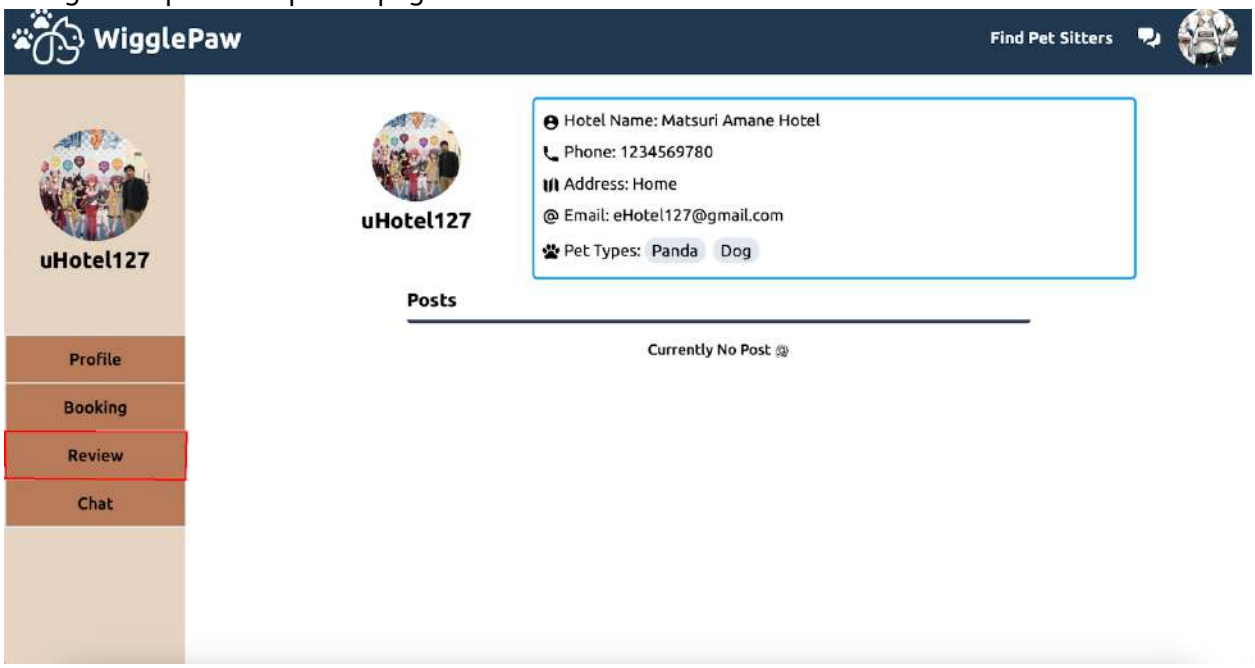
2. Users can click on the desired chatroom, and continue chatting.

Function: Review

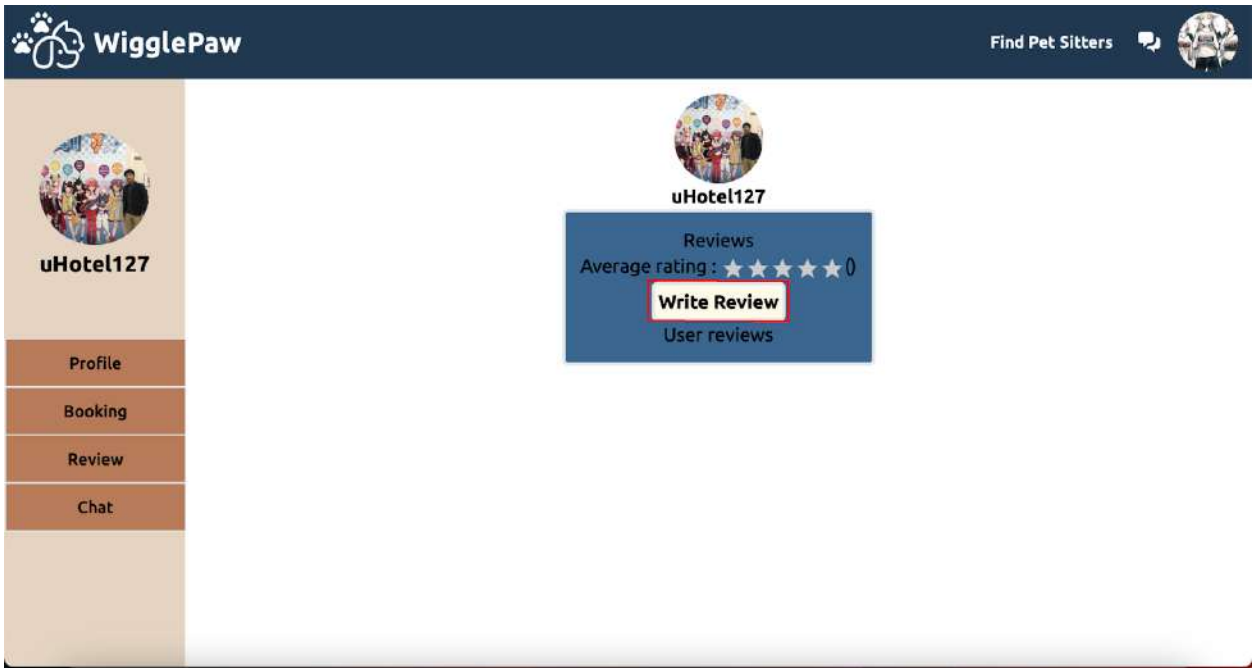
Role : Pet owner

Instruction

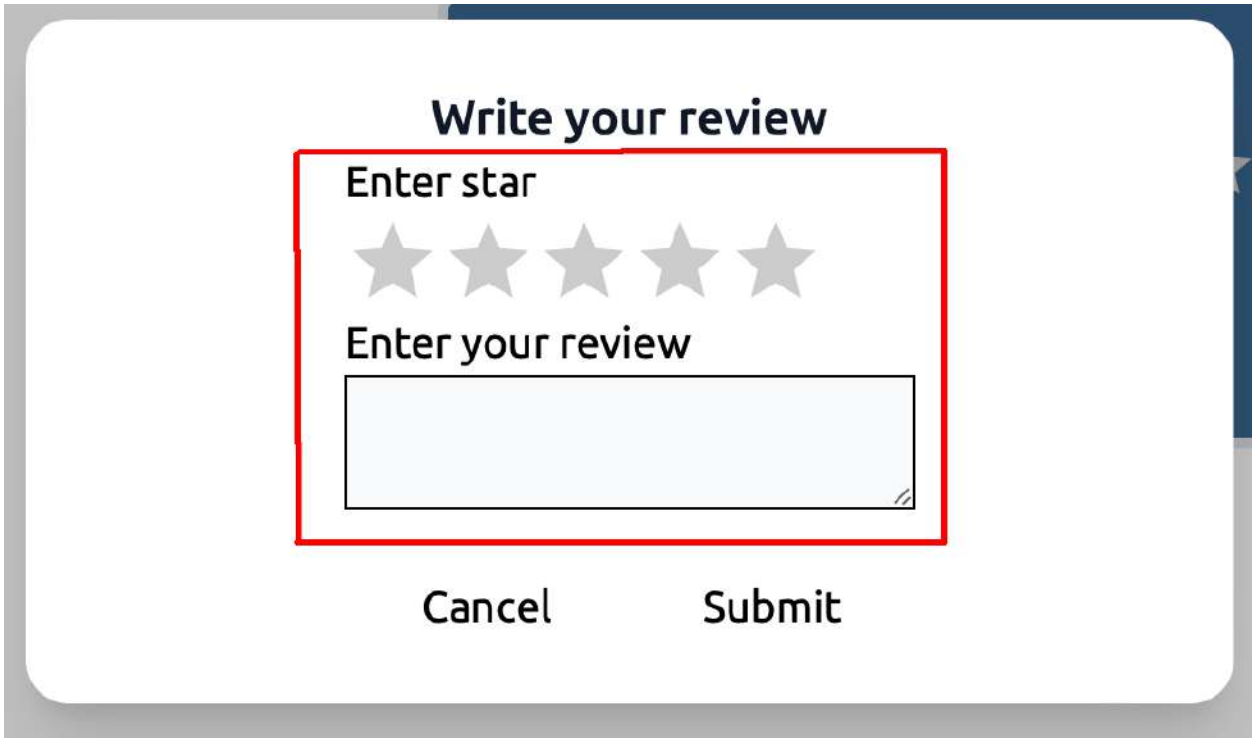
1. User goes to pet sitter profile page and click on the “**Review**” button



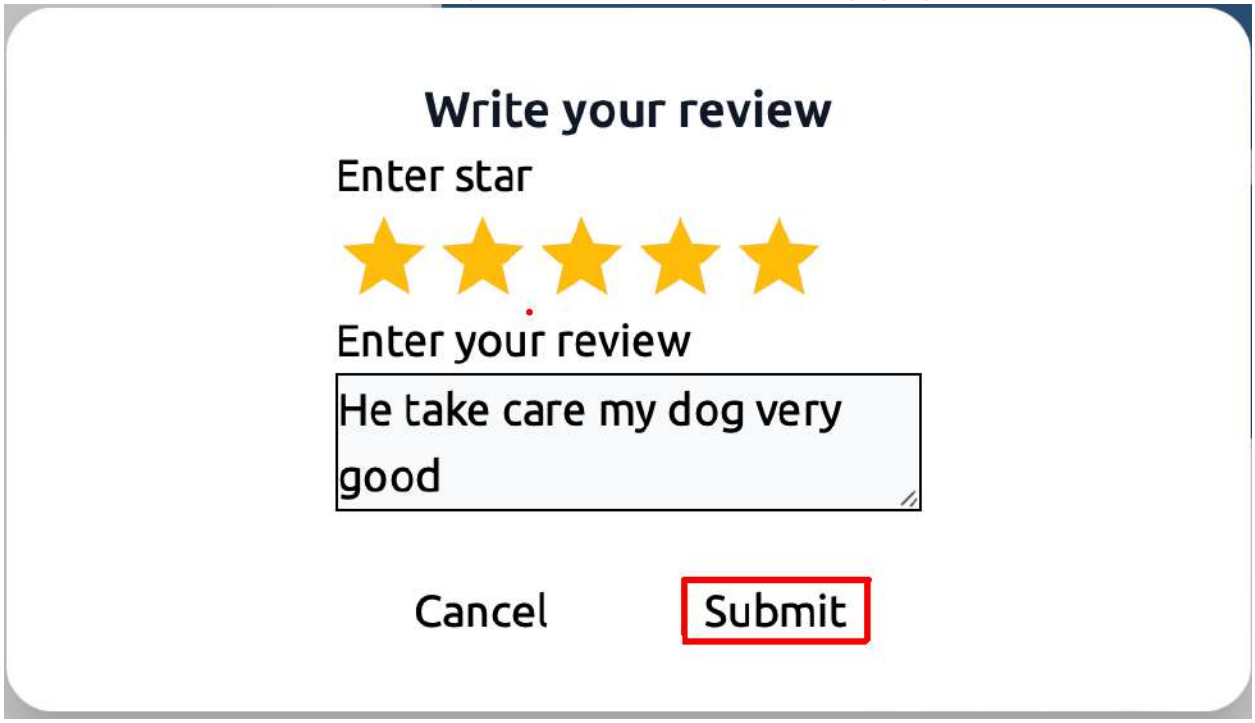
2. User clicks on the “**Write Review**” button



3. User clicks on the star to rate and writes review in the box.



4. If user wants to enter a review, click on the "**Submit**" button. User may also click the "**Cancel**" button to delete all review progress and close the review pop-up box.

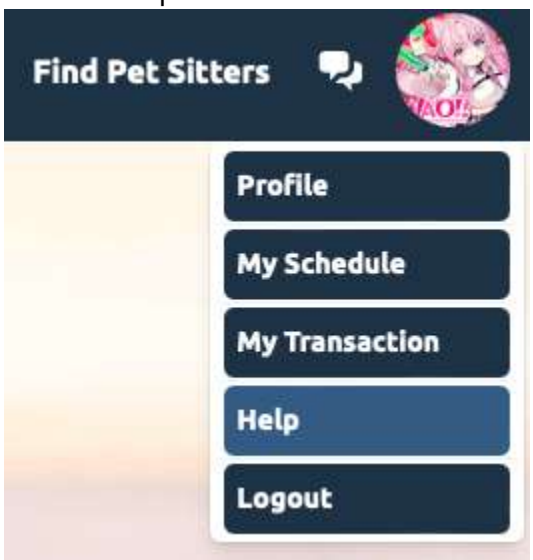


Function: Reporting Problems

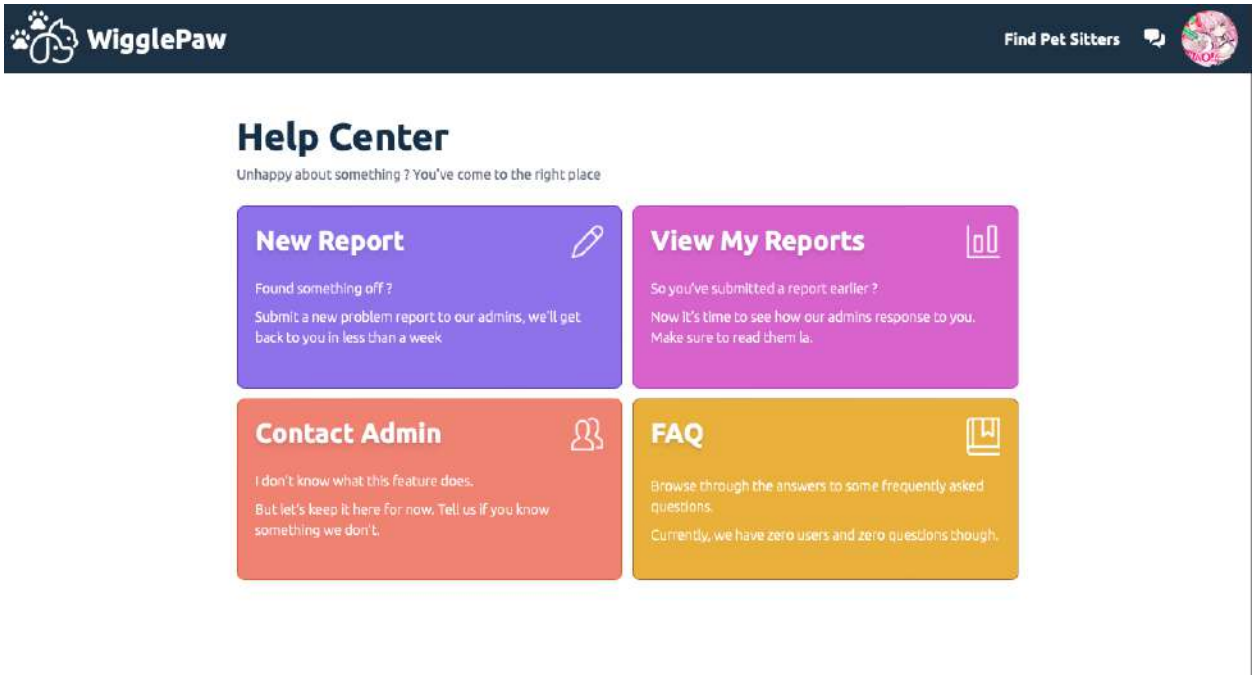
Role : User

Instruction for reporting a problem

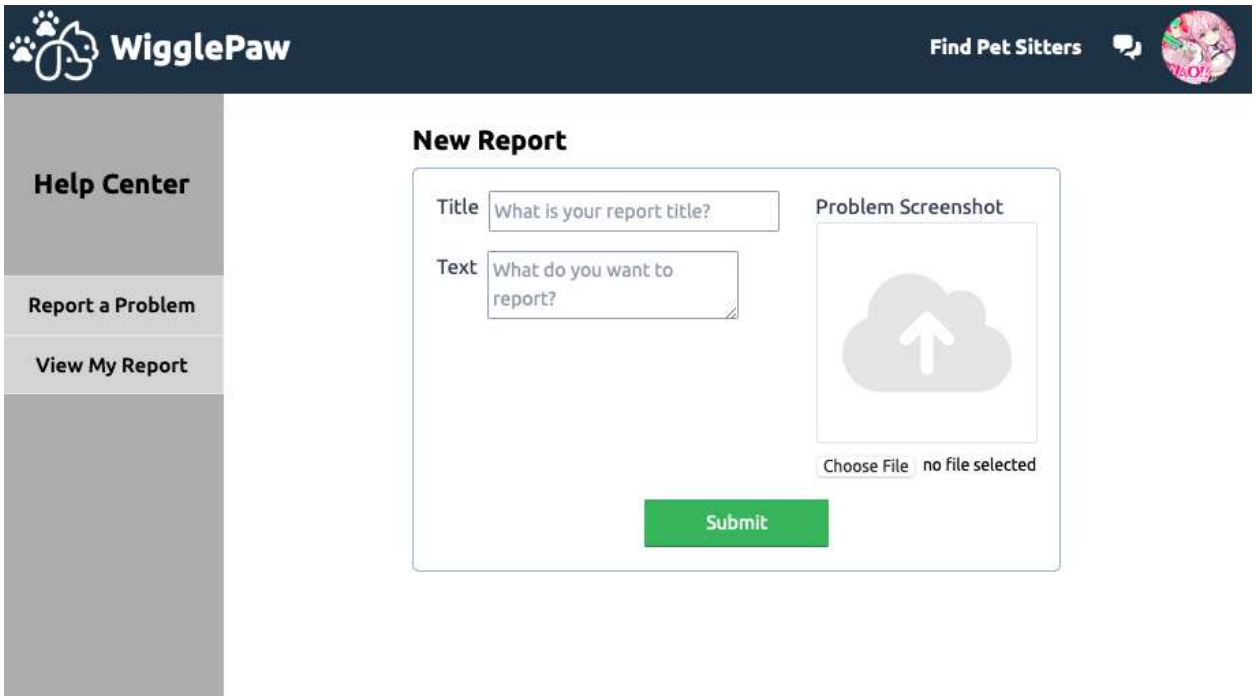
1. Go to the Help pages by clicking the profile picture in the header and select the Help option from the dropdown



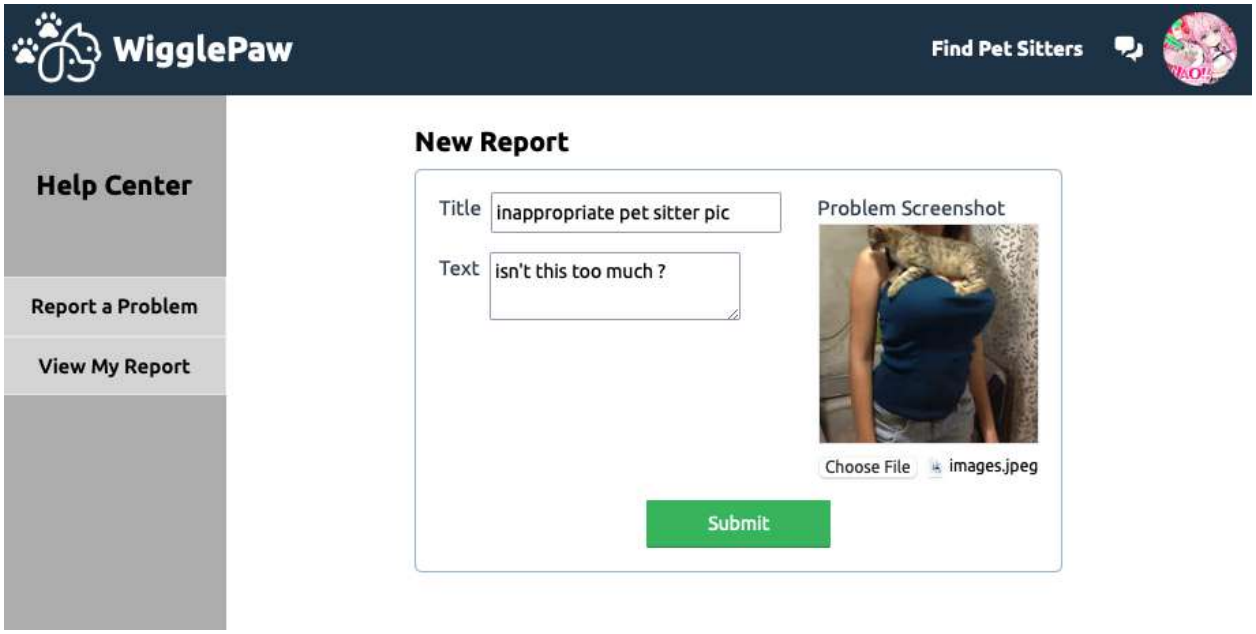
2. Click on 'New Report'



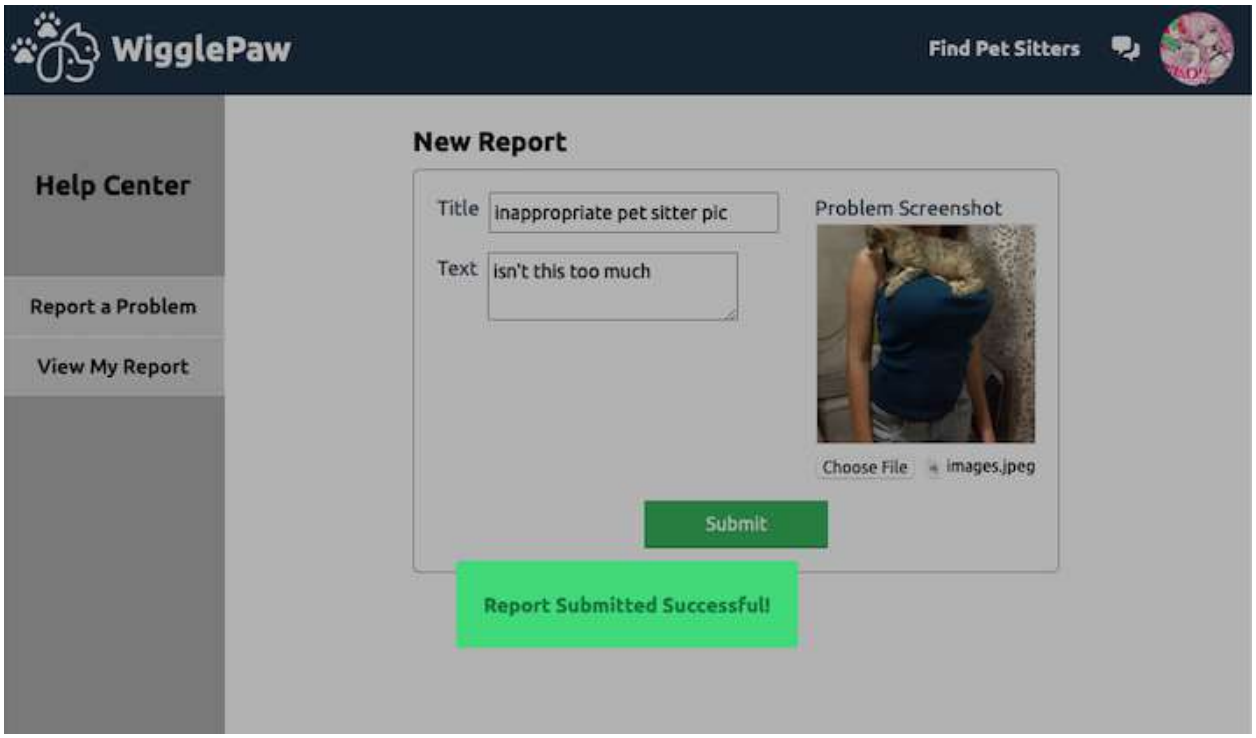
3. Fill in the form



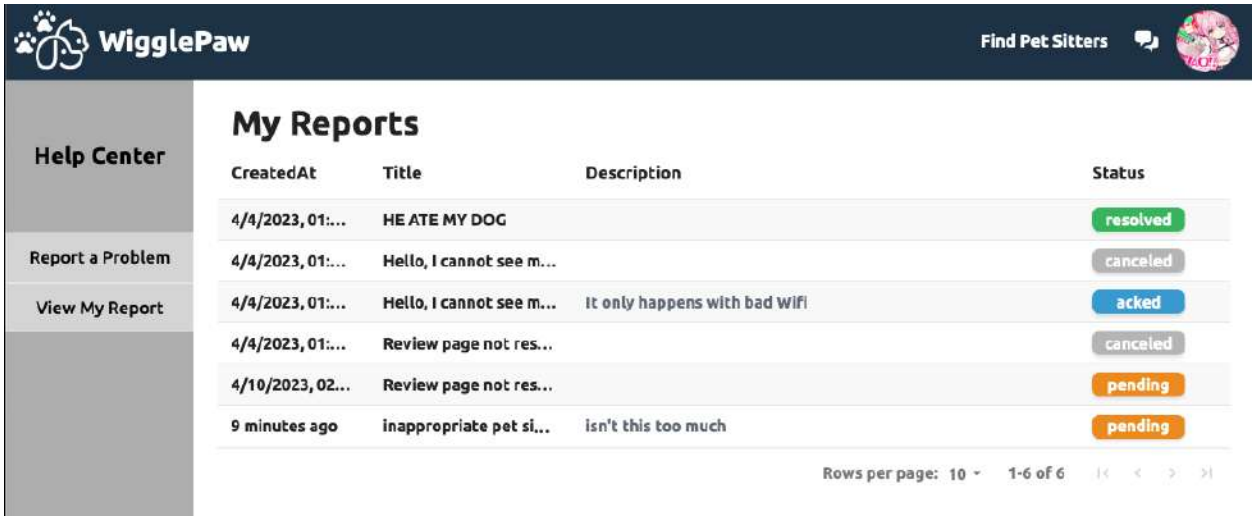
4. Upload any relevant images



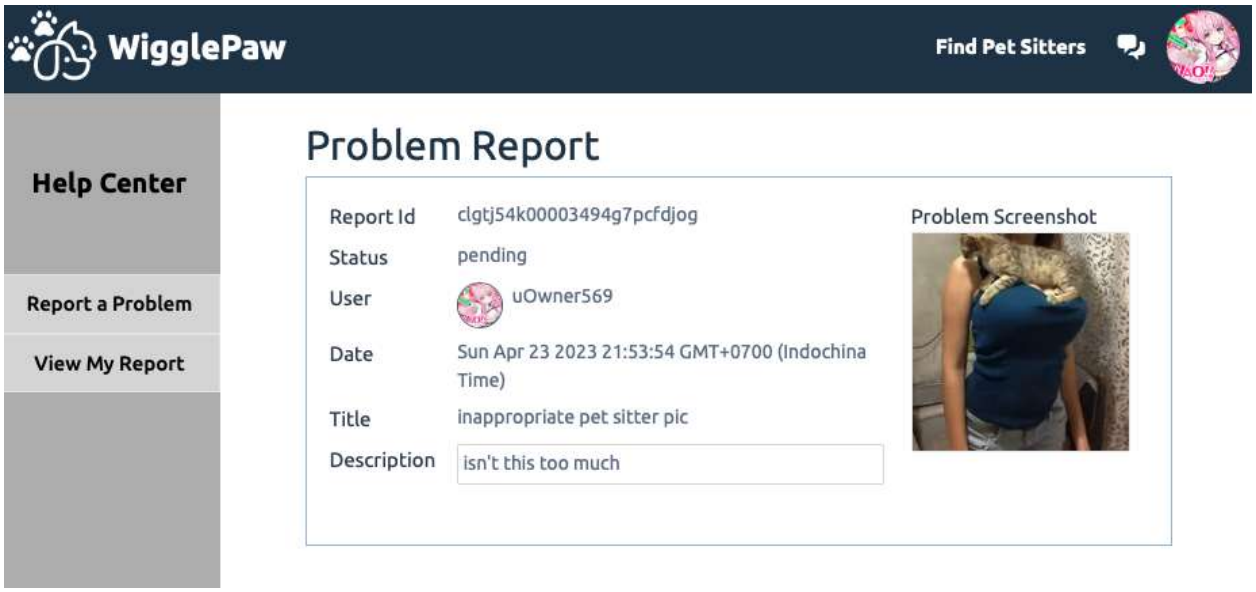
5. Click Submit



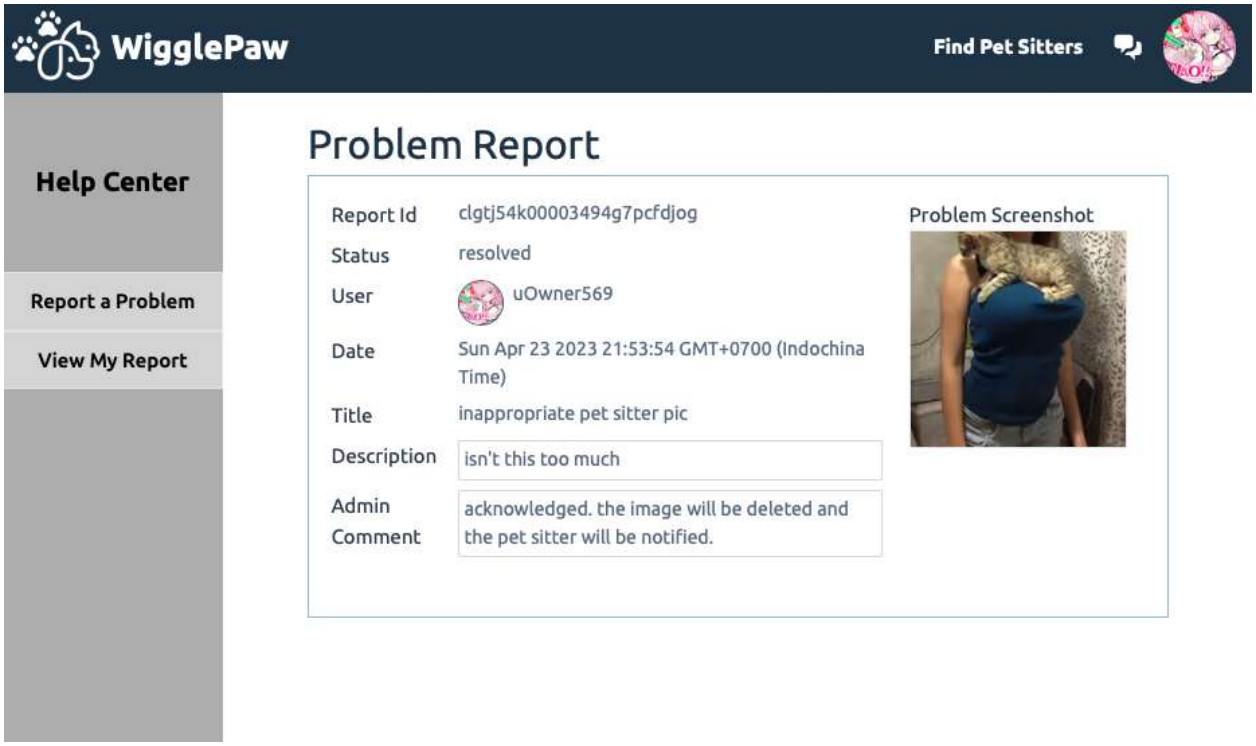
6. To view the submitted reports and their statuses, click on the View My Report button in the side tab.



7. Click on the row to view the detail

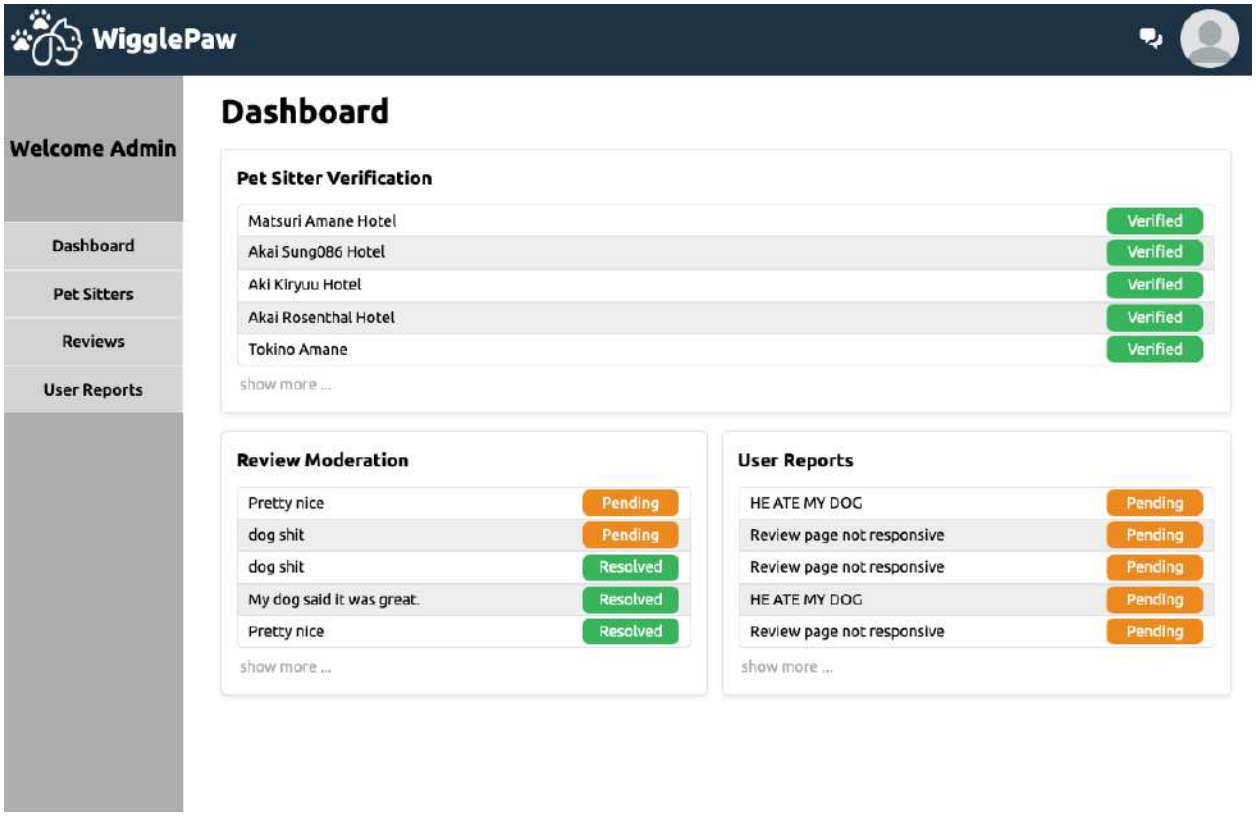


8. When an admin acknowledges and respond, comments will be displayed here

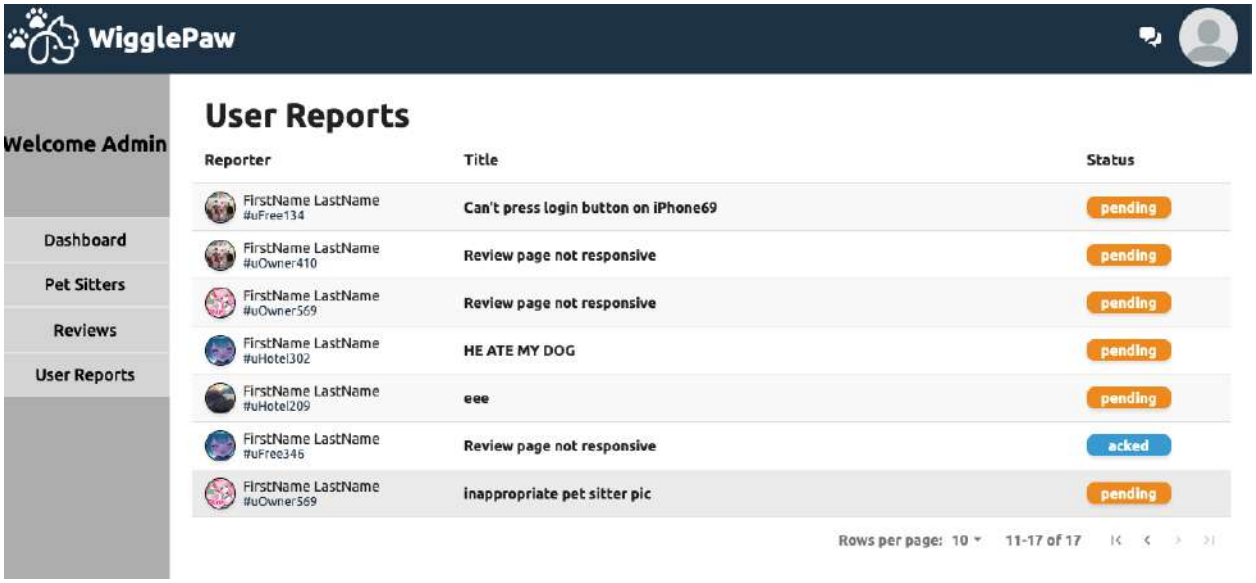


Role : Admin

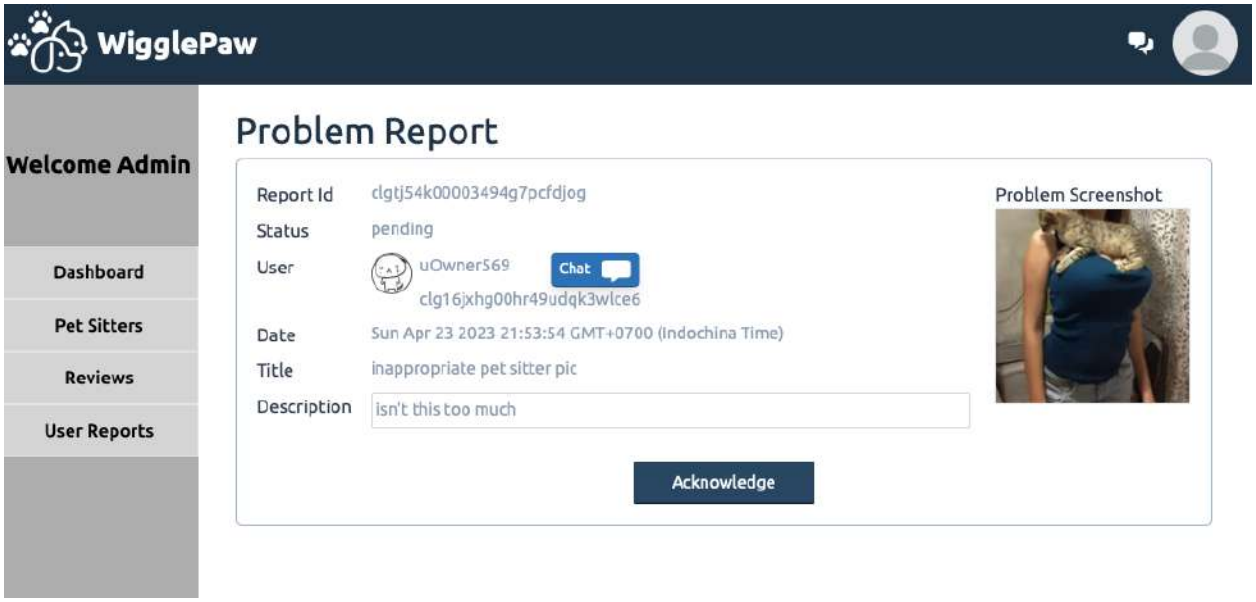
Instructions for managing problem reports submitted by USERs
1. Log in to the system. Then you'll be redirected to the admin dashboard. Click on either the 'User Reports' in the side tab or the User Reports Panel in the center of the page to see all lists of all unacknowledged reports submitted by the users.



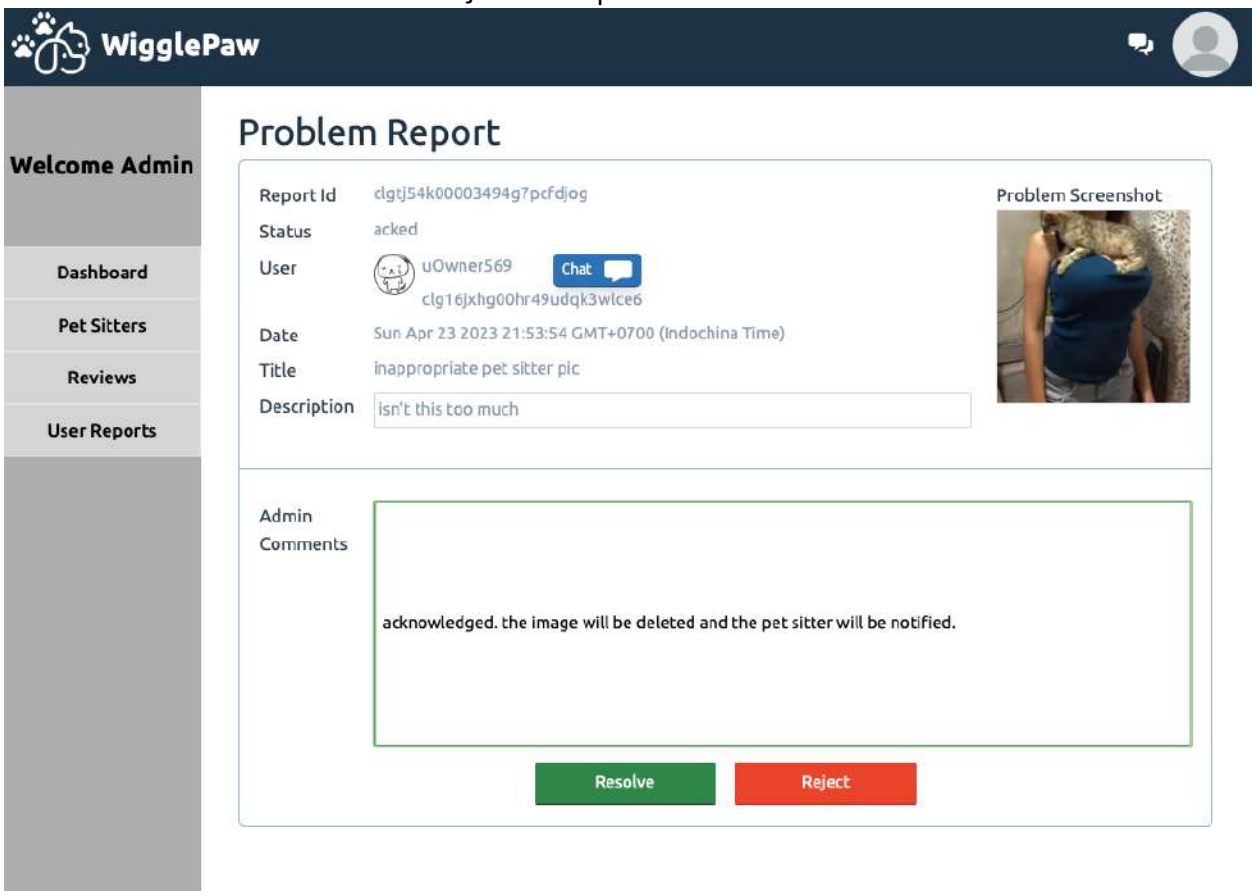
2. Click on a row to manage that report. In this example, the admin clicks on the inappropriate picture report submitted by the USER as mentioned above.



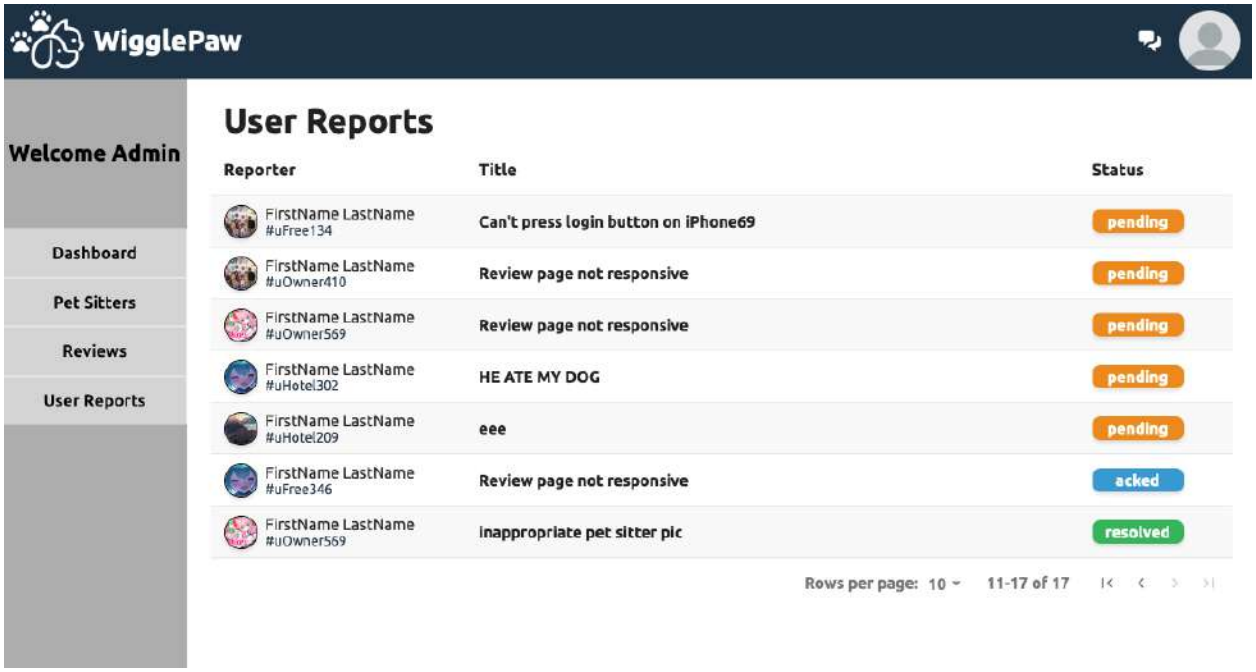
3. Click acknowledge the report. This will inform the submitted user of the updated status and also notify others admins too.



4. Write comments and resolve (or reject the report)



5. The report status will be updated accordingly.


WigglePaw
Welcome Admin
Dashboard
Pet Sitters
Reviews
User Reports
User Reports
Reporter
Title
Status
FirstName LastName #uFree134
Can't press login button on iPhone69
pending
FirstName LastName #uOwner410
Review page not responsive
pending
FirstName LastName #uOwner569
Review page not responsive
pending
FirstName LastName #uHotel302
HE ATE MY DOG
pending
FirstName LastName #uHotel209
eee
pending
FirstName LastName #uFree346
Review page not responsive
acked
FirstName LastName #uOwner569
inappropriate pet sitter pic
resolved
Rows per page: 10
11-17 of 17

Limitations of the Implementation

Register and Login

- The system currently allows users to type more than the character limit of a field in input boxes.
- The system allows user to “paste” into password input box.

Matching

- Only support 9 predefined pet types even though the registration & profile supports user-defined pet types.
- Search results won’t load in the first place. The user must click the search button first.
- Search results don’t show price in number.
- The system does not support finding by “address” or “location”.

Schedule

- Pet in each booking data is only a dummy data, not the real data from the booking.
- Profile pictures for each booking have responsive problems.
- Profile pictures still use a dummy picture, not a real profile picture.

Booking

- It should have the price rate of the pet sister in the booking page for recommending pet owners who want to book.
- There is no confirmation for booking. Therefore, human error is irreversible.

Review

- There is no pop-up box or confirmation for delete or report. Therefore, human error is irreversible.
- The design might be a bit plain.
- Does not show review time. It only shows the date in day-month-year.
- Average rating star cannot show in a part of a star.
- Don’t have a response when a pet sitter writes a review.

Guest

- Some apis supposed to be protected are still public.

Chat

- Unable to send images due to time factor.
- No real time notification.
- Block function won't remove chatroom from chat page.
- Real time chatting depends on the socket server.

Payment

- The system has only been tested in “testing mode”, meaning no actual payment has been performed. As such, there may be unexpected behavior once “real payment” is enabled.
- Currently, our platform fee is set to 0%
- It impossible to mutate card nor bank account details once created
- The transaction receipt that was downloaded does not exactly match the one displayed on the web page in terms of appearance.

Support

- As of now, there has been no external notification regarding the problem, and the supporter needs to visit the webpage to see it.

Deployment

- The deployment platform cannot replicate and so may not scale-well for very large workload
- The object storage (Cloudflare R2) free tier has a high but finite read-write limit
- Deployment and cold start can take a very long time
- Image may take some time to load

Soft copy of all documents

SRS Report

https://mycourseville-default.s3.ap-southeast-1.amazonaws.com/useruploaded_course_files/2022_2/32207/attachment_slots/SRS_Review_Report-467217-16745772080223.pdf

Product Backlog and Sprint1 Backlog

https://view.officeapps.live.com/op/view.aspx?src=https%3A%2F%2Fmycourseville-default.s3.ap-southeast-1.amazonaws.com%2Fuseruploaded_course_files%2F2022_2%2F32207%2Fattachment_slots%2FWigglePawBacklogs-467217-16745771685061.xlsx&wdOrigin=BROWSELINK

Sprint 1 Burndown Chart and Sprint 2 Backlog

https://mycourseville-default.s3.ap-southeast-1.amazonaws.com/useruploaded_course_files/2022_2/32207/attachment_slots/G.2WigglePawBacklog12-467220-16765578188371.pdf

Sprint 2 Burndown Chart and Sprint 3 Backlog

https://mycourseville-default.s3.ap-southeast-1.amazonaws.com/useruploaded_course_files/2022_2/32207/attachment_slots/G.2WigglePawBacklog23-467220-16787274036665.pdf

Sprint 3 Burndown chart

https://mycourseville-default.s3.ap-southeast-1.amazonaws.com/useruploaded_course_files/2022_2/32207/attachment_slots/WigglePawBacklog3-467220-16805395944091.pdf

Design document

https://mycourseville-default.s3.ap-southeast-1.amazonaws.com/useruploaded_course_files/2022_2/32207/attachment_slots/Design_Document_G2_1-466731-16822629409719.pdf