Software Engineering II
Group 3

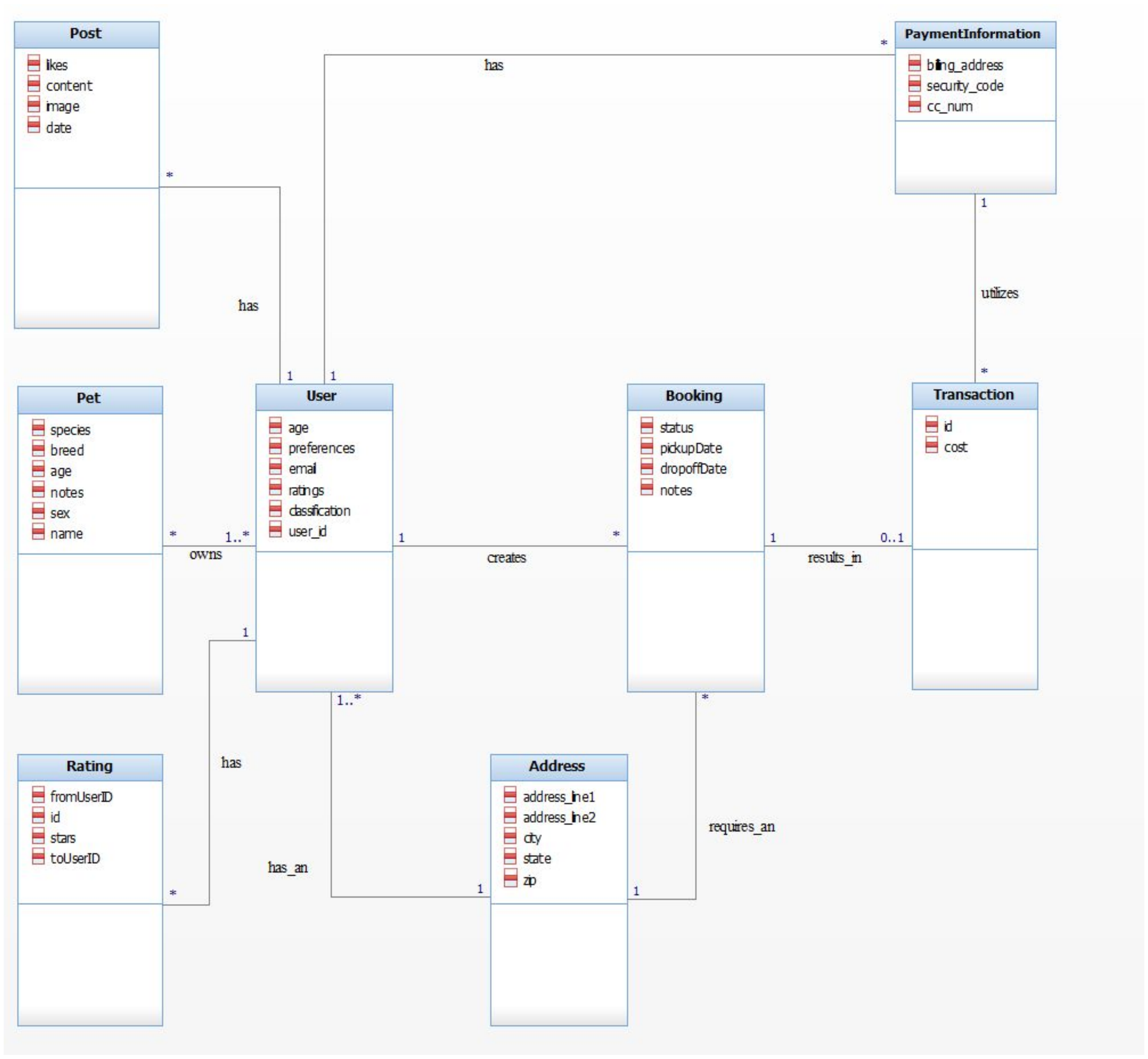
Final Deliverable

Software II Project

Ryan Cave
Annie Mathis
Ejay Mallard
Pearson Reese
Morgan Wesemann

Table of Contents

Requirements Documents



Use Case Modeling - Casual Form

Register Account

Main Success Scenario: The user will provide their information prompted for on the screen. That information being the user's name, email (which will also function as the username), password, and account type (owner/sitter). In the future, a user will be able to upgrade their account to become both a sitter and owner if the wish. The system then generates the account accordingly, and then stores it in the database. The user is now successfully registered by the system.

Alternate Scenarios:

If a user is already registered under the given email, the website will prompt them to login instead of creating a new account.

Match/Hire A Sitter

Main Success Scenario: Pet owner goes to website and navigates to search. The owner will be matched with sitters by filtered criteria from which they can then select one after viewing their rating and profile.

Alternate Scenarios:

If the pet owner returns to the home page, the search is discarded.

If no pet sitters are found within the given criteria, present a response screen and suggest expanding the search criteria.

Rate Other Owners/Sitter

Main Success Scenario: User will go to the website and navigate to the owner or sitter they'd like to rate. They will input information about their experience and give a star rating. They will submit the rating (1-5 stars) and optional text review and it will be sent to an admin for review. Owners may not rate other owners, and Sitters may not rate other sitters.

Alternate Scenarios:

Client returns to the home page. Rating is discarded.

Client neglects to put a star rating. They are prompted to input one before they can submit the rating.

Use Case UC01: Register Account

Scope: Pet Sitting Web Application

Level: User Goal

Primary Actor: User

Stakeholders and Interests:

- User: Wants to be able to quickly create a profile along with setting their prospective preferences depending on whether they are an owner or sitter.

Preconditions: The user registering does not already have an account on the site.

Postconditions: The user's account was created and their information stored in the database.

Main Success Scenario:

1. User selects the option to create a new profile/account.
2. The user specifies whether they are a pet sitter or owner.
3. System displays the appropriate fields for the account type specified.
4. The pet owner/sitter will provide their personal/payment information including (for sitters) experience and availability, and (for owners) pet type and preferences.
5. The system generates the account accordingly and then stores it in the database.
6. The user is successfully registered by the system.

Extensions (Alternative Flows):

4a. The email given by the user is already associated with a registered account:

1. The website will prompt them to login instead of creating a new account

4b. A required information field is left empty:

1. Highlight the respective fields and prompt the user to enter the missing information

Special Requirements:

- Text displayed on the screen must be readable.

Technology and Data Variations List:

4a. Credit account information entered by keyboard

Frequency of Occurrence: Any time there is a new user

Open Issues: N/A

Use Case UC02: Match/Hire a Sitter

Scope: Pet Sitting Web Application

Level: User Goal

Primary Actor: User

Stakeholders and Interests:

- User (Sitter): Wants to be able to be matched to owners so that they might hire them
- User (Owner): Wants to be able to find relevant sitters that would be valid options to take care of their pets

Preconditions: The User (sitter) and the User (owner) both possess accounts on the system.

Postconditions: The User (owner) found a suitable User (sitter) that they want to hire.

Main Success Scenario:

1. User (owner) logs into the system.
2. User (owner) navigates to the search feature.
3. User (owner) prompts the system to run a search on sitters.
4. User (owner) is presented with the results of the search.

Extensions (Alternative Flows):

1a. User does not have an account with the system.
  1. User will be directed to create an account.

4a. There are no results of the search.
  1. User will be prompted to search again or change the parameters of the search.
  2. Go to step 4 of Main Success Scenario.

Special Requirements:

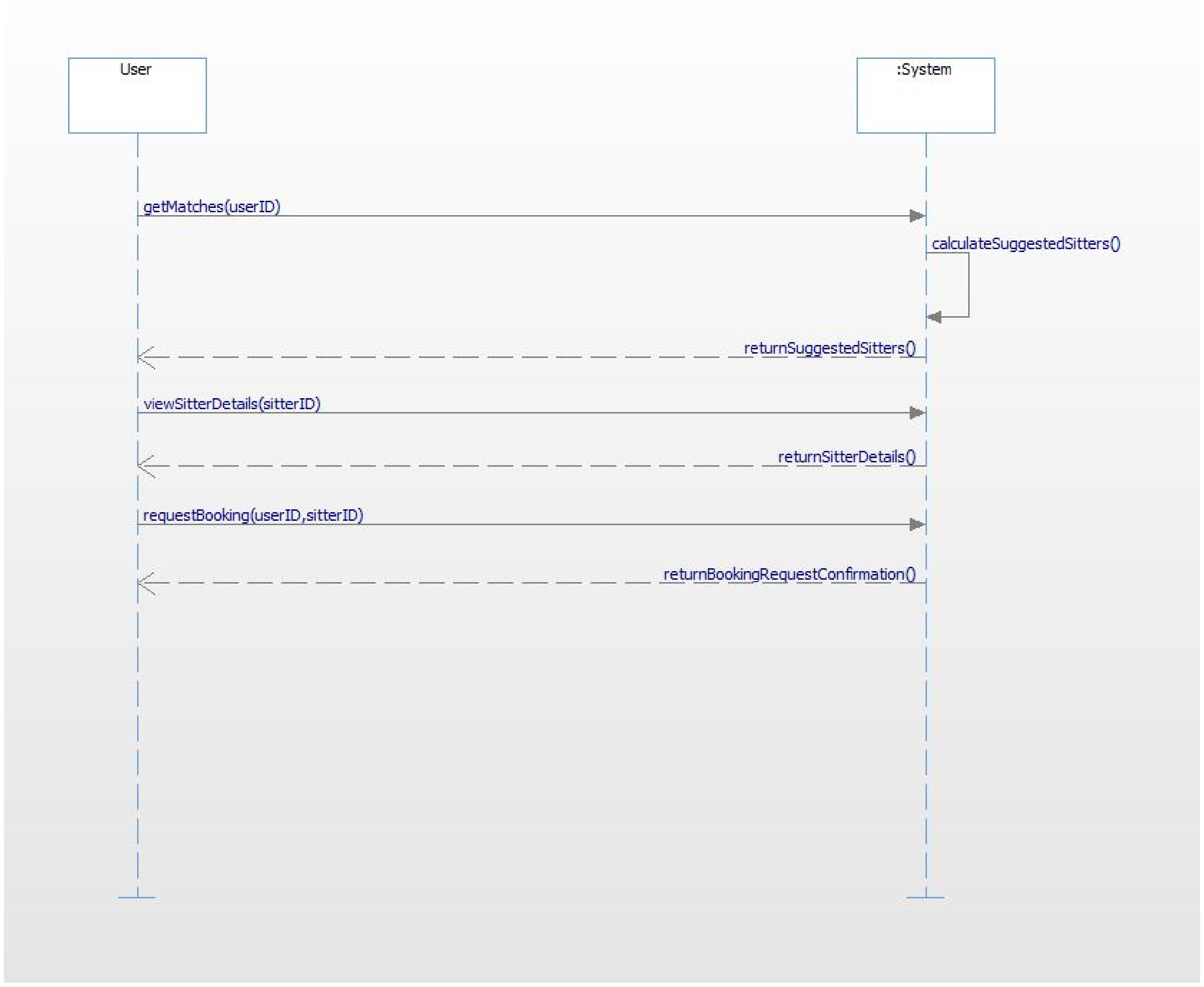
- Text displayed on the screen must be readable.

Technology and Data Variations List:

N/A

Frequency of Occurrence: Any time a user starts a new search.

Open Issues: *N/A*


User
:System
getMatches(userID)
calculateSuggestedSitters()
returnSuggestedSitters()
viewSitterDetails(sitterID)
returnSitterDetails()
requestBooking(userID,sitterID)
returnBookingRequestConfirmation()

<u>**Operation Contracts for Match/Hire a Sitter UC:**</u>

<u>Contract CO1</u>: **getMatches(userID)**

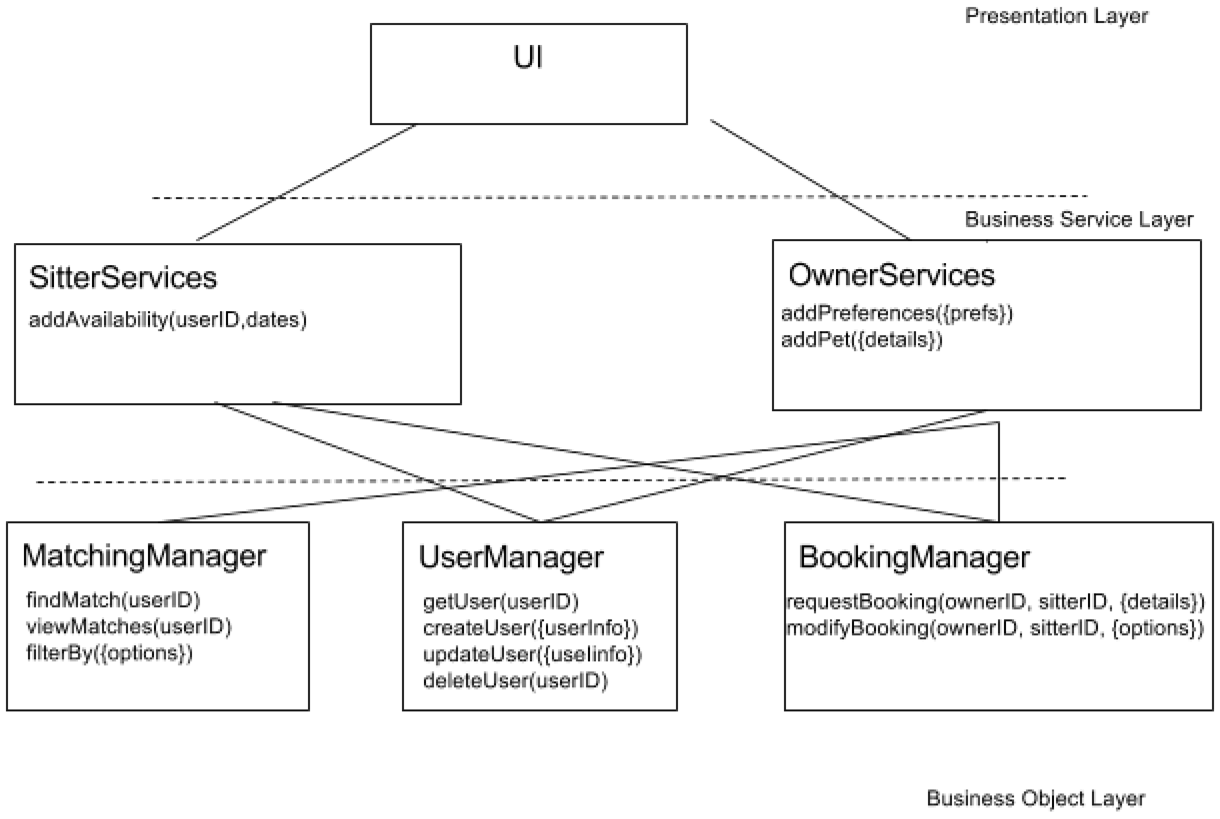
Preconditions:
a) The user has been authenticated.
Postconditions:

a) The suggested sitters are calculated by the system based on a criteria and a list of them is returned to the user.

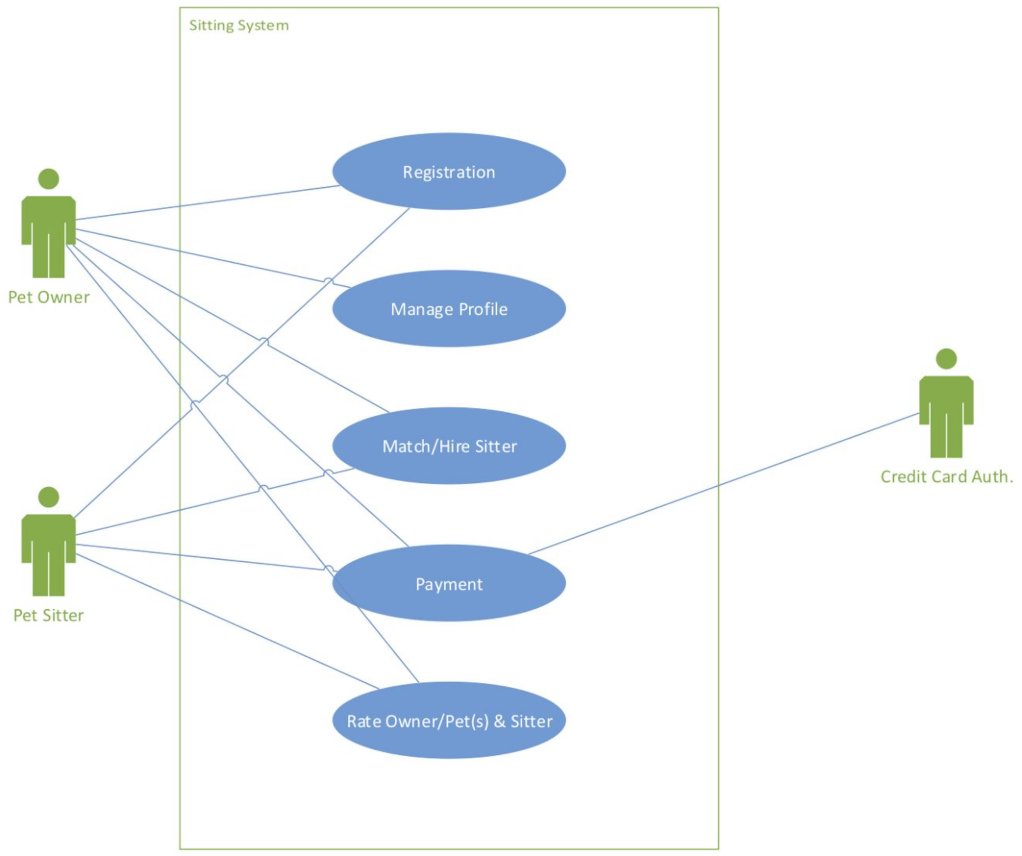
Contract CO2: **viewSitterDetails(sitterID)**
Preconditions:
a) The user has been authenticated.
Postconditions:
a) The system returns the details for the designated sitter from the matches list.

Contract CO3: **requestBooking(userID,sitterID)**
Preconditions:
a) The user has been authenticated.
b) The sitter is available for the booking.
Postconditions:
a) The booking request is registered in the system and notifies the sitter, who can accept or decline the booking.

Static Architecture Model



Use Case Diagram



Design Overview

Group 3 Design/Architectural Patterns

- At this point of the project, we have utilized the architectural pattern Model-View Controller to develop user-interfaces, which has enabled us to make changes to the UI quickly and easily without impacting core functionality.

- On the front-end, we have implemented many principles laid out in GRASP, including but not limited to high cohesion and low coupling. This allows us to make modules that are stand-alone and easy to move about the project without impacting functionality elsewhere on the front-end.

- We are using a facade pattern for the communication between the back-end and the front-end. This ensures that user interactions with the front-end have no possibility of performing unauthorized or unintended actions. The front-end has no knowledge of how operations on the back-end are conducted, it simply asks for information or provides information and the back-end handles it.

Sequence Diagrams

Test Plan

Tempeturs – Group 3 Test Plan

Table of Contents

Introduction

Purpose

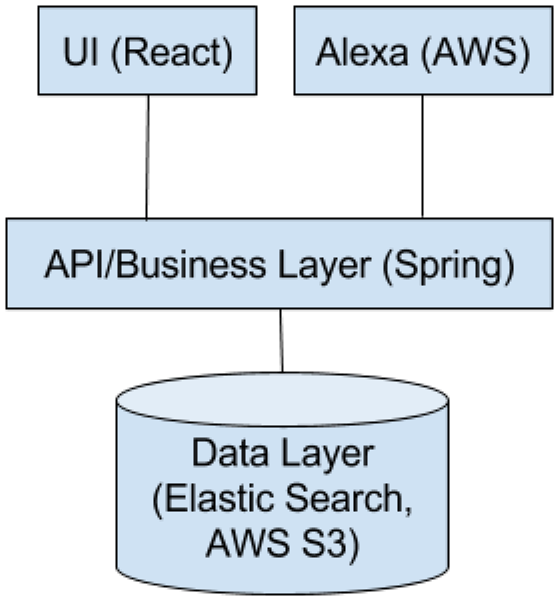
This Test Plan document for the Tempeturs Site supports the following objectives:

1. Identify the software components involved in Milestone 3 that should be tested
2. Specify the recommended requirements for testing
3. Describe the order and methods of the testing demonstration for Milestone 3
4. List the artifacts created during Milestone 3

Background

The Tempeturs site is an online pet sitter finder that owners and sitters alike can register for and be matched with suitable sitters. Owners and sitters can manage profiles, add pets, and other relevant information for requesting and accepting pet sitting bookings (address, payment info, etc). There will be a search and recommendation system based on preferences, availability, and ratings of sitters. The application is designed with a multi-layer architecture:

- UI - React and Node.js.
- Alexa Skill - Node.js and AWS Lambda
- API/Business Layer - Java Spring Framework
- Data Layer - Elastic Search (database), AWS S3 (images)


UI (React)
Alexa (AWS)
API/Business Layer (Spring)
Data Layer (Elastic Search, AWS S3)

Scope

The Tempeturs web application will be unit tested and system tested. Unit tests are implemented to establish the integrity of created functions and service layer calls. System tests will establish that the functionality of the application matches what a user would expect. Additionally, group 3 will conduct a demonstration of the expected functionality of Tempeturs for Milestone 3.

Documentation artifacts

The following are documents are found in the project repo (and Piazza):

- Domain Model
- Architectural Model
- Use Cases
- Use Case Diagram
- Vision document

Requirements for Testing

The list below describes the items that have been identified, by the group, as targets for testing:

Database Testing

- Verify that user information including profile information, pet information, payment information, and address information can be entered and retrieved.
- Verify that booking information can be entered and retrieved.

API Testing

- Verify that a user can be created, retrieved, updated, and deleted.
- Verify that a booking can be created, retrieved, updated, and deleted.
- Verify that a user can authenticate using a username and password.
- Verify that a user can update their password.
- Verify that a user can search for recommended sitters.

User Interface/Functional Testing

- Verify that all pages are routed correctly.
- Verify that all pages exist.
- Verify that all pages display correct information.
- Verify that data entry fields ar functional.
- Verify that data entry fields validate information such as phone numbers, email addresses, and payment information.
- Verify that a user can sign in with credentials.

- Verify that a user can register for an account.
- Verify that a user can update account information.
- Verify that a user can request, accept, or cancel a booking.
- Verify that a user can search for recommended sitters.

Usability Testing

- Verify that a user can sign up without assistance.
- Verify that a user can search for recommended sitters without assistance.
- Verify that a user can request, accept, or cancel a booking without assistance.

Security Testing

- Verify that a user cannot access another user's data.
- Verify that a not-logged in user cannot access any user's data.

Configuration Testing

- Verify that website works in multiple browsers such as Chrome and Firefox.

Test Cases

Test Case 0

- Description:
    - Case 0 tests the main success scenario for the Register use case.
- Precondition:
    - The email address for the user does not already exist in the system.
- Test Inputs:
    - User X types correct information in the registration field (name, email, sitter/owner)
- Expected Results:
    - Account for user X is created and they are logged into the system
- Actual Results:

Test Case 1

- Description:
    - Case 1 tests alternative flow 4a of the Register use case where an email address already exists in the database.
- Precondition:
    - The email address for the user X already exists in the system.

- Test Inputs:
  - User X types correct information in the registration field (name, email, sitter/owner)
- Expected Results:
  - Account for user X is not created
  - System notifies user X that account already exists
  - System prompts user X to login
- Actual Results:
  - Account for user X is not created

Test Case 2

- Description:
  - Case 2 tests alternative flow 4b of the Register use case where a required information field is left empty.
- Precondition:
  - The user does not already exist in the system
- Test Inputs:
  - User X types correct information in the registration field (name, email, sitter/owner)
- Expected Results:
  - Account for user X is not created
  - System notifies user X that account already exists
  - System prompts user X to login
- Actual Results:
  - Account for user X is not created

Test Case 3

- Description:
  - Case 3 tests the ability for a user to successfully login.
- Precondition:
  - The user X already exists and is not logged in.
- Test Inputs:
  - User X types correct information in the login fields (email, password)
- Expected Results:
  - The account details are validated with the database, and the user is logged in. The user can refresh the page and will stay logged in.
- Actual Results:

Test Case 4

- Description:
  - Case 4 tests the when a user provides invalid credentials when trying to login.
- Precondition:
  - The user X already exists and is not logged in.
- Test Inputs:
  - User X types incorrect information in the login fields (email, password)
- Expected Results:
  - The account details are validated with the database and rejected
  - The user is not logged in.
  - The user is notified their login did not work
- Actual Results:

Test Case 5

- Description:
  - Case 5 tests the ability for a user to add a pet.
- Precondition:
  - The user X already exists and is logged in.
- Test Inputs:
  - User X types correct information in the add pet fields
- Expected Results:
  - A pet is added to the user and shows up on the profile page for user X
- Actual Results:

Test Case 6

- Description:
  - Case 6 tests the main success scenario of the Rate a User use case.
- Precondition:
  - The user X giving the rating already exists and is logged in.
  - The user Y receiving the rating exists
  - The user Y receiving the rating is different than the user giving the rating
- Test Inputs:
  - User X types correct information in the add rating fields for user Y
- Expected Results:
  - A rating is added to user Y and shows up on the profile page for user Y

- Actual Results:

Test Case 7

- Description:
  - Case 7 tests the ability for a user to initiate and make a booking.
- Precondition:
  - The user X initiating the booking already exists and is logged in.
  - The user Y receiving the booking initiation exists
  - The user Y receiving the booking initiation is different than the user giving the booking initiation
  - User Y is available for the date and length of the booking
  - User Y is a sitter
  - User X is an owner
- Test Inputs:
  - User X types correct information in the initiate booking fields for user Y
- Expected Results:
  - A booking is initiated between user X and user Y and shows up on the bookings page for both users as requested
  - User Y is notified they have a new booking initiation and can choose to accept or decline the booking
- Actual Results:

Test Case 8

- Description:
  - Case 8 tests when a user initiates a booking where the sitter is not available for the dates requested.
- Precondition:
  - The user X initiating the booking already exists and is logged in.
  - The user Y receiving the booking initiation exists
  - The user Y receiving the booking initiation is different than the user giving the booking initiation
  - User Y is not available for the date and length of the booking
  - User Y is a sitter
  - User X is an owner
- Test Inputs:
  - User X types correct information except for date and length in the initiate booking fields for user Y
  - User X enters a date where the sitter is not available
- Expected Results:

  - A booking is not initiated between user X and user Y
  - User X is notified the sitter is not available on the specified date
- Actual Results:

<u>Resources</u>

Members

- Project Manager: Pearson Reese
- Requirements Engineer: Ejay Mallard
- Design Engineer: Annie Mathis
- Quality Assurance: Ryan Cave
- Project Librarian: Morgan Wesemann

System

- Frameworks: Java Spring (API), Node.js + React (frontend)
- System Database: Elastic Search on Heroku
- Team Repository: https://gitlab.com/fall17-group3
- Continuous Integration: Gitlab CI to Heroku

Installation Guide

To install this project, first make sure you have Node.js and npm installed.

Then clone the frontend repo:

Frontend: https://gitlab.com/fall17-group3/tempeturs-frontend

Run “npm install” to download dependencies.
Run the frontend locally by running “npm run dev.”

Note that the project uses the live Heroku app for the backend, so the backend repo is not required to run the website locally.

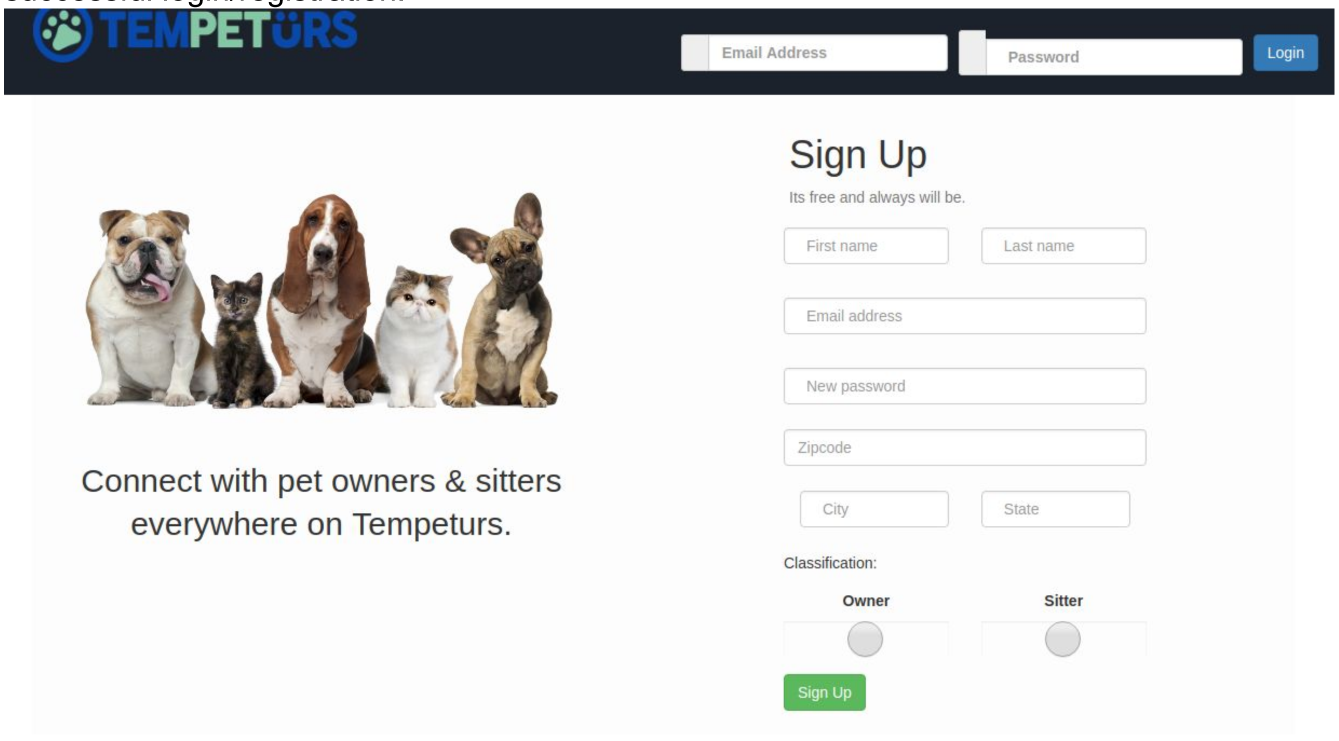
Optional:
To clone the backend:

Backend: https://gitlab.com/fall17-group3/group-3

Import the backend using the gradle project import tool using the guide we were given in class.

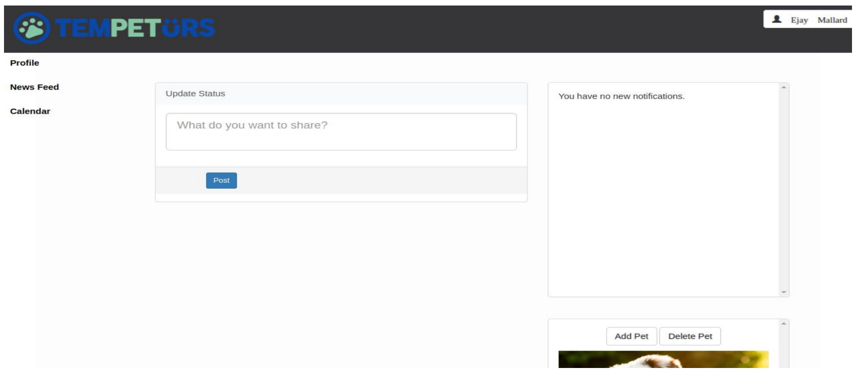
User Manual

Login/Signup:
Upon reaching our landing page, you will be given the option to sign up or login. If logging in, simply provide the email address and password to your associated account else use the sign in panel to provide your user information. You will be redirected upon successful login/registration.



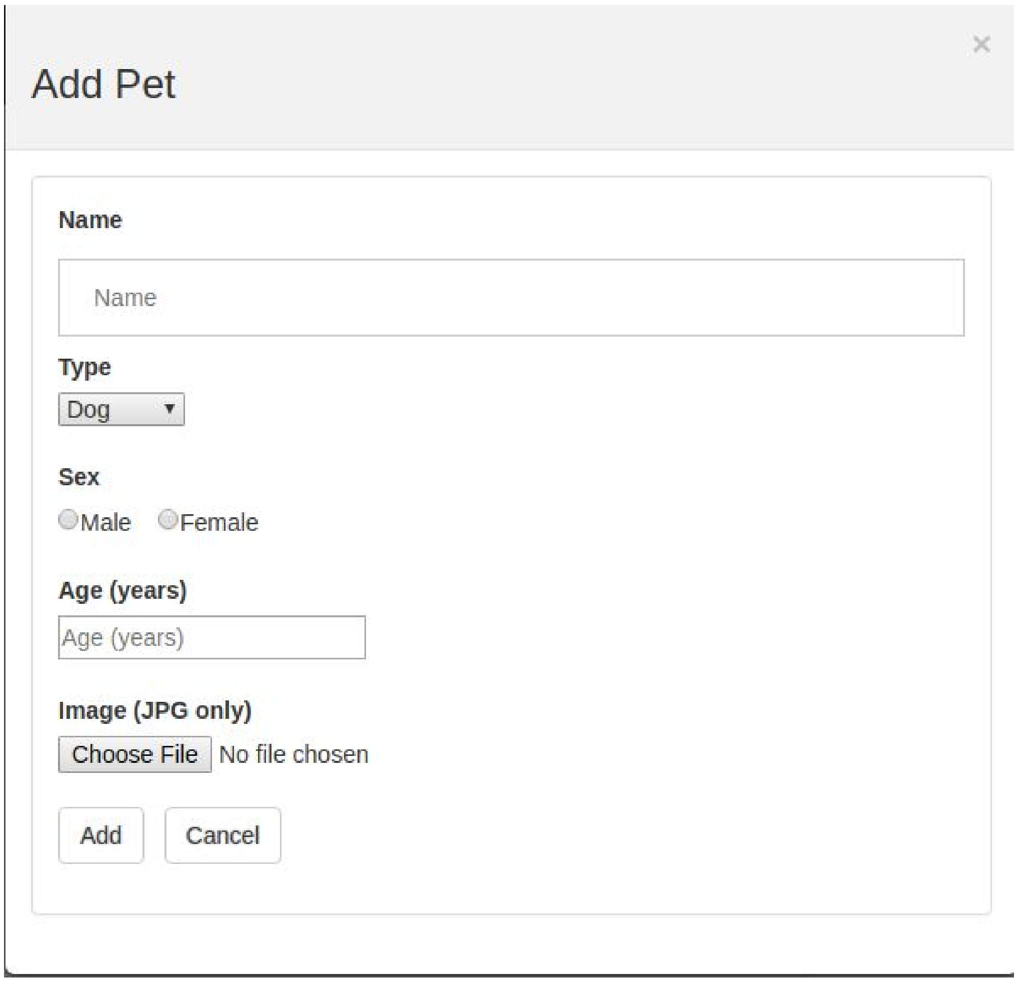
Connect with pet owners & sitters everywhere on Tempeturs.

Dashboard:
Once logged in, you will be redirected to your user dashboard where you will have the opportunity to begin your user experience. Here you will see a side panel with options for "Profile", "Calendar"/"Find Sitter", and "News Feed": You will see on the right your notifications panel and pets panel.

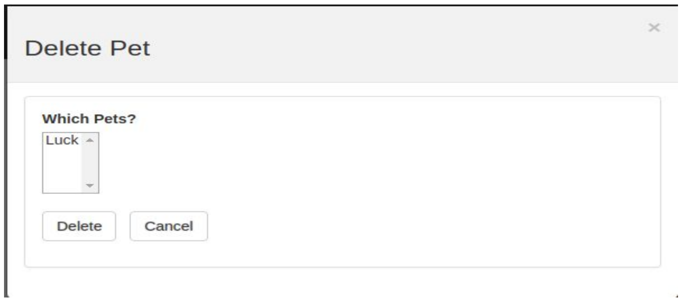


You have no new notifications.

Add Pet:
To add a pet, simply click the “Add Pet” button and a modal will appear. Provide all necessary and required information about your pet including a pet image, and your pet will be successfully added.

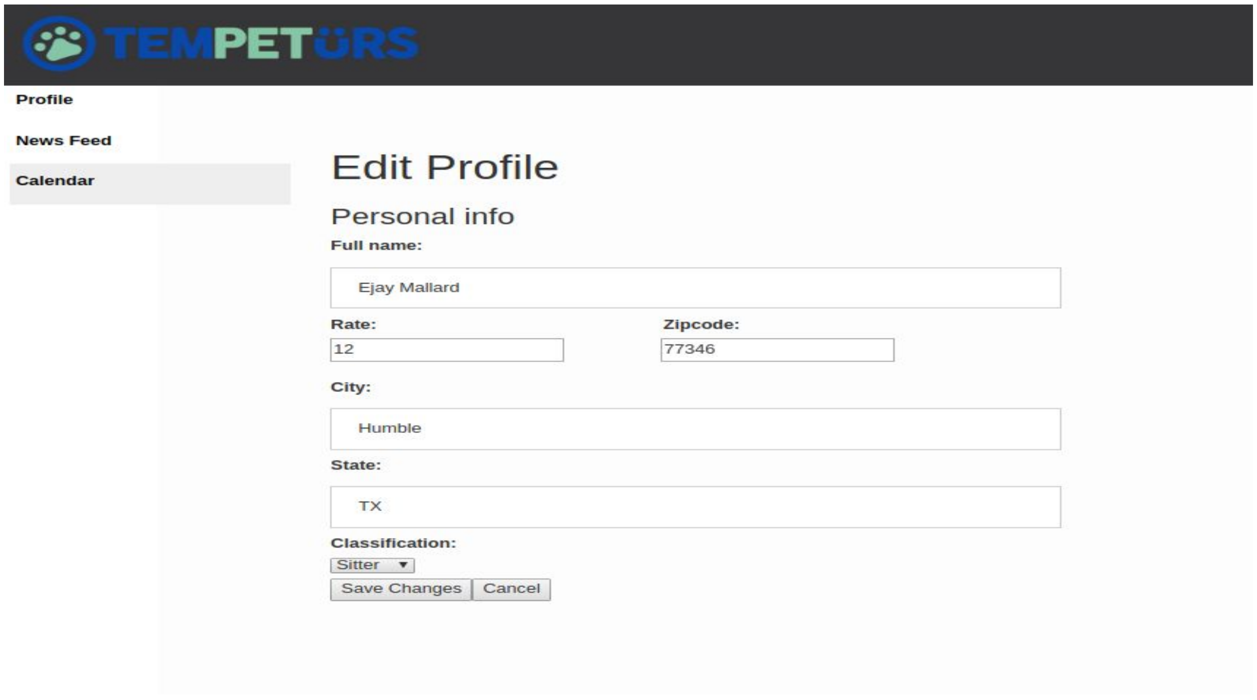


Delete Pet:
To delete a pet, simply click the “Delete Pet” button and a modal will appear. Select the pet from your list of pets and select ‘Delete’ and the pet will be removed accordingly.



Which Pets?
Luck

Edit Profile:
To edit your profile and/or change classification, simply select your name in the upper right hand corner of the page, and select 'Edit Profile' option where you will be redirected to a page that allows you to modify your user information.



Edit Profile

Personal info

Full name: Ejay Mallard

Rate: 12

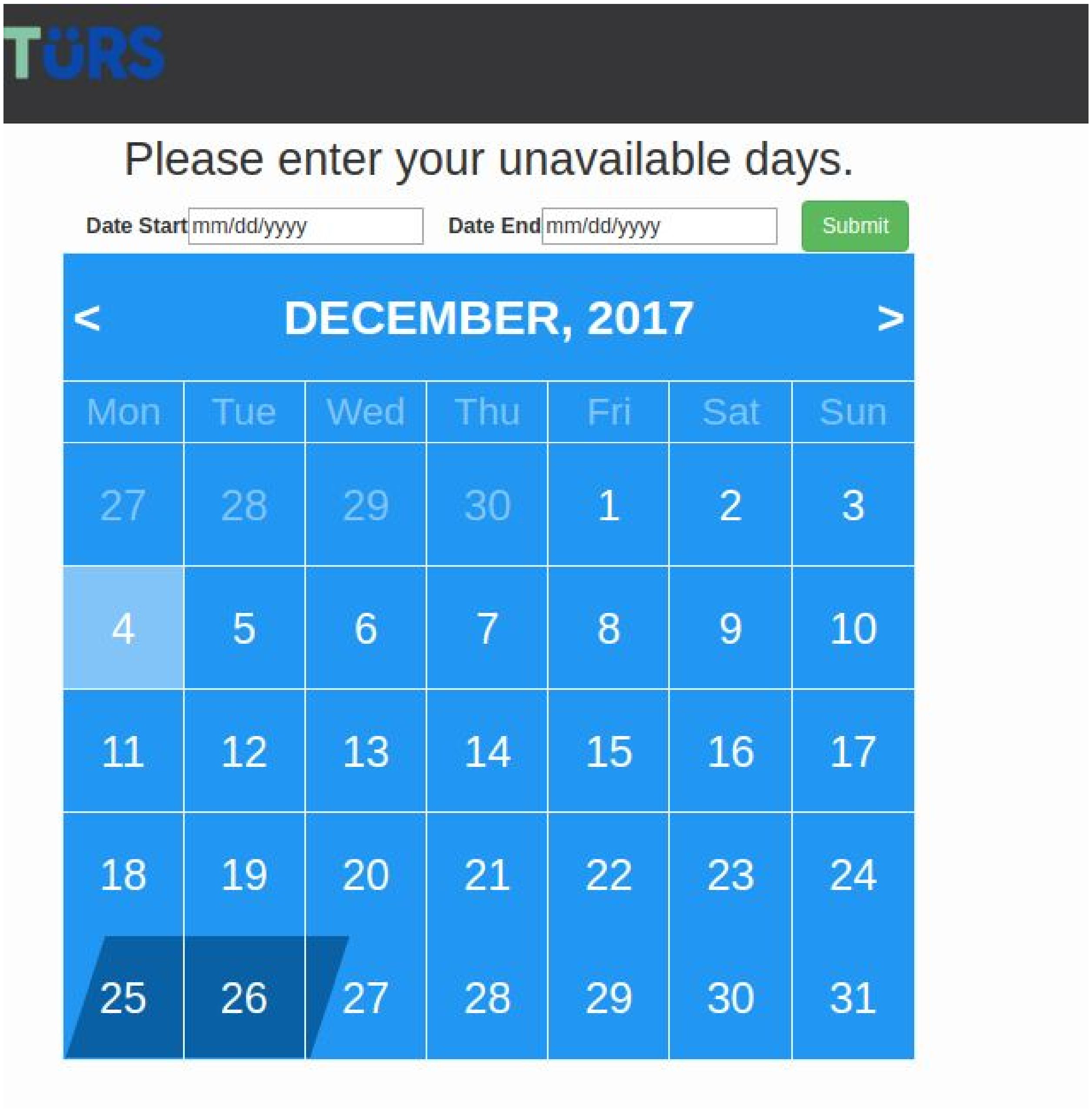
Zipcode: 77346

City: Humble

State: TX

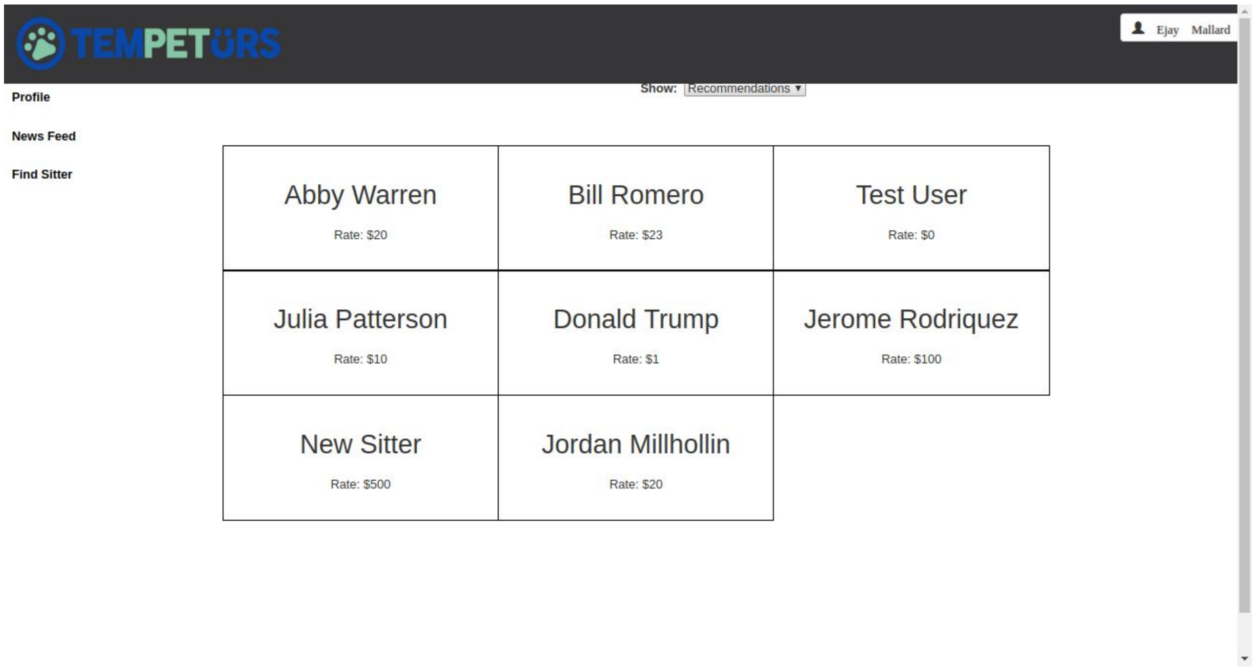
Classification: Sitter

Update Unavailability:
To update your unavailability, simply select the "Calendar" option on the sidebar to the right. You will be redirected to your calendar page (if your classification is a sitter). Input the start and end dates of your unavailability and select submit. The page will refresh, updating the calendar with dark blue icons indicating that those days are unavailable.



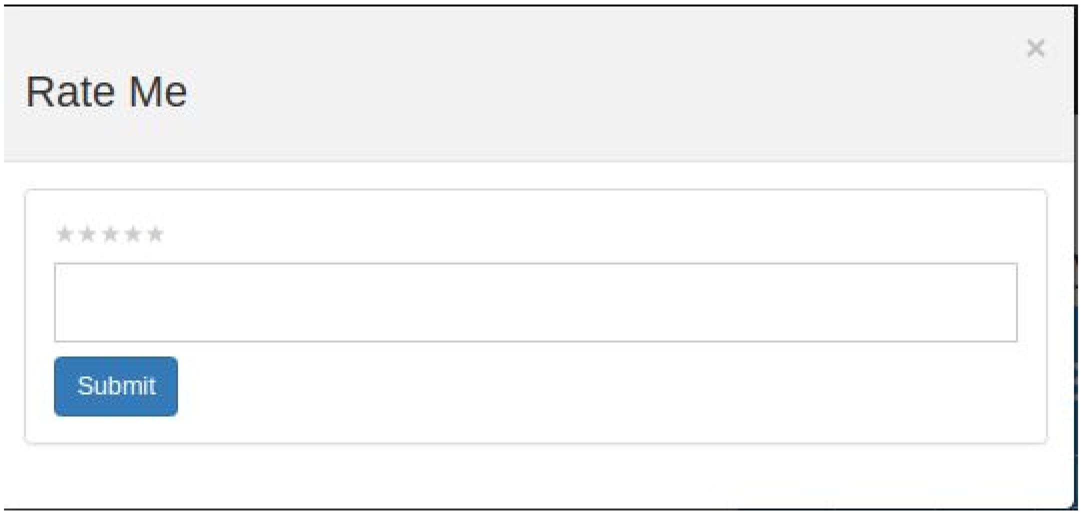
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Find a Sitter:
To find a sitter, first ensure your classification is of type Owner. This can be changed via the Edit Profile option as detailed previously. Once a owner, select the ‘Find Sitter’ option in the left sidebar and you will be redirected to a page that provides you with sitters.



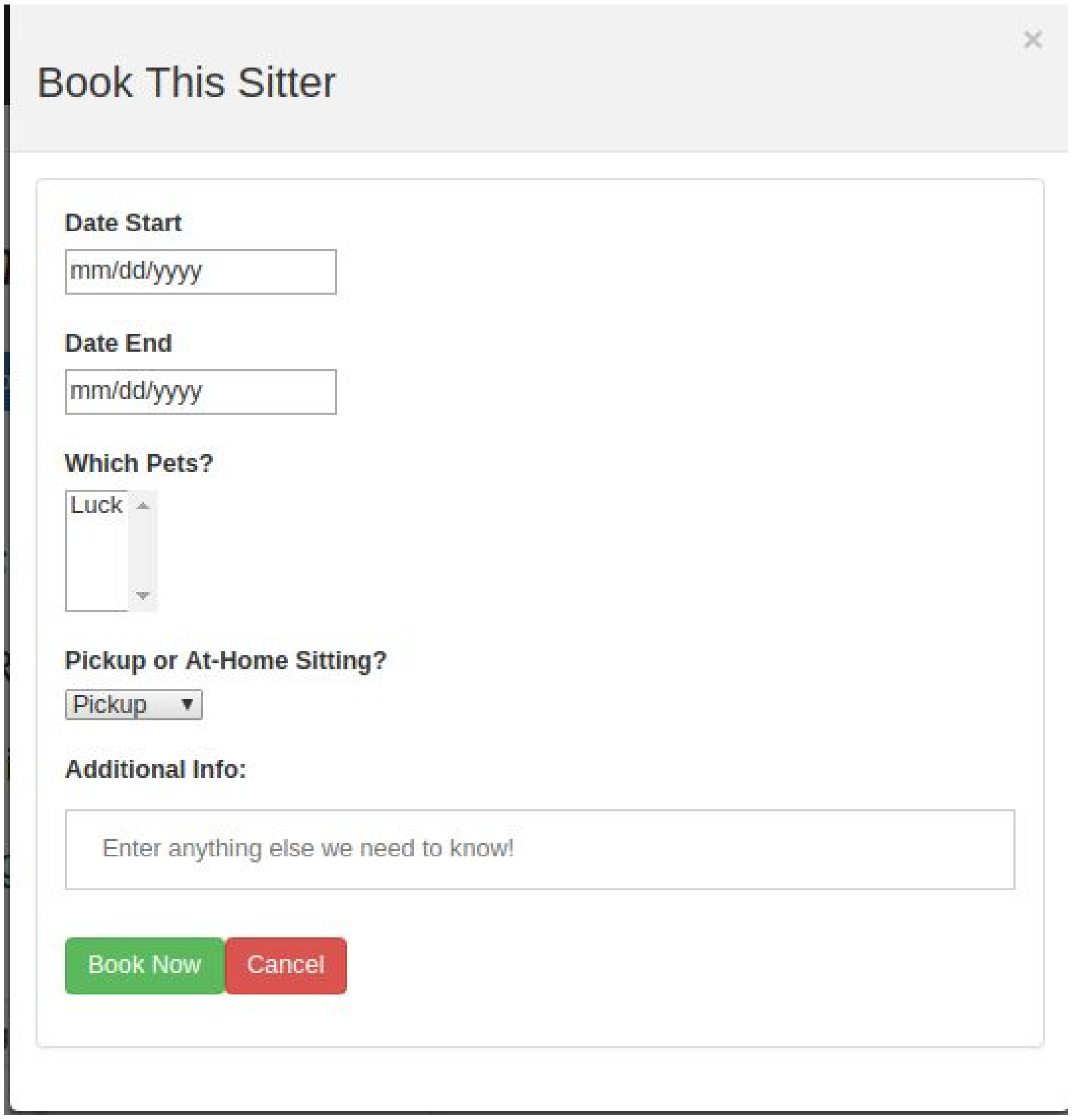
Abby Warren Rate: $20	Bill Romero Rate: $23	Test User Rate: $0
Julia Patterson Rate: $10	Donald Trump Rate: $1	Jerome Rodriquez Rate: $100
New Sitter Rate: $500	Jordan Millhollin Rate: $20	

Rate a Sitter:
Once you have selected a sitter, you may rate him or her by selecting the ‘Rate Me’ button their profile page. This will display a modal with adequate stars and an optional comment box. Select the rating your wish to provide and detail any comments then select submit. The page will refresh, and you will notice that the rating has been successfully added.



Book a Sitter:
Once you have selected a sitter, you may book him/her by selecting the ‘Book Me’ button on their profile page. This will display a modal that provides you with options for

start to end date, pet selection, and location of appointment with an additional comments box if needed.



Book This Sitter

Date Start

mm/dd/yyyy

Date End

mm/dd/yyyy

Which Pets?

Luck

Pickup or At-Home Sitting?

Pickup

Additional Info:

Enter anything else we need to know!

Contribution Summary

Ryan Cave - Primarily served in role ensuring Quality Assurance. Assisted in initial setup with elasticsearch, initial backend endpoints, etc. Remained flexible and assisted on back and front end development as needed over the course of the semester. Assisted in documentation/deliverables, testing, prototyping design elements, and organizing additions to the project.

Ejay Mallard - Front end developer; Responsible for working on the front end utilizing React, a Javascript library, which included creating landing page, page redirection, login/signup capability, and various other user interface components. Tailored the user experience to come as close to an easy-to-use interaction as possible

Annie Mathis - Front end developer; Assisted Ejay with using React to create the user interface and connect to the database. Created mechanisms for booking sitters and rating sitters, and implemented the notifications system. Did research on tools Ejay and I could use to enhance our experience using React.

Pearson Reese - Mostly worked week to week on the deliverables with Morgan doing a lot of the project documentation and ensuring our design met the original requirements. Throughout the semester, I also updated the tasks on Trello and kept all the documentation organized in our Google Drive.

Morgan Wesemann - Was the Backend API developer. Worked on backend functionality including user management, pet management, login, authentication, ratings, booking management, notifications, file upload and retrieval in AWS. Was part of setting up Gitlab CI for frontend and backend. Worked on deliverables throughout the semester.

Group Meetings Summary

Notes from Mentor Meeting on 9/5

Questions Asked w/Answers:

1. What is ThymeLeaf and are we using it?
    a. ThymeLeaf is an HTML templating engine for Spring. It was put in for the demo set up and does not have to be used.
2. Does the backend have to be built to build and run the front end?
    a. The backend was coupled with the frontend, but we are allowed to separate the two for independent development.
3. Where will Elastic Search be hosted?
    a. Elastic Search will be hosted on Heroku. They have an integration for it to help get set up quickly.
4. How will continuous integration be set up?
    a. Continuous integration will be set up using GitLab and Heroku.
5. How to set up project in IntelliJ?
    a. See guide posted by Annie on Piazza.

Things to Research Into:

1. CI with Heroku and GitLab
2. Elastic Search with Heroku

*Important: Reschedule Thursday 9/14 and 9/28 meetings if possible due to schedule conflicts with Morgan and Ejay

*Try and watch time, the 20 minutes went very fast

Group Meeting - 9/12/17

The group met to finish the original project setup. Getting acquainted to the technology stack. Further discussed task delegation among the group.

Group Meeting - 9/13/17

The group met to complete the first milestone. We tested our CI with the frontend repo and deploying to Heroku. We scaffolded the pages and used CI to push to heroku. Discussed Alexa skill and created template skill. Worked on CI for backend but ran into issues building and running on Heroku. Will ask mentors for help in meeting tomorrow.

Mentor Meeting - 9/14/17

Demoed current product to mentors, including frontend on Heroku, continuous integration for the frontend, Amazon Alexa skill, and hitting backend API from Postman. Annie had to miss due to a physics exam.

Questions Asked w/Answers:
Can we have help cleaning up our GitLab repos (permissions to delete)?
Christopher deleted them for us
Looking for help for CI with backend + Heroku deploy
Mentors are working on update that Gradle build file
Why is it called Tempetūrs?
Temporarily the Pet is Yours

Mentor Meeting - 9/14/17

We demoed current product to mentors, including frontend on Heroku, continuous integration for the frontend, Amazon Alexa skill, and hitting backend API from Postman. Annie had to miss due to a physics exam.

Group Meeting - 9/18/17

The group met to complete the Credera deliverable. We designed/drew out our ElasticSearch indexes for the project. We then finished connecting ElasticSearch and added in some test data for the mentor meeting demo.

Mentor Meeting - 9/19/17

We started out the meeting discussing what we did that week and what we were looking to work on in the coming week. We then discussed our design/decisions for our ElasticSearch indexes and showed being able to retrieve data from ElasticSearch.

They also helped us figure out our issue with deploying our backend, so our backend is now live on Heroku.

Mentor Meeting - 9/28/17

We started out the meeting discussing what we did that week and what we were looking to work on in the coming week. Morgan and Ryan arrived a few minutes late due to other responsibilities; Ejay had class, so he missed the meeting all together. We then discussed the possibility of switching to using Express with the mentors and Dr. Song. It was decided that we would stick with Spring. Following this, the mentors helped us fix our issues connecting to ElasticSearch.

Mentor Meeting - 10/3/17

We started out the meeting discussing what we did that week and what we were looking to work on in the coming week. We discussed that we had gotten ElasticSearch connected. We didn't really have many questions to ask, so the meeting ended a few minutes early.

Mentor Meeting - 10/10/17

We started out the meeting discussing what we did that week and what we were looking to work on in the coming week. We discussed that we had gotten ElasticSearch connected. We didn't really have many questions to ask, so the meeting ended a few minutes early.

Group Meeting - 10/12/17

Met during class to discuss the next few weeks of the project and renewed focus on the Alexa skill that will be a part of the next couple milestones and deliverables. Most of the group will work on frontend so that can get built out quickly. The goal is to have most of the APIs ready next week.

Mentor Meeting - 10/17/17

We started out the meeting discussing what we did that week and what we were looking to work on in the coming week. Ryan was unable to make it due to a doctor's appointment. We discussed that we had finished the majority of the API. We were having one minor issue with the backend though, but the mentors were able to help us resolve it. Following that, we showed off the progress Anni/Ejay had made on the front end. Lastly, we ended the meeting with Morgan demonstrating the functionality of our Alexa skill that we have done so far and that concluded the meeting.

Group Meeting - 10/17/17

Met during class to discuss the next few weeks of the project and rediscussed the overall design of our application and how we want it to look. We did this in order to make sure we were all on the same page. Most of the group will continue to work on the frontend so that can get everything built out quickly. The goal is to have most of the front end ready next week.

Mentor Meeting - 10/24/17

We started out the meeting discussing what we did that week and what we were looking to work on in the coming week. Following that, Anni/Ejay showed off the progress they had made on the front end. Ejay then had a few front end questions/comments that he discussed with the mentors. To end the meeting, we discussed our sitter suggestion algorithm with the mentors that we had turned in for the previous project deliverable.

Group Meeting - 10/26/17

We met after class and discussed the overall design of our application again. We did this in order to make sure we were all on the same page. The group then continued to work on their individual tasks. Like last week, the goal is to have most of the front end ready next week for Milestone 3.

Mentor Meeting - 10/31/17

We started out the meeting discussing what we did that week and what we were looking to work on in the coming week. Following that, Annie/Ejay showed off the progress they had made on the front end and discussed some design decisions we had made as a group. Morgan then demoed the new backend functionality with the API. Lastly, we briefly discussed our thoughts for our sitter suggestion algorithm/preferences with the mentors.

Group Meeting - 10/31/17

We met during class and discussed the overall design of our application again. The group continued to work on their individual tasks. Annie and Ejay worked on the front end. Morgan continued to work on the API and Ryan/Pearson worked on updating documentation.

Group Meeting - 11/2/17

We met during class again. The group continued to work on their individual tasks. Annie and Ejay worked on the front end. Morgan continued to work on the API and Pearson worked on updating documentation.

Group Meeting - 11/3/17

We met outside class and worked to clean up all of our code. We then worked to integrate the API into the front end now that we have the API configured. Lastly, we finished the rest of P10, which included updating/new documentation.

Mentor Meeting - 11/7/17

We started out the meeting discussing what we did that week and what we were looking to work on in the coming week. Following that, Ejay showed off the progress made on the front end. Morgan then asked a question he had about authentication and that was about it. We ended the meeting a few minutes early.

Mentor Meeting - 11/14/17

This week Ejay was unable to attend due to another responsibility. We started out the meeting discussing what we did that week and what we were looking to work on in the coming week. Following that, Morgan showed off some progress made on the back end and we ended the meeting a few minutes early.