Pets Boarding Project Report

A mobile application for helping pet owners find pet resorts and pet sitters in a more efficient way

Team Member: Xiaojing Ji - Master in Computer Science

Motivation and Objective

When pet owners have travel plans, they usually don't bring pets with them. Instead, they are more likely to request a comfortable and affordable place from either individual pet sitters or pet resorts. Cities have many pet resorts and individuals to provide pet boarding services. However, choosing a good deal among all available options is not as easy as it sounds. Here's what will happen if a user is looking for pet sitting services: first use a search engine to get relevant search results and possibly a map displaying locations of nearby pet resorts. Then visit websites suggested by the search engine to find information about service prices and restrictions. Those restrictions include but not limited to the fact that some companies only run their businesses for certain types of animals like dogs or cats, so people who own other types of animals have a frustrating time to find useful services. Finally, make a decision after exhaustingly reading through all those websites. As you can see, due to the lack of information integration, the process of finding a satisfiable pets resort is exhausting and time-consuming, and it becomes the main reason of designing this application. At the current stage, no website or released mobile application is offering similar services. This mobile application aims at providing integrated data of urban pets resorts and individual pets sitters to simplify the process of finding suitable service providers. Users can refine searching results by selecting hourly ranges, service types and other features before comparing the returned options to make a better decision with fewer efforts.

Related Tools

- Meteor: A Javascript platform for web and mobile app development. I choose Meteor because of its advantages in the deployment and reactivity. Meteor application can be deployed on both iOS and Android. For the full stack reactivity. UI reflects real-time state with minimal developing effort. In addition, it supports "Data on Wire", which means the client renders data sent from the server without sending HTML.
- MongoDB: A NoSQL database system which maintains data in JSON-like documents. Meteor stores data in MongoDB as "Collection" by default.
- mLab: Provide cloud database service which hosts and supports MongoDB databases. In this application, it runs on Amazon Web Services cloud.

System Implementation

- **Data Source**

  Profiles of pet resorts are collected from third-party sources and profiles of pet sitters are created by registered users who consider themselves as caregivers. The third-party sources include the searching results returned by Google search engine, Faroo API, and crawled contents from pet resort websites. Faroo API can be an alternative for deprecated Google Web Search API[1]. Queries are able to be returned in multiple formats like XML or JSON. In order to collect more detailed data, contents on some pet resort websites will be crawled as well. Both the crawled data and the JSON formatted data returned by Faroo need to be preprocessed in order to successfully extract useful information from the raw resources. The preprocessed data is stored on the mLab cloud database, so once the user opens this application on a mobile device, it automatically connects to the cloud database to get stored information. Because the current dataset is fairly small and limited to Atlanta, the system also keeps a local version of those preprocessed pet resorts data for local testing.

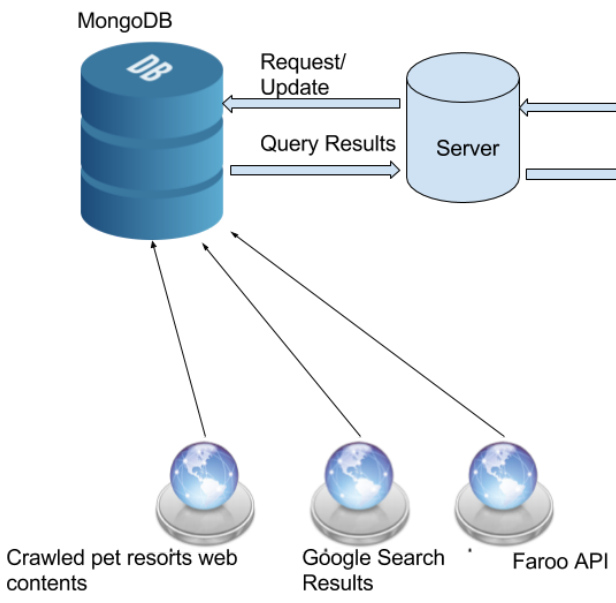
- **System Functionality**

  The functionalities and features of this application are exactly the same as what has been addressed on the project proposal. The goal of developing this mobile application is providing a mobile application for pets owners to find and compare nearby pet resorts and pet sitters. It helps users integrate data collected from local pet resort websites and registered care providers. Users can easily specify features like cost or animal types to filter the results. The refined results will be displayed in a nicely-formatted list.

  Based on the essential requirements listed above, the following features are completed:

1. **Search Filter**: Sorting and refining data is the key feature of this project. Filters include city, accepted animal types, ratings, and cost range. Users can specialize their requests or leave some filters empty accordingly.
2. **Display Results in Comparison Format**: Pop-up refined service providers to the user. Users can compare different providers in a more efficient and straightforward list. Additionally, on each entry, users can redirect themselves to previous reviews or leave a comment to the corresponding service provider.
3. **Rating**: Users do care about the rating of pet resorts and individual pet sitters. Good feedback will attract more customers and prove the reliability. The system provides a feature for entering ratings and comments. After a review is submitted, its rating will be accumulated to the rating system, and the overall score will be updated.
4. **New caregiver registration**: Considering the number of pet resorts is limited and their services might be expensive, this system provides pet sitters' information as an alternative solution. Any registered users can easily register themselves become pet sitters. After the caregiver registration completes, if the given information matches the queries, their profiles show their contact details, interests, and hourly/daily rates with other service providers. Besides, evaluations from previous customers will be displayed. This feature gives more options and more flexibility to pet owners.

- **System Architecture**



The system is built in Meteor platform which integrates nicely with MongoDB database. I have tested the application on Ripple Emulator, which allows the developer to check how the system looks like on different devices. Additionally, the application has successfully launched on both iPhone iOS environment and Android emulator. Whenever a user sends a request, the client evokes an event activity corresponding to that certain request. The template in Meteor groups all helper functions and event functions. More precisely, each single template is responsible for one major activity(like a page). Switching between pages usually implies switching on templates as well. For example, the login page is a login template and the register page is a register template in this project. Event functions inside login templates only take actions for the login behaviors without caring about activities from any other templates. The server stores a default dataset of pet resorts and has a light implementation because the client handles most actions in this project.
The appendix shows all the tasks. I have completed all the assigned works, but based on the suggestions, I add some future work for this project.

Deliverables

- The major component of the final deliverables is a deployable mobile application. The application can also be tested on localhost following instructions posted on official Meteor website.
- The deliverables also include this project report as the complementary because it covers a detailed description of the system architecture and explanations of the implementation.

- Video Demo: https://www.youtube.com/watch?v=C5crAa-YV7A&feature=youtu.be
- Source Code: https://github.com/XJi/PetsBoarding

Concerns

The most critical concern comes from the data source. Users expect to obtain useful information about pet boarding service providers, and they want the results as adequate as possible. This fact implies that the project is data-driven and it can only succeed with sufficient data resources. For example, if a user cannot find any service providers in New York simply because this system doesn't contain data for New York City, this user may quit using the application as it fails to meet the requirement.

Another concern is related to the rating system. The current rating system accepts any submitted comments without filtering out spam or fake reviews. It causes a potential issue because reviews may not reflect the actual quality of the services. Additionally, newly registered pet sitters have no rating initially, so they are likely to face the cold-start problem. Due to the fact people seriously consider previous reviews for decision making, new pet sitters will have a difficult time to start businesses.

Future Work

- Google Map API: Even though the current application returns the refined results with their address information, it is still not a user-friendly design because many users cannot imagine how far those places are by just reading their addresses. However, location does become a factor for the decision-making as people generally prefer to take a nearby service provider. If this system gets Map API embedded, it can display caregivers' locations on the map to help users get a clearer view.

- Third-party Login Feature: As it mentioned earlier, the application doesn't count the reliability of user reviews so people can abusively use this rating system. Because its rating system purely relies on its own data in this "closed" environment, the current implementation cannot detect malicious users. Detecting spam on the rating system has been recognized as a difficult task. However, if this system allows the third-party login feature, it can evaluate users' reliability and credibility from third party information and adjust itself accordingly. For example, if a user receives extremely low credits from third-party resources, then this application may place constraints for that user to leave comments.

Both of them are from the feedback of my project presentation. Due to the time limitation, I don’t have a chance to finish implementing both features, but they are thoughtful and really applicable to this project. Since I will continue working on this project afterward, I will add both features to the current design.

Reference

1. Faroo. "FAROO - Free Search API." *FAROO - Free Search API*. N.p., 01 Jan. 2010. Web. 17 Feb. 2017. <http://www.faroo.com/hp/api/api.html#jsonp>.

Appendix

- **Schedule and Work Plan**

Date	**Milestone**
February 12th - February 18th	Complete the project proposal
February 19th - February 25th	Collect data source from Google, Faroo API, and crawled contents from pet resort websites
February 26th - March 4th	Preprocess collected data, write scripts to extract keywords and useful information from collected websites
March 5th - March 11th	Import data to MongoDB database. On meteor platform, server side initializes the data and client side displays the data
March 12th - March 18th	Create page for caregiver registration. Add new user to the database
March 19th - March 25th	Create UI for user login.
March 26th - April 1st	Implement the filtering feature. Allow users to select options on the frontend
April 2nd - April 8th	Continue implementation on the filtering feature. Backend processes the requests and returns refined results to the client
April 9th - April 15th	Implement rating feature. Create UI for users to enter comments and scores.
April 16th - April 27th	Display comments and overall ratings for each care provider.