Alpaca Design

Overview

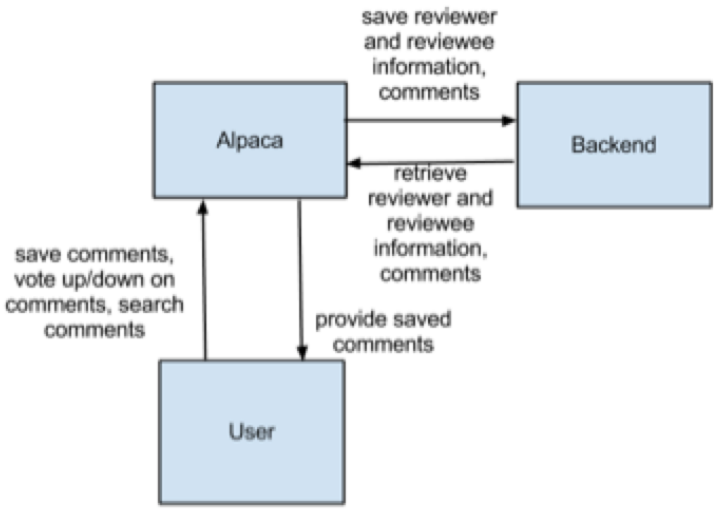
Goals & Purposes

Alpaca is a website for reviewing individuals as romantic partners. On this website people can share their personal experiences interacting with particular individuals and use such information to help decide whether an individual is a good boyfriend or girlfriend candidate.

Motivation

This project is motivated by the difficulty that people often have when trying to learn about newly acquainted friends as potential boyfriends or girlfriends . One approach to the problem is to acquire personal information through social network websites, a method commonly referred to as “stalking”. Information about the “stalkees” from these websites, however, is often hidden or biased towards the stalkees. By gathering reviews on the individuals in interest, Alpaca provides information about the specific individuals in a more objective and accessible way, making partner hunting an easier process.

Context Diagram



Type of users:

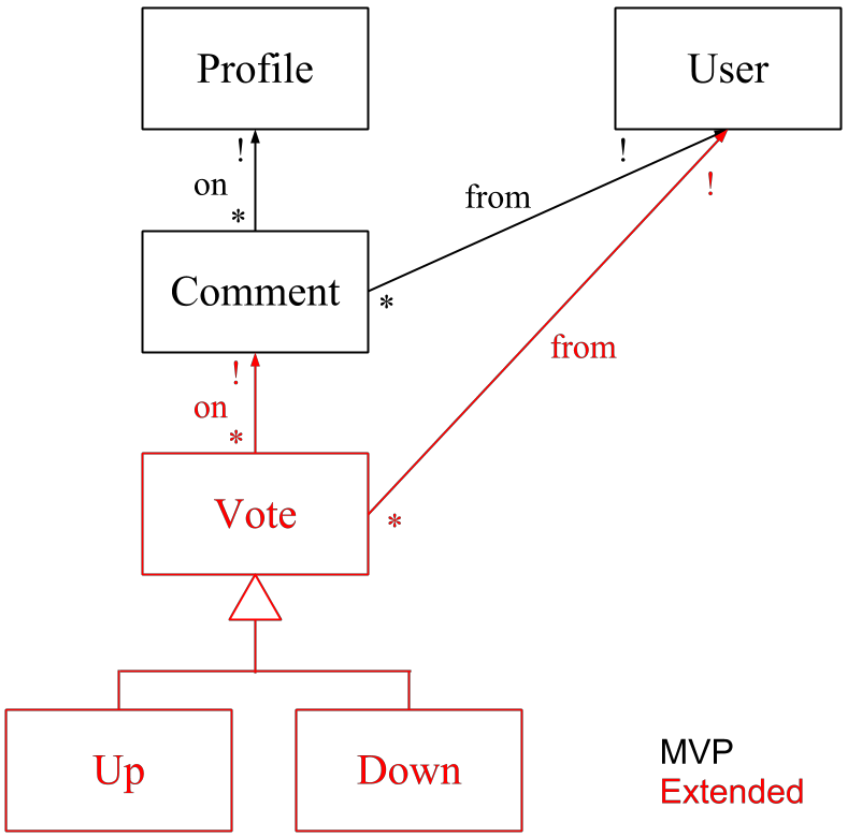
- reviewer: person who signed up and wrote comments on reviewees
- viewer: person who just view comments on the reviewees

Concepts

Key Concepts

- **User**: the person who registers in the site. He can post comments on his Facebook friends and view their Facebook friends' profiles.
- **Profile**: A page contains a reviewee's identity (name, Facebook profile picture) and comments on that reviewee
- **Comment**: comment on a profile
- **Vote**: vote up or vote down on a comment

Object Model



Behavior

Feature Descriptions

Alpaca provides the following list of features for a user:

1. **Log in**: Users (reviewers) can log in via their facebook account. Note that, when a user first logs in, he/she is asked (by Facebook) whether he/she allows *Alpaca* to access basic information such

as name, sex, list of friends and relationship status or not.

2. **Comment on a Friend**: Each user can comment on his/her Facebook friend.
3. **(Extension) Vote Comment Up/Down**: Each user can vote each comment made on his/her Facebook friend up or down.
4. **View Profile List**: Each user can view list of all his/her Facebook friends' profiles. Profile pictures along with their names are displayed.
5. **View Friend Profile**: Each user can see his/her Facebook friend along with the comments made on that person. Note that all the comments are shown to be anonymous except the comment of the viewer. Vote results are also shown in this page.
6. **(Extension) Search**: Users can search friends and comments.

Security Concerns

The system needs to guarantee that all comments are shown anonymously to a user except for those made by the user himself. We use Facebook authentication system API to protect users' private information.

We might also want to ask for password everytime the user is logging into our site or making a comment. The purpose of this is to prevent security concerns when user forgets to log out either his/her Facebook account or Alpaca account from a machine.

Standard Attacks & Mitigation

- Injection Attack: attackers may inject sql commands to steal private user information or produce spam reviews in bulk. Mitigation: sanitizing user comment input.

- Cross site scripting (XSS): since Alpaca uses facebook credentials, attackers can steal facebook login session cookies. Mitigation: accepting only certain html tags and with well-tested parser

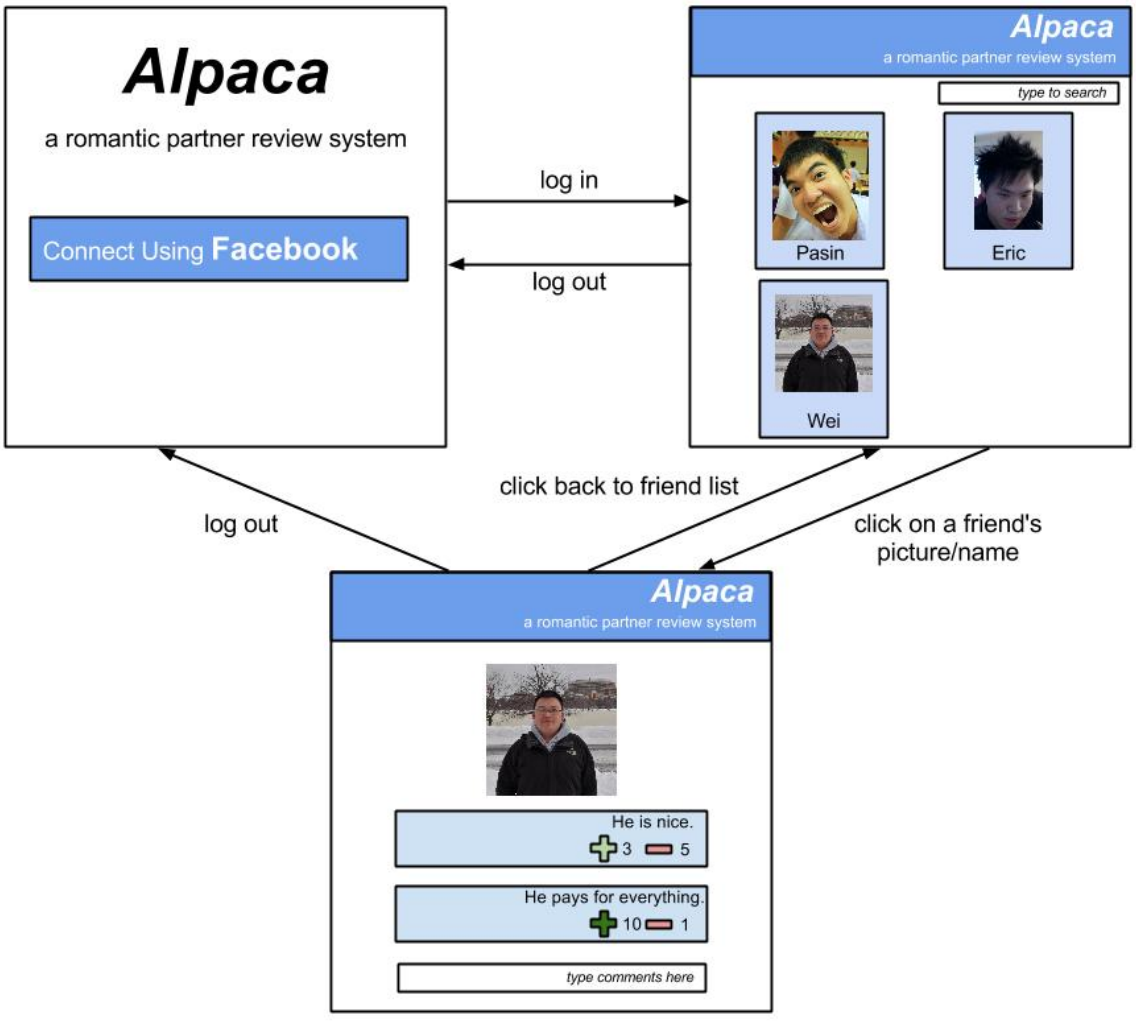
- Cross site request forgery (CSRF): for users who visited the attacker's website, attackers can use these users' credentials to post fake/spam reviews on Alpaca and Facebook. Mitigation: generate secret tokens for user sessions.

Privacy Concerns

We will have terms and conditions when users log in and we will also have disclaimer on the website that reviewers should be responsible for their comments. Users are responsible for their comments on any public forum in this country. We will friendly remind users their responsibility.

User Interface

The following diagram provides a basic sense of what the user interface looks like. User interface changes are done by AJAX except the login/logout which is done by normal redirection.



Challenges

Design challenges

1. How do we layout/arrange the reviewees profiles?

- Display the profiles in the form of table, and sort the profile alphabetically. The html page is extendable. Whenever the user logs in, the DOM creation will base on the size of the friends list.
- Overlap the profiles if the user generates more than one windows html page [defined size] of the profiles

Will apply the first approach since it is easy for user to view and organize the profiles. The second approach is hard to implement because the complexity of the UI structure (overlapping).

2. How do we display the comments of the selected reviewee?
   - Display/attach the comments right underneath the reviewees' profiles. The comments will be displayed in the order of creation
   - Redirect to a separate page if the user select a profile. On the separate page, the comments will be displayed in the order of creation

Will apply the second approach since the UI will be cleaner to the users. Rearrange the profiles since more comments added take more effort on the background. Separate the comments and from the profile page will give us more flexibility to apply Ajax.

3. How do we control the spam/fake reviews put off on the reviewees?
   - Complete open the saved profiles to all users; everyone can comment the profiles
   - Reviewer can view the profiles, which has the certain relationships with the reviewer and reviewee. The relationship must be within the friend network, but not limit to the direct friendship, could be friend of friend's friends.
   - Require reviewers to be Facebook direct friend with reviewee at the minimum

Will apply the third approach because it is more feasible to implement given the skills and techniques we learned so far. There is no evidence shown the first one would be less effective than others. However, as we also consider the privacy concerns, it is not a good practice to adapt the first one. The second approach is hard to implement. One reason could be multiple levels of friendships. Friendships are hard to keep tracking of and to follow if the reviewers have large networks.

Implementation challenges

1. Alpaca heavily depends on the Facebook API, how can we maintain the site/services availability if the Facebook API goes down?
   - Site goes down if the Facebook API goes down
   - Site maintains a limited services to the users if the Facebook API goes down; the limits services will exclude the followings: sign up using Facebook, attain the most recent copy of the friends list from Facebook, and request to update the reviewee's profile

Will apply the second strategy. The first strategy will ruin the site reputation badly. Even though without the Facebook API, we still be able to retrieve the profiles since they are saved locally within the application servers.

Development Challenges

1. Alpaca could be a controversial site. The success of the Alpaca will depend on the amount of the

user profiles we can get. How can we attract more people to join Alpaca?
(TBD)