Alpaca Design

Overview

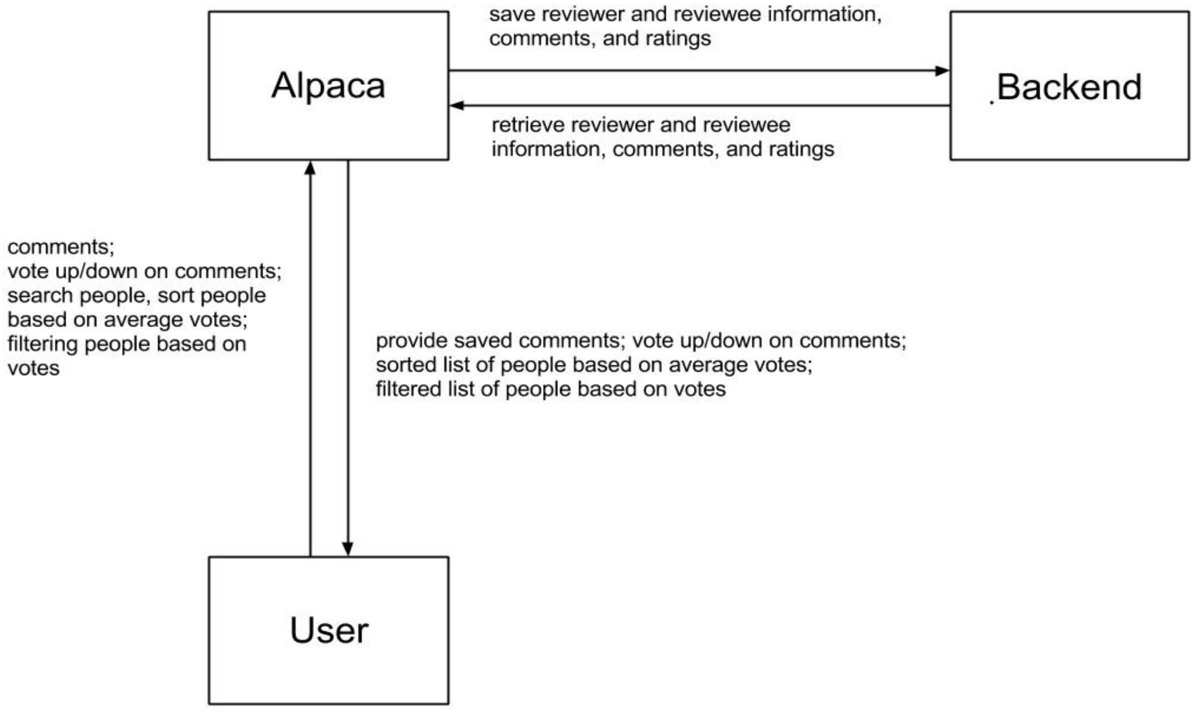
Goals & Purposes

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


Alpaca
Backend
User
save reviewer and reviewee information, comments, and ratings
retrieve reviewer and reviewee information, comments, and ratings
comments; vote up/down on comments; search people, sort people based on average votes; filtering people based on votes
provide saved comments; vote up/down on comments; sorted list of people based on average votes; filtered list of people based on votes

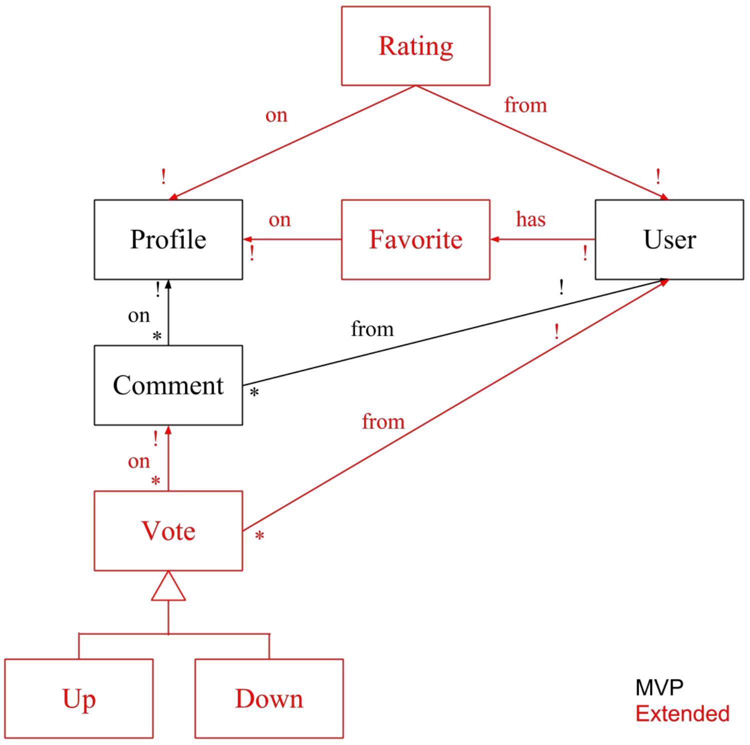
Note that here in our application, users are reviewers who give reviews and reviewees are profiles created by users. Reviewees do not need to be users.

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
- **Favorite**: each user can mark highly favorable profiles as favorites and view them in one place later conveniently.
- **Rating**: each user can rates each profile from half a star to five stars. There are four categories to rate on: appearances, IQ, EQ, bangability. These ratings can be used in sorting and filtering.
- **Locale**: The language users prefer to use. English, Thai, Traditional Chinese, and Simplified Chinese are picked as the locales to be translated to because our team members are able to translate our website to these languages.
- **Gender Preference**: Some people might have another sex partner preference

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. **Vote Comment Up/Down**: Each user can vote each comment made on his/her Facebook friend up or down.
4. **Internationalization**: The website can be viewed in multiple languages. For the purpose of the project, we choose English, Thai, Traditional Chinese, and Simplified Chinese.
5. **View Profile List**: Each user can view list of all his/her Facebook friends' profiles. Profile pictures along with their names are displayed.
6. **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.
7. **Search:** Users can search friends and comments.
8. **Images lazy loading**: the images only load if the user scrolls down the pages
9. **Gender preference**: the default gender preference will be the opposite sex of himself/herself. One can change it in the account setting, and only editable once.
10. **Rating**: users can rate each profile from half a star to five stars on the following categories: appearances, IQ, EQ and bangability.
11. **Sorting**: a user can sort his/her Facebook friends based on rating on each category.
12. **Filtering**: a user can filter has/her Facebook friends based on rating on each category.
13. **List of favorites**: a user can save profiles which he/she is particularly interested into a list

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

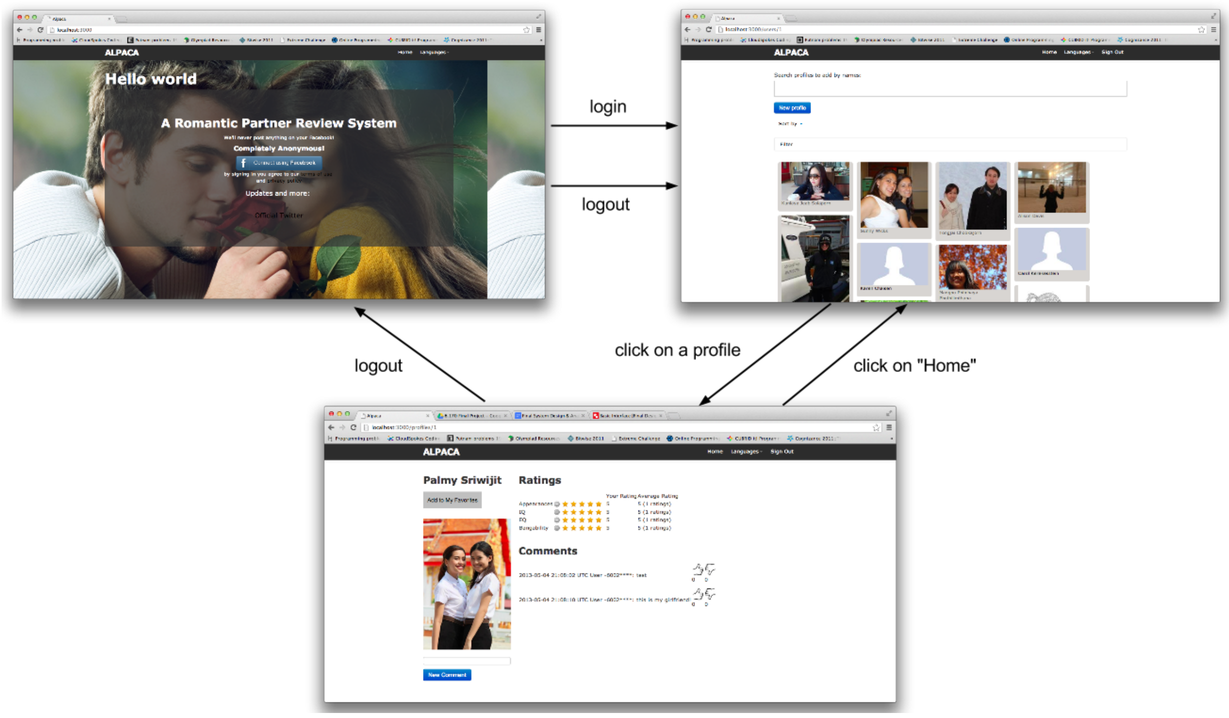
- Cross site request forgery (CSRF): for users who visited the attacker's website, attackers can use these users' credentials to post fake/spam reviews on Alpaca and Facebook. Mitigation: generate secret tokens for user sessions.

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.

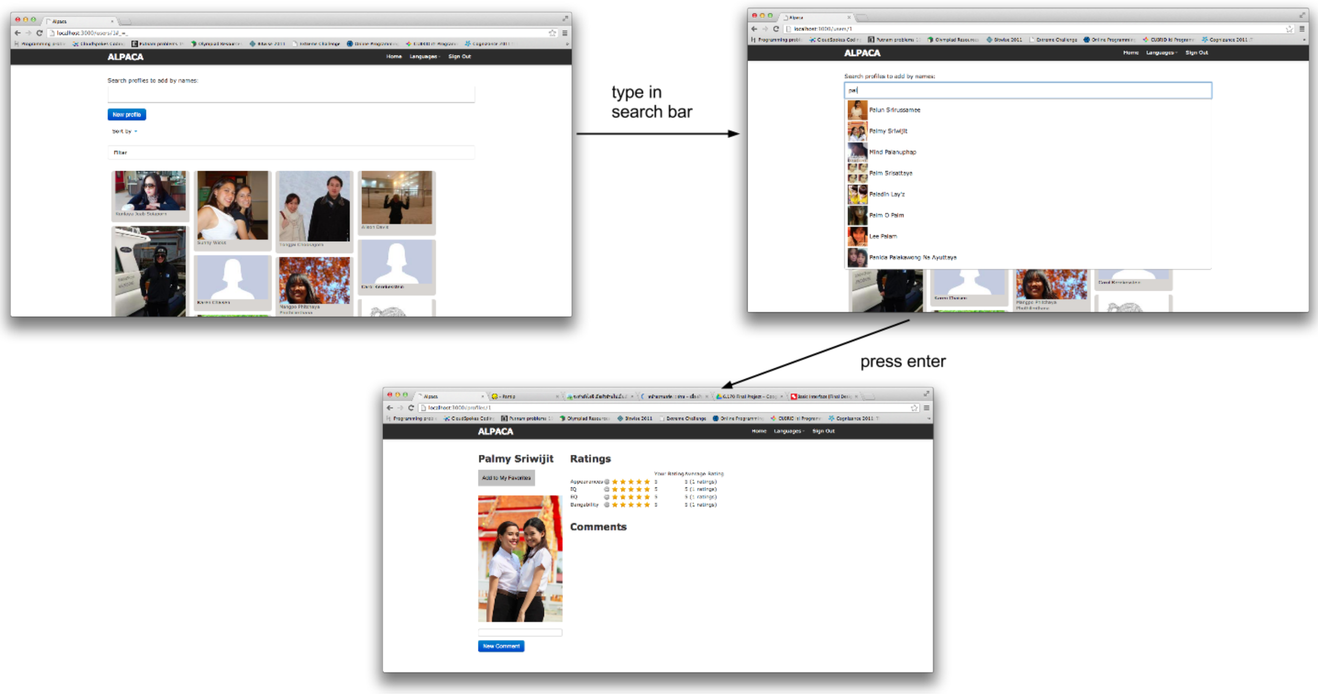
To protect privacy of reviewers, we display the hashed identities of the reviewers; moreover, we increase the identity confusion by showing only a few digits of each hashed ID. Naturally we also discourage people from finding specific reviewers' credential information, unlike what reviewing website usually do. All these measures prevent users from finding out the real identities of reviewers.

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. Note that if unauthorized users try to access, they are redirected to the homepage.

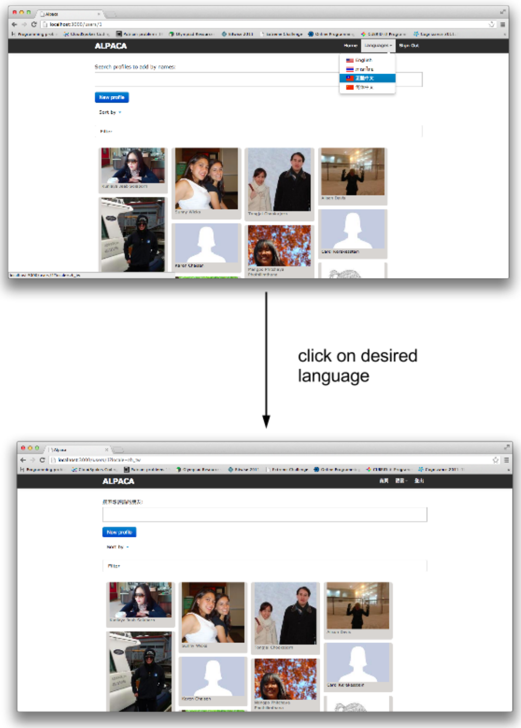


The detailed interface for each feature is displayed below.

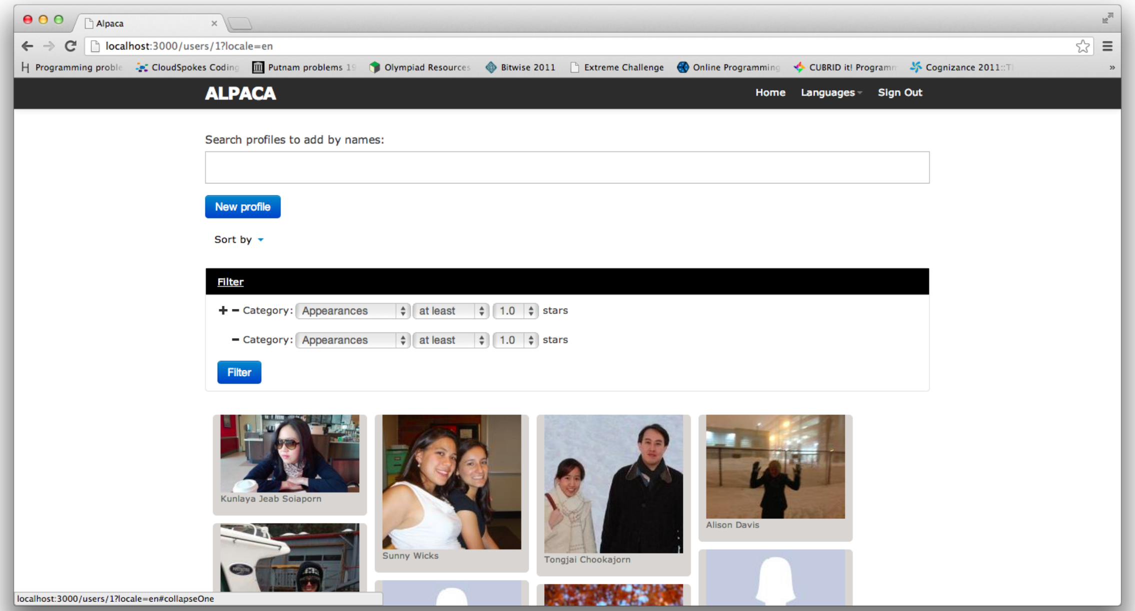
Searching



Internationalization



Filtering



Alpaca

localhost:3000/users/1?locale=en

ALPACA Home Languages Sign Out

Search profiles to add by names:

New profile

Sort by

Filter

Category: Appearances at least 1.0 stars

Category: Appearances at least 1.0 stars

Filter

Kunlaya Jeab Soiaporn

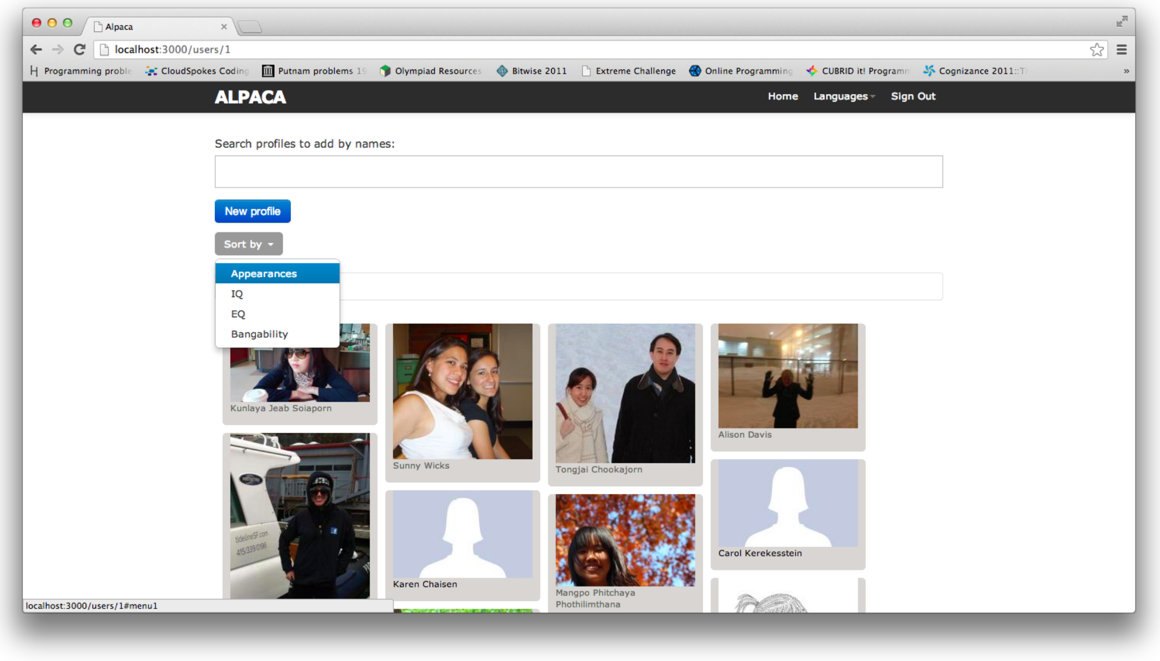
Sunny Wicks

Tongjai Chookajorn

Alison Davis

localhost:3000/users/1?locale=en#collapseOne

Sorting



Challenges

Design challenges

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).

How do we deal with the latency come with loading many images that result in bad user experience?

- Separate the profiles into pages
- User lazy loading the images as the user scrolls down the friend list page. This is the idea of dynamic loading for the user page

Will apply the second approach as the lazy loading is the standard approach for dealing with problem combinations of many images and heavy internet latency.

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.

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.

How do we deal with the internationalization language setting for this app?

- Save the setting as the temp variable within the user model
- Save the setting as the application variable at the system/application level
- Save the setting as the attribute within the user model, and save that into the database

Will adapt the last approach since it is more likely a standard approach. The language setting should be saved within the user mode, and also to be saved into the database as the user need to use that later login. The application itself should be independent of user's language setting, and should be only displayed the default language if there is no user sign in., or no such preference presents in the user.

Implementation challenges

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.

How to deal with the slowness of the machine since Heroku only provides some limited resource unless we pay for the premium service?

- Get a premium account from the Heroku
- Limited the number of the profiles show on the friend list page

Will adapt the second approach since the premium account is very expensive for us. We are planning to limit the number of profiles on each friend list page to be less than 500. We discussed this with the TAs, and he is fine with that.

Development Challenges
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?

We probably going to recommend friends to try out for the experimental stage. Once we feel the system is stable enough, we might going to do the public invitation only for public Beta testing.

Teamwork Plan

Stakeholders

Code Ninja: Wei, Eric, Pasin, Minshu
Project Secretary: Minshu
Project manager: Wei

Resources

Every team member are going to contribute at least 15 hours to this project per week on this project. We are expecting to spend at least 60 hours for this project. We are all agree to spend more time if we face the time constraint.

Tasks:

Welcome page:

- Facebook login JavaScript (Eric, Due 4/25)

User:
- Facebook authentication, create user if new; create session for the login user (Eric, Due 4/25)
- Fetch the most updated friends list from Facebook (Pasin, Due 4/25)
- Profile list page directs to the individual profile page (Pasin, Due 4/25)

Profile:
- Show the reviewee profile, includes the name, image, and etc (Pasin, Due 4/25)
- Backend JavaScript to the receive the comments from the server (Wei, Due 4/25)
- Favorites (Minshu, Due 5/7)
- Rating (Pasin, Due 5/7)
- Categorized rating of profiles. sorting / filtering (Pasin, Due 5/7)
- Gender Preference (Wei, Due 5/7)

Comment:
- Create the comment and set the profile id as the attribute (Minshu, Due 4/25)
- Attach the comment underneath the individual profile (Minshu, Due 4/25)
- Sanitize comments (Minshu, Due 4/25)
- Anonymization (Wei, Due 5/7)
- Vote up/down (Minshu, Due 5/7)

Lazy loading
- Reduce the request to the facebook server (Wei, Due 5/7)
- Delay loading of profiles unless the users scroll down the pages (Wei, Due 5/11)

Internationalization
- Code/Infrastructure implementation (Eric ,Due 5/7)
- Set up the translation local files with the translations (Eric, Due 5/7)

Risks:
Since the semester is approaching to the end, every team member has exams and final projects to deal with. Even though we split the tasks and set the target delivery date for the project, we are still facing uncertainties, such as sickness, or the unknown bugs that takes longer than expected. We are expecting to meet regularly, preferably daily google hangout and the physical meetup to report the individual progress. We are all agree that if one can’t fulfill his tasks due to sickness, others should pick up the remaining tasks depends on the date of the incomplete / undeliverable notification.

Minimum Viable Product

In minimum viable product, we are going to support the following features:
(basic GUI, no fancy css design)

1. Log in: people can log in with their facebook information
2. View profile list: Each user can view list of all his/her Facebook friends' profiles. Profile pictures along with their names are displayed.
3. View friend's 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.
4. Comment on a friend: Users can comment on a profile

Extension

For our extensions, we are going to implement the following features:

1. Vote up/down: each user can vote up/vote down each comment on his/her Facebook friend. As shown in the user interface diagram, there are plus and minus signs where the user can click to vote up or vote down the comment.
2. Better GUI
3. Vote Comment Up/Down: Each user can vote each comment made on his/her Facebook friend up or down.
4. Internationalization: The website can be viewed in multiple languages. For the purpose of the project, we choose English, Thai, Traditional Chinese, and Simplified Chinese.
5. Search: Users can search friends and comments.
6. Images lazy loading: the images only load if the user scrolls down the pages
7. Gender preference: the default gender preference will be the opposite sex of himself/herself. One can change it in the account setting, and only editable once.
8. Rating: users can rate each profile from half a star to five stars on the following categories: appearances, IQ, EQ and bangability.
9. Filtering: a user can filter has/her Facebook friends based on rating on each category.
10. List of favorites: a user can save profiles which he/she is particularly interested into a list