Group 2 (Taaag) Aspirations

Aspiration 1

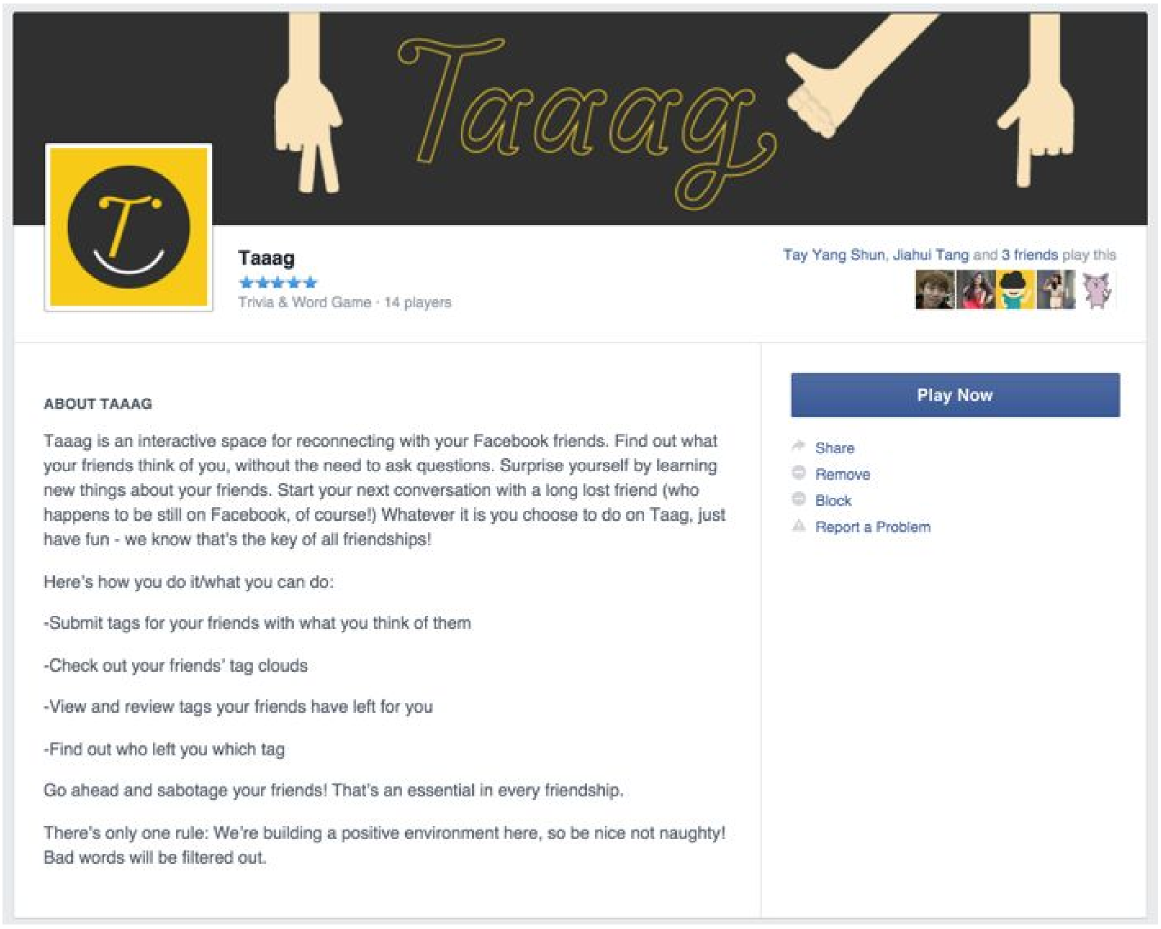
Choose to do a Facebook Canvas application, a standalone application, or both. Choose wisely and justify your choice with a short writeup.

The Facebook canvas was chosen because Taaag is based on an idea that requires users to interact extensively with their Facebook friends. The perks of the canvas app is that as it is embedded in Facebook, users can use Taaag as an extension of their Facebook experience, instead of clicking onto an external link. This is done all while we positively exploit Facebook's technologies like Graph API.

Aspiration 2

Taaag

Aspiration 3



Taaag
Taaag
★★★★★
Trivia & Word Game · 14 players
Tay Yang Shun, Jiahui Tang and 3 friends play this
Play Now
Share
Remove
Block
Report a Problem

ABOUT TAAAG

Taaag is an interactive space for reconnecting with your Facebook friends. Find out what your friends think of you, without the need to ask questions. Surprise yourself by learning new things about your friends. Start your next conversation with a long lost friend (who happens to be still on Facebook, of course!) Whatever it is you choose to do on Taag, just have fun - we know that's the key of all friendships!

Here's how you do it/what you can do:

-Submit tags for your friends with what you think of them

-Check out your friends' tag clouds

-View and review tags your friends have left for you

-Find out who left you which tag

Go ahead and sabotage your friends! That's an essential in every friendship.

There's only one rule: We're building a positive environment here, so be nice not naughty! Bad words will be filtered out.

Aspiration 4, 5

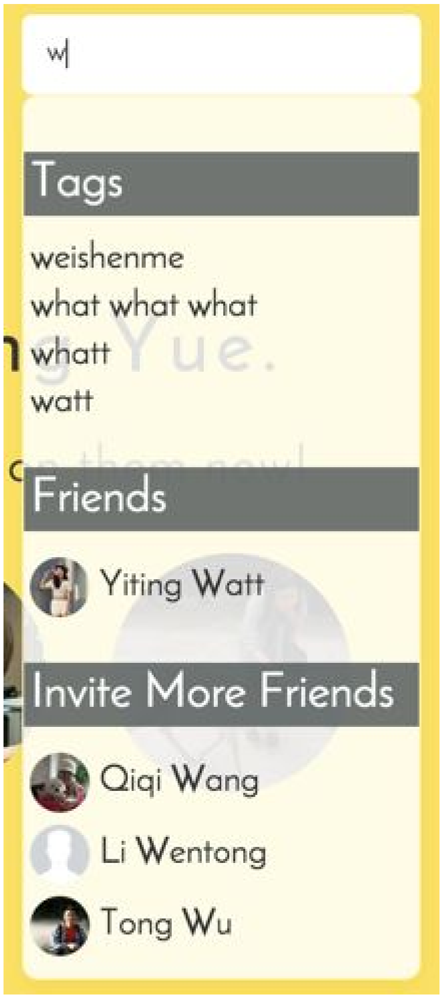
N/A

Aspiration 6

1. Query all tags of a certain user, together with the information of taggers.
   SQLAlchemy:
   ```
   tagger = aliased(User, name="tagger")
   return self.tags.join(tagger, tagger.id == Tagging.tagger_id). \
       order_by(Tagging.created).with_entities(Tag.name, tagger).all()
   ```
   Generated SQL (Don't read):
   ```
   SELECT tags.name AS tags_name, tagger.id AS tagger_id,
          tagger.created AS tagger_created, tagger.updated AS tagger_updated,
          tagger.name AS tagger_name, tagger.profile_url AS tagger_profile_url,
          tagger.access_token AS tagger_access_token,
          tagger.privacy AS tagger_privacy
   FROM taggings, tags JOIN users AS tagger ON tagger.id = taggings.tagger_id
   WHERE taggings.taggee_id = ? AND tags.id = taggings.tag_id
   ORDER BY taggings.created
   ```
2. Query all tag of a certain user, ordered by the time when each tag is created.
   SQLAlchemy:
   ```
   self.tags.with_entities(Tag.name, func.min(Tagging.created).label('first')).
   \
       group_by(Tag.name).order_by('first desc').all()
   ```
   Generated SQL (Don't read):
   ```
   SELECT tags.name AS tags_name, min(taggings.created) AS first
   FROM tags, taggings
   WHERE taggings.taggee_id = ? AND tags.id = taggings.tag_id
   GROUP BY tags.name
   ORDER BY first desc
   ```
3. Query all users with a certain tag, together with the number of taggers tag them as this tag, order by the number of taggers.
   SQLAlchemy:
   ```
   self.taggees.filter(User.id.in_(friends_list)). \
       with_entities(User, func.count(Tagging.id).label('votes')). \
       group_by(User.id).order_by('votes DESC').all()
   ```
   Generated SQL (Don't read):
   ```
   SELECT users.id AS users_id, users.created AS users_created,
          users.updated AS users_updated, users.name AS users_name,
          users.profile_url AS users_profile_url,
          users.access_token AS users_access_token, users.privacy AS
   users_privacy,
          count(taggings.id) AS votes
   FROM users, taggings
   WHERE ? = taggings.tag_id AND taggings.taggee_id = users.id AND
         users.id IN (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
   GROUP BY users.id
   ORDER BY votes DESC
   ```

Aspiration 7

- Login: Get basic information of a user, such as username, uid, profile url and access token.
- Friend list: For every user, we retrieved all his or her friends who have joined our app, to be displayed on friend list and tag each other.
- Invitable Friend List: This is used to suggest some friends that are not yet using the app when users type in the search bar. If the user clicks on them an 'invite' window will pop up with pre-filled recipient.



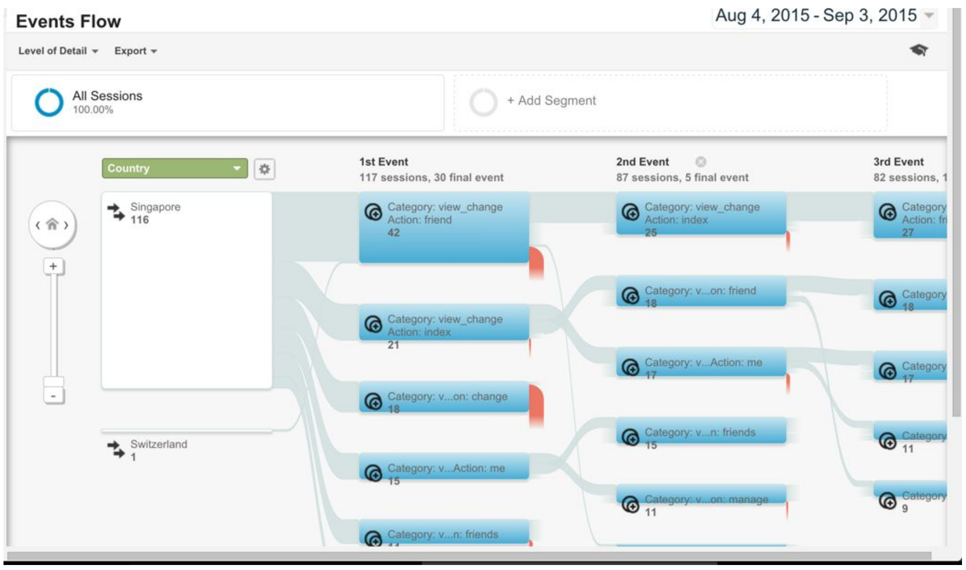
Aspiration 8

Users can like others' tag clouds or take a shot of their own clouds and publish these to their walls. Facebook JS SDK's dialogs are used to provide integrated look and feel, as Taaag is a canvas app. However, these two kinds of posts (the sharing of others' or of one's own cloud) are actually different, which will be explained in details later in Aspiration 14 Open Graph.

Aspiration 9
We decided to place the Like button on the bottom left of the canvas screen, on its own and away from distractions. Other positions worth considering (upper left, upper right, bottom right) were not chosen, because it would feel as if we're trying to push too many features together. Another option was to add the Like button to the menu bar, but it would be hiding the Like button away from the users, which is contrary to what we want - a Like button obvious and attention-seeking, to draw users to go for that call to action.

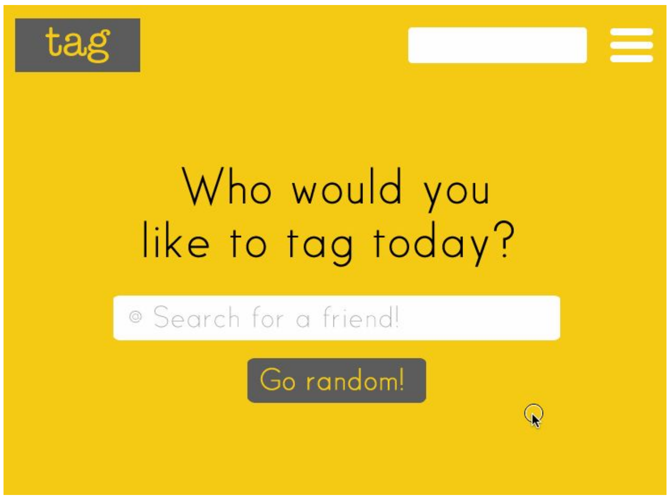
Aspiration 10
When users remove our application, we will remove all of their relevant data from our database, including personal information we have gotten from Facebook, the tags added to them, and the tags they have added to others. Besides, we will clear the cached friend list of their friends, so that when their friends log in, they will not be able to find the user who has already quit. This will be done by making use of Facebook's deauthorize callback.
This action is necessary because as developers, we must respect our users' privacy and choices. When they remove this app, their data will be deleted from our database permanently. If they choose to return, the data will be 'reinserted' into our database, rather than 'restored'. This is because we will lose our credibility if we retain their information without permission.
This action is logical, since when a user removes our app from their app list, they are indicating that they do not have interest in this app anymore. This is a deauthorization process, where they express that we do not have the permission to keep or access their data in the future, and we must respect their decision.
We have not violated Facebook's terms and conditions, because Facebook states that when IP content is deleted, it is done so in a manner similar to emptying the recycle bin on a computer - exactly what we have in mind, in deleting the user's data permanently when the user removes our application.

Aspiration 11
Since all views of the app share the same url and all interaction is done through ajax, we used custom events to trace users' actions.



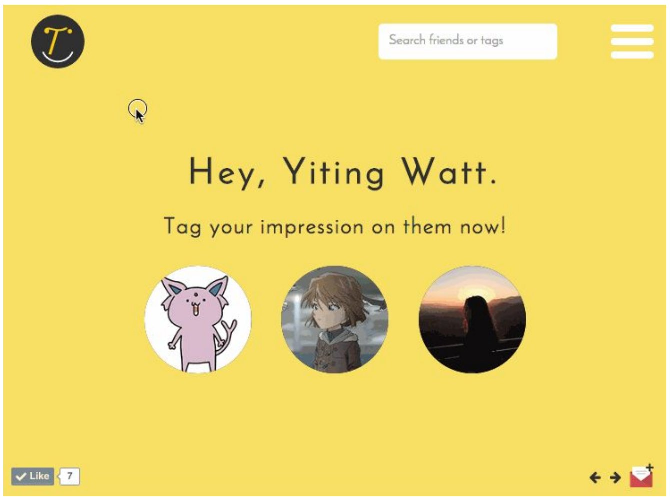
Aspiration 12
For the 3 user interactions, we'll be talking firstly about the suggestions of friends on the homepage, the ability to capture and share a screenshot of one's own tag cloud, and the ability to manage one's own tags.

<u>Suggestions of friends on the homepage</u>



[See A12-1-old.gif]

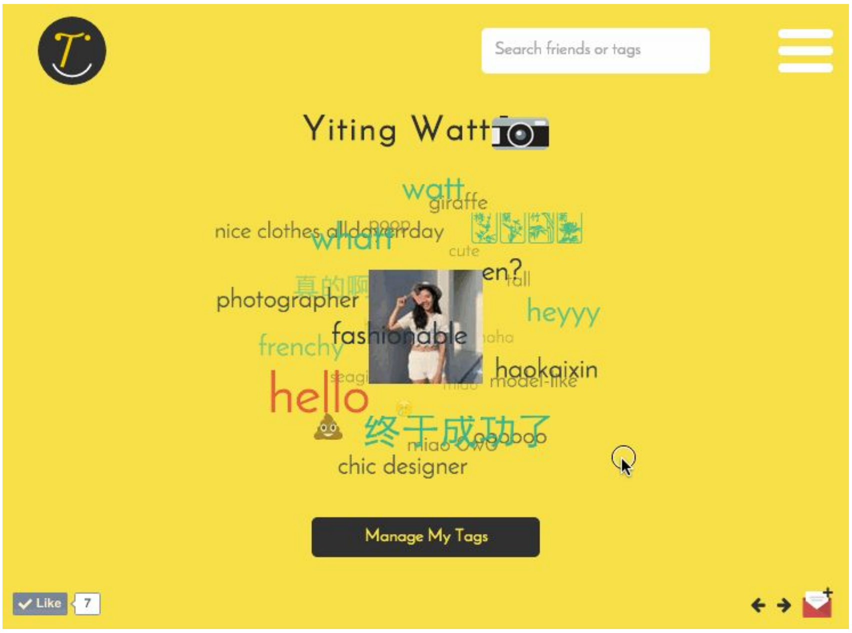
Our first ideas for the homepage was actually influenced by the Google Chrome extension - Momentum. We wanted something simple and minimalist, with direct and obvious calls to action, since there would only be 2 options - to either search for a friend, or to go random. We chose to remove this alternative, because while we liked how we were able to randomly generate a friend for the user, it might be restrictive for the user to only have 1 friend. Furthermore, the search bar wastes the user's time as they would have to stop and think of which friends they would like to tag. It is particularly restrictive in that the friends that will show up suggested on the search bar would have to already be on the app as well.



[See A12-1-new.gif]

With that in mind, this is how we have changed up the homepage: when you first login to Taaag, you'll be greeted by 3 random friends for you to choose to tag, as seen in the gif. Our rationale for this is simple - by suggesting friends for the user, they do not have to spend time thinking about friends that they would have to tag. We went with just 3 options at every one time, because we do not want to overwhelm the user with too many choices, and too little would be odd as well. By randomly suggesting friends, we hope to push them to tag friends they have never thought of tagging. We had wanted to keep the randomize function initially, but eventually discarded that idea, as it would have diverted the attention away from our 3 main suggestions and it is likely to be a feature that would not be largely appreciated by the users, given its restrictiveness.

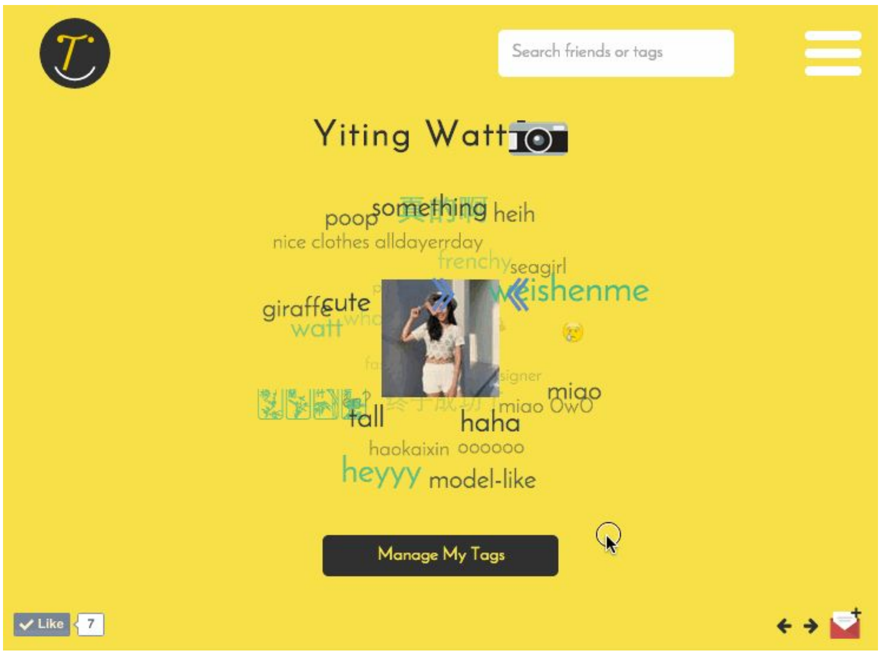
<u>Ability to capture and share screenshot of one's own cloud</u>



[See A12-2.gif]

Social networking often promotes the ability to post something interesting about the user onto their walls. This is typical on Facebook, where users like to share their achievements on apps onto their Facebook walls. Hence, we exploited this and came up with the feature of capturing a screenshot of one's own cloud, for users to share their wall of tags. Friends may then comment and even like the post. Something else really cool is that the user can adjust their cloud of tags to their own liking, then grab a screenshot of that, rather than have a random screenshot of the tag cloud. Also, by incorporating the sharing function into this camera function, it no longer feels like an annoying task that the users have to go through.

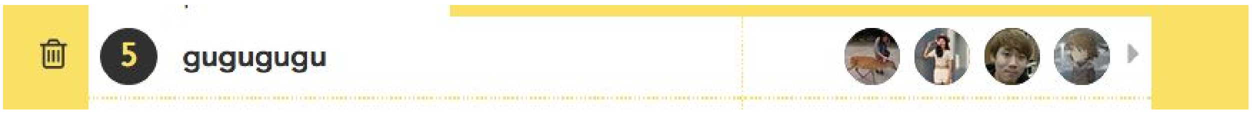
Ability to manage



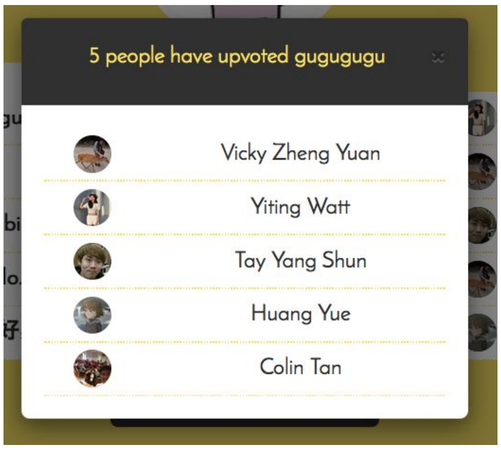
[See A12-3.gif]

Under the manage settings, we introduced the option to review tags - in the form of deletion of tags and to check which friends have added which tags for the user. The user will also be able to see which friends have upvoted which particular tag. Initially, we came up with the same idea, but we had wanted to make the delete option appear when one hovers over the tags in their own tag clouds. We realized later that it was not aesthetically appealing, and later decided to make managing easier by introducing a table form. Now, users are able to review and manage their tags at one glance.

Aspiration 13



When you open our app, click to your profile 'Me' in the menu list, then click "Manage my tags" on the bottom, and you will reach the view to manage your tags. You can click on the trashbin to delete them, and clicking on the triangle on the right reveals all taggers and upvoters of the particular tag.



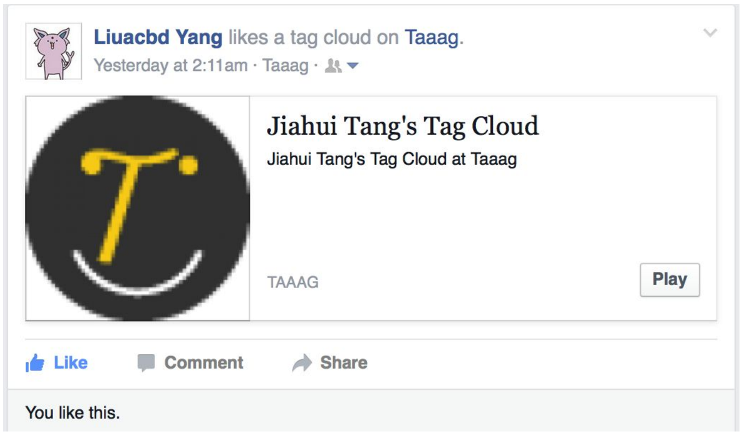
5 people have upvoted gugugugu

Vicky Zheng Yuan
Yiting Watt
Tay Yang Shun
Huang Yue
Colin Tan

Aspiration 14

We made use of a self-defined Object - the Tag Cloud.

1. Users can share tag clouds of their friends on their own timelines. This would be a link to the public version of the tag clouds, as seen in the image below. However, these friends may also choose to allow only their own friends to be able to view their tag cloud through the link. In this way, the cloud can still be shared by the user (eg: Liuacbd Yang in this example), but other people (Liuacbd Yang's friends) who are not friends with the user's friend (eg: Jiahui Tang) would be asked to login to the app to view the tag cloud upon clicking on the link.

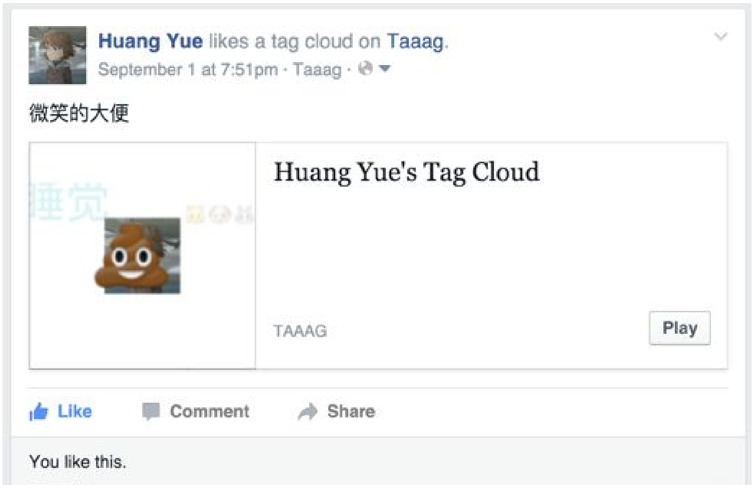


Liuacbd Yang likes a tag cloud on Taaag.
Yesterday at 2:11am · Taaag ·

Jiahui Tang's Tag Cloud
Jiahui Tang's Tag Cloud at Taaag
TAAAG

You like this.

2. Users can drag their own tags to any position they want and share the screenshot of their tag cloud by clicking on the camera icon beside their names. This actually generates a new 'Tag Cloud' Open Graph instance on the spot (so that the screenshot will appear in the feed). The link still leads to the same public version of cloud, just like when it is liked and shared by their friends. Basically, when a user or their friends share the user's cloud, the link on the feed would be to the same public cloud.



Huang Yue likes a tag cloud on Taaag.
September 1 at 7:51pm · Taaag ·

微笑的大便

Huang Yue's Tag Cloud
TAAAG

You like this.

Aspiration 15 (Optional)
We have used animations a couple of times throughout the app.

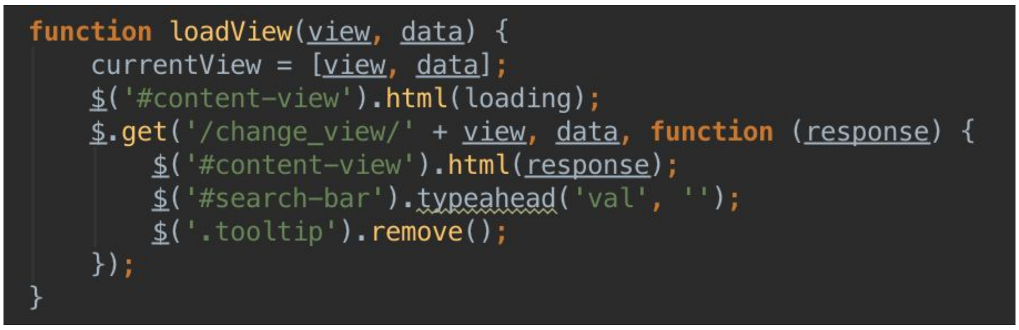
The first is the animating of the menu. When hovering on the menu button on the top right corner, a series of buttons will expand downwards. When the user does not need to access the navigation bar to access other pages, the menu will remain hidden to the user. This design keeps our page layout neater, and is less distracting.

We have also used CSS to animate some of our features - like the friends list, where upon hovering on the friends' profile picture, their names would appear. On the login page, we used CSS for the clouds to create a more realistic floating of clouds, and we have also used CSS for the login button itself, to make it pop out more.

Another area where we have animations is the insertion of gifs into the app, to increase more entertainment value for the user. The first is the login page for the first time user, where a gif was used for our mascot, to liven up the page and also to enhance and draw attention to the login button. Another area where a gif was used is in the settings page. As we do not have too many settings requirements, we used a gif of our mascot as a builder to make the page more fun and less plain and boring.

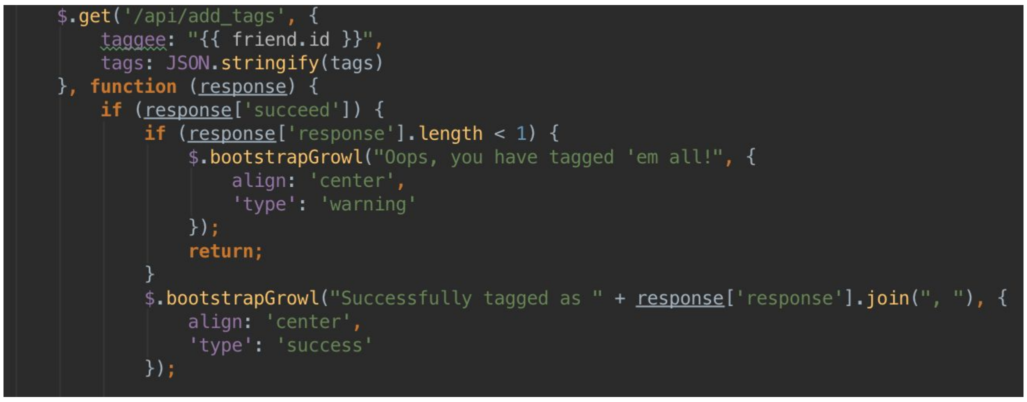
Aspiration 16 (Optional)
We used AJAX through the entire app.

1. To improve performance, we avoided redirection and used AJAX get method to obtain data when the views need to be changed. Almost all the view changing are done by AJAX.



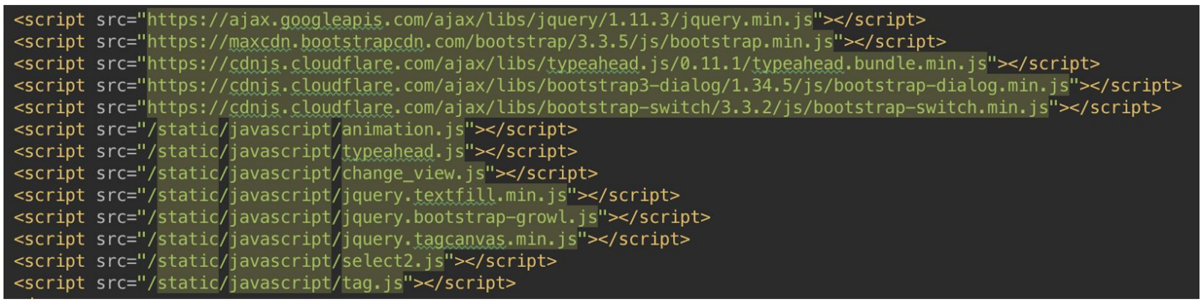
```
function loadView(view, data) {
    currentView = [view, data];
    $('#content-view').html(loading);
    $.get('/change_view/' + view, data, function (response) {
        $('#content-view').html(response);
        $('#search-bar').typeahead('val', '');
        $('.tooltip').remove();
    });
}
```

2. When use add/delete tags, we used AJAX to get quick and instant response.



```
$.get('/api/add_tags', {
    taggee: "{{ friend.id }}",
    tags: JSON.stringify(tags)
}, function (response) {
    if (response['succeed']) {
        if (response['response'].length < 1) {
            $.bootstrapGrowl("Oops, you have tagged 'em all!", {
                align: 'center',
                'type': 'warning'
            });
            return;
        }
        $.bootstrapGrowl("Successfully tagged as " + response['response'].join(", "), {
            align: 'center',
            'type': 'success'
        });
```

Aspiration 17 (Optional)
We’ve used quite a few jQuery plugins to enhance user experience.
For example, we used ‘tagcanvas' to display the tag cloud of a certain user, and added some effects to entertain them. We also used plugins like ‘typeahead’ or ‘select2’ to simplify user flows.



```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/typeahead.js/0.11.1/typeahead.bundle.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap3-dialog/1.34.5/js/bootstrap-dialog.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-switch/3.3.2/js/bootstrap-switch.min.js"></script>
<script src="/static/javascript/animation.js"></script>
<script src="/static/javascript/typeahead.js"></script>
<script src="/static/javascript/change_view.js"></script>
<script src="/static/javascript/jquery.textfill.min.js"></script>
<script src="/static/javascript/jquery.bootstrap-growl.js"></script>
<script src="/static/javascript/jquery.tagcanvas.min.js"></script>
<script src="/static/javascript/select2.js"></script>
<script src="/static/javascript/tag.js"></script>
```