NUSHello

NUSHello is anonymous, real-time chatting made exclusively for the NUS community to discover and make new friends.

Why NUSHello?

Many NUS students find it hard to meet new friends, either normal friends or a romantic partner. There are not many chances to meet people from other faculties except joining a lot of CCAs. Most of them are hesitant about trying friend-making apps or dating apps because there's too much noise in the list of suggested people. Besides that, they are also worried about being judged by their friends because of using a dating app. The social circle of most students is restricted to fellow coursemates who study within the same faculty.

Technical Stack

Frontend

ReactJS + Reflux
Bootstrap
Webpack
Gulp

Backend

Ruby on Rails
PostgreSQL

Hosting

Amazon Web Services (AWS) Elastic Cloud 2 (EC2)

Aspiration 1 - Application Type

We chose to build a standalone app instead of a canvas app because we want to have greater control over the UI/UX. As canvas app has many limitations, such as fixed width viewport, can only access through a promotional page or a HTML tab from my Facebook app page. Making a canvas app to work properly on mobile needs more work.

Besides that, since this app is only exclusive for NUS students, but not for all Facebook users, we feel that a standalone app is a better choice.

Bonus Aspirations

What are the pros and cons of each method of visibility control? When should one use the JavaScript method and when should one use the PHP method?

The difference is whether the rendering is done on the server side or on the client side. Basically, JS SDK (client-side login) is faster as the request goes directly from the client to Facebook, and from Facebook directly back to client, instead of client to server to Facebook back to server back to client. Thus the response time would be much shorter, and at the same time reduce the load on our server. This arises from the advantage of client-side rendering. It is also more universal. Almost every server supports JavaScript but not every server uses PHP. Client side rendering also enables us to do AJAX calls to do conditional rendering for the login buttons.

However, the tokens that the JS SDK obtain are only short-term tokens which will expire in about one to two hours. While using the PHP SDK (server-side login), the server is able to obtain a long-lived token from Facebook which can last for 60 days.

When we want to ensure that the visibility of the buttons is done without refresh, JS is better.

What is the primary key of the home faculties table?

The combination: (`matric_no, faculty`)

Aspiration 6 - SQL Queries

User Authentication

SQL

```
SELECT "users".* FROM "users"
WHERE "users"."facebook_id" = $1
AND "users"."access_token" = $2;
```

ORM

```
User.where(facebook_id: facebook_id, access_token: access_token).first
```

To find the right user in the current session, we look for the User whose `facebook_id` matches and `access_token` is the one provided.

Retrieving Conversations

SQL

```
SELECT  "conversations".* FROM "conversations"
WHERE ((user_1_id = $1 AND user_2_id = $2)
OR (user_1_id = $2 AND user_2_id = $1))
ORDER BY "conversations"."id" ASC LIMIT 1;
```

ORM

```
Conversation.where('(user_1_id = ? AND user_2_id = ?) OR (user_1_id = ? AND
user_2_id = ?)', user_1.id, user_2.id, user_2.id, user_1.id).first
```

To find the right conversation, we look for the Conversation whose `user_1_id` and `user_2_id` matches correctly with the 2 given ids. (First id is equal to the first or second user and second id is equal to the second or first user, respectively.)

Adding Messages

SQL

```
UPDATE "messages" SET
 “content" = $1,
 “conversation_id” = $2,
 “user_id” = $3,
 "updated_at" = $4;
```

To insert the conversation into the database, we insert into the Messages table the content, along with the id of the conversation that the message belongs to and the user who sent the message.

Aspiration 7 - Facebook Graph Queries

- Getting user profile (`/me`): Authentication of user, verification of identity
- Getting user's friend list (`/{friend-list-id}`): Filter out Facebook friends to allow users to make new friends

Aspiration 8 - Facebook Feed

The application will ask to post to Facebook Feed after a user has found some matches. This allows him to share his initial step in reaching out to the wider NUS community to make new friends.

Aspiration 9 - Positioning of Like Button

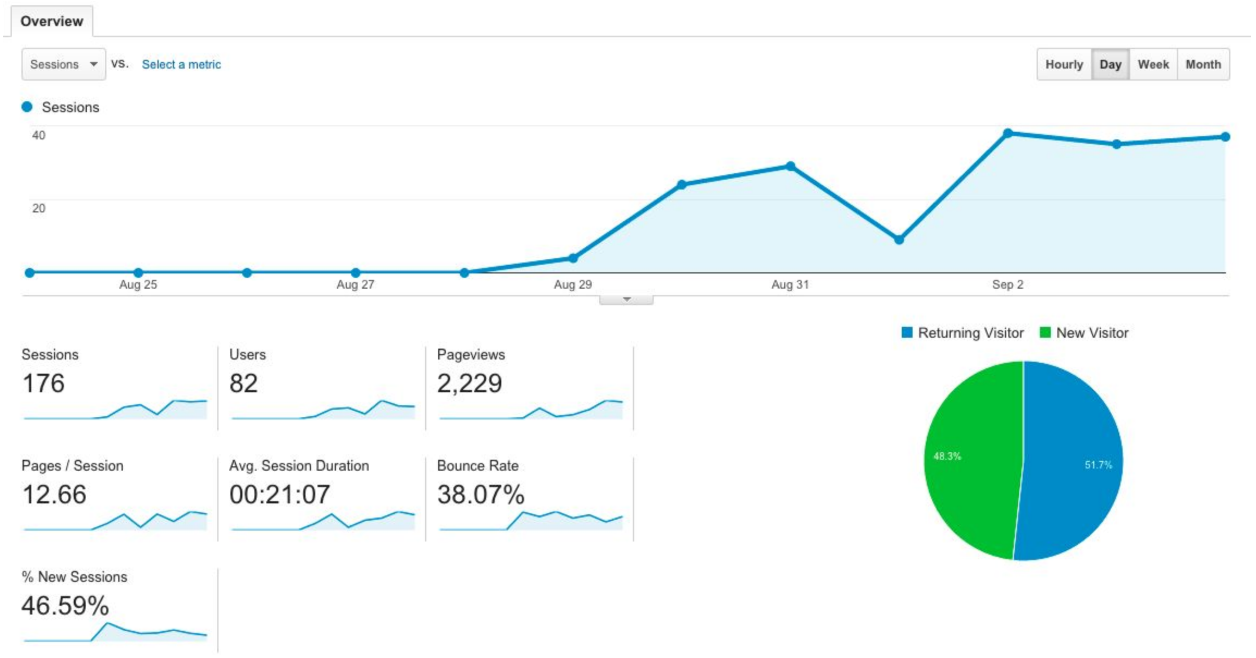
Many other websites today place a 'Like' button at the footer, where there are links to other social media sites. Users would consequently expect to find a similar button at our site. Thus, the button does not feel out of place, yet it would receive users' attention. Users who are interested in our social media page will then be provided with the option to 'like' the page directly from the footer, instead of having to navigate directly to our application's Facebook page.

Aspiration 10 - Handling of a User's data

Leave the data in the database, however, have a Boolean marker be modified to mark the User as 'deleted'. By deleting the data, other users who have been previously chatting with this particular user will find their conversations magically disappearing. This would not be a good experience for them. Instead, if users navigate to a conversation whereby a user has left the app, we can disable the input and put a small line notifying the user of the situation.

Additionally, according to Facebook's policy for developers, we are required to "delete all of a person's data [we] have received from [Facebook] (including friend data) if that person asks you to". We are otherwise not required to delete any of the user's data, and thus would not be violating Facebook's policy.

Aspiration 11 - Google Analytics Screenshot



Sessions
176

Users
82

Pageviews
2,229

Pages / Session
12.66

Avg. Session Duration
00:21:07

Bounce Rate
38.07%

% New Sessions
46.59%

Aspiration 12 - User Interactions

Implementing a footer and sidebar layout

On static, non-conversation-related pages, we have implemented a footer, whereas on the conversation-related pages, we have decided to implement a sidebar for navigation. Initially, we had considered various implementations, such as using a navbar or having no fixed navbar. However, we decided that given the style of our landing page, we wanted to attract the user's attention to the main banner and call to action button, thus implementing a subtler footer. On pages which are related to chatting, having a sidebar is intuitive and good for listing out the conversations in a clear manner to the user. The icons and buttons at the top of the sidebar are the only navigation options available for the user to reduce confusion as compared to if there were a footer as well.

The chat page as the home page

After logging in, old users are redirected to the conversation view so that they are able to see all of their current conversations. Since chatting is the main purpose of our application, returning to our application, users would have two main motives: either that of talking to someone they had been matched with, or finding new matches. We have thought about putting the page of matches for browsing as the home page but it felt unintuitive. For instance, apps like Instagram are more focused on browsing whereas ours was clearly a communication app more focused on chatting. Although some users may go on to browse for new matches, we decided that the chatting view is still the most natural.

Focus on chat input

The chat input is immediately focused for the user upon reaching the chat page. Thus, the user does not need to move the mouse or take one extra click before starting to chat. This helps to reduce user fatigue. Moreover, there is a subtle animation on opacity of the chat input bar if the user clicks away. We believe that having such a transition better aids the user in case the user might have clicked away and found that the input was not focused. The difference in colour enables to user to tell if they can just start typing away.

Aspiration 13 - DOM Manipulation

We use conditional rendering from ReactJS to render the avatar of the users differently. On the matches page, the avatars are centered. This is done using an if-else expression, and determined by using the properties passed in from the parent element in React. On the matches page, the avatars are centered by adding a class. The class is added by determining if the value of `shouldCenter` is true. However, on the sidebar, they are aligned to the left.

Aspiration 14 - Facebook Story

We decided to create a Facebook story at the point when two users begin chatting with each other. The Action is **found** and the object will be **match**. This is for the user to share with his friends the new user he is talking to and hopefully trigger the curiosity of other users to use the application.

Aspiration 15 - Animations Used in our Application

We made use of a Bootstrap theme, which had CSS classes preloaded that included transitions and animations. Therefore most of the animations seen on the page is from the Bootstrap theme.

Aspiration 16 - AJAX Used in our Application

Our application is a Single Page Application which uses AJAX to send GET and POST requests. Since the server does not have to send the entire HTML back to our client, our web app behaves more like a native app that is responsive and fast, thus improving the user experience.