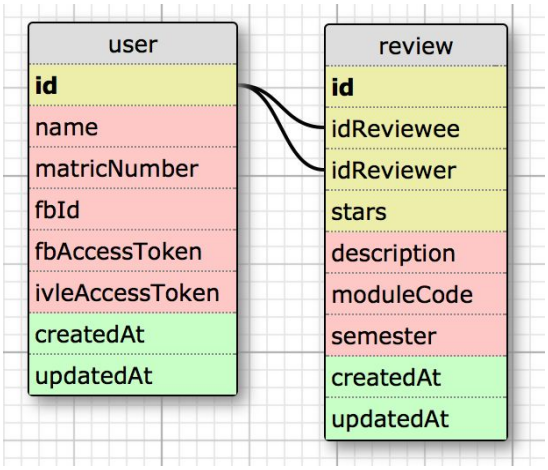
Aspiration 1: Choose to do a Facebook Canvas application, a standalone application, or both. Choose wisely and justify your choice with a short write-up.

In NUS, it is a fact that a lot of students do not/cannot choose groupmates with enough information since there are no data points. They are usually assigned with random people. This leads to numerous problems such as: different behaviour among groupmates, limited skillsets in the teams, etc. So we came up with an idea to make a platform where students can add reviews on their group mates. Future students can know more about their classmates through past reviews reviews, know who probably fits them the best and form a strong group.

Because the idea is not related too much Facebook, we wanted to make it a standalone application. It would also make it easier for a user to use the app (just enter a URL).

Aspiration 5: Draw the database schema of your application.



Aspiration 6: Share with us some queries (at least 3) in your application that require database access. Provide the actual SQL queries you use and explain how it works.

Query 1:

```
SELECT `users`.*, `modules`.`id` AS `modules.id`, `modules`.`uuid` AS `modules.uuid`,
`modules`.`userId` AS `modules.userId`, `modules`.`code` AS `modules.code`,
`modules`.`title` AS `modules.title`, `modules`.`createdAt` AS `modules.createdAt`,
`modules`.`updatedAt` AS `modules.updatedAt` FROM (SELECT `users`.`id`,
`users`.`createdAt`, `users`.`updatedAt`, `users`.`name`, `users`.`matricNumber`,
`users`.`fbId`, `users`.`fbAccessToken`, `users`.`ivleAccessToken`, `users`.`isRegistered`,
`users`.`haveFetchedModules` FROM `users` AS `users` WHERE `users`.`id` = 8 LIMIT 1)
```

```
AS `users` LEFT OUTER JOIN `modules` AS `modules` ON `users`.`id` =
`modules`.`userId`;
```

This query is executed when loading the profile page. It fetches a user's details, along with the modules that he/she is taking. It does a left outer join of the users table with the modules table with the user id as the foreign key. (Left Outer join since we want the user's profile even if the person does not have any modules)

Query: 2

```
SELECT `reviews`.`id`, `reviews`.`idReviewee`, `reviews`.`idReviewer`,
`reviews`.`createdAt`, `reviews`.`updatedAt`, `reviews`.`stars`, `reviews`.`description`,
`reviews`.`moduleCode`, `reviews`.`semester`, `reviewer`.`id` AS `reviewer.id`,
`reviewer`.`createdAt` AS `reviewer.createdAt`, `reviewer`.`updatedAt` AS
`reviewer.updatedAt`, `reviewer`.`name` AS `reviewer.name`, `reviewer`.`matricNumber`
AS `reviewer.matricNumber`, `reviewer`.`fbId` AS `reviewer.fbId`,
`reviewer`.`fbAccessToken` AS `reviewer.fbAccessToken`, `reviewer`.`ivleAccessToken` AS
`reviewer.ivleAccessToken`, `reviewer`.`isRegistered` AS `reviewer.isRegistered`,
`reviewer`.`haveFetchedModules` AS `reviewer.haveFetchedModules`, `reviewee`.`id` AS
`reviewee.id`, `reviewee`.`createdAt` AS `reviewee.createdAt`, `reviewee`.`updatedAt` AS
`reviewee.updatedAt`, `reviewee`.`name` AS `reviewee.name`, `reviewee`.`matricNumber`
AS `reviewee.matricNumber`, `reviewee`.`fbId` AS `reviewee.fbId`,
`reviewee`.`fbAccessToken` AS `reviewee.fbAccessToken`, `reviewee`.`ivleAccessToken` AS
`reviewee.ivleAccessToken`, `reviewee`.`isRegistered` AS `reviewee.isRegistered`,
`reviewee`.`haveFetchedModules` AS `reviewee.haveFetchedModules` FROM `reviews` AS
`reviews` LEFT OUTER JOIN `users` AS `reviewer` ON `reviews`.`idReviewer` =
`reviewer`.`id` AND `reviews`.`idReviewer` LEFT OUTER JOIN `users` AS `reviewee` ON
`reviews`.`idReviewee` = `reviewee`.`id` AND `reviews`.`idReviewee` WHERE
`reviews`.`idReviewee` = '8' LIMIT 0, 10;
```

This query is executed while fetching reviews written by and for a user. It first does a left outer join of the users table with the reviews table with the userId matching the reviewer's ID and then another one from that to the user's table with the reviewee's ID matching the user ID.

Query 3:

```
SELECT `id`, `createdAt`, `updatedAt`, `name`, `matricNumber`, `fbId`, `fbAccessToken`,
`ivleAccessToken`, `isRegistered`, `haveFetchedModules` FROM `users` AS `users` WHERE
`users`.`matricNumber` IN ('a0118897', 'a0113672', 'a0111840', 'a0059806', 'a0111764',
'a0096765', 'a0113627', 'a0112054', 'a0102147', 'a0115503', 'a0119416', 'a0116603', 'a0114156',
'a0131125', 'a0127828', 'a0119447', 'a0110674', 'a0122356', 'a0133920', 'a0091372', 'a0113011',
'a0121437', 'a0112156', 'a0099429', 'a0112068', 'a0099112', 'a0110781', 'a0121520', 'a0091539',
'a0112171', 'a0111010', 'a0108385', 'a0111697', 'a0108232', 'a0126539', 'a0105952', 'a0099667',
'a0124368', 'a0111889', 'a0116208', 'a0132763', 'a0105786') AND `users`.`fbId` IS NOT NULL;
```

Executing (default): SELECT `id`, `uuid`, `userId`, `code`, `title`, `createdAt`, `updatedAt` FROM `modules` AS `modules` WHERE `modules`.`uuid` = '56119325-0ea4-41f9-94aa-24c3668b0a67' LIMIT 1;

This is executed in the module page to separate groupmates users vs students in the class who are not with groupmates. This query basically fetches all user rows if the matric number is in the class roster returned by IVLE. It uses a simple `where NOT IN` query

Aspiration 7: Show us some of your most interesting Graph queries. Explain what they are used for. (2-3 examples).

public_profile:
We use this to fetch a person's fbid and name to populate the profile initially. We also use this to link an IVLE user in our database to a facebook user once they sign up

friends_list:
TODO

Aspiration 8: We want some feeds! BUT remember to put thought into this. Nuisance feeds will not only earn you no credit but may incur penalization!

We created a share dialog in each review for users to share that review. User can decide to publish it or not. More important, we think that review is the most important thing of the web-site. So user may want to share good/bad reviews for discussion, of his/her own or others.

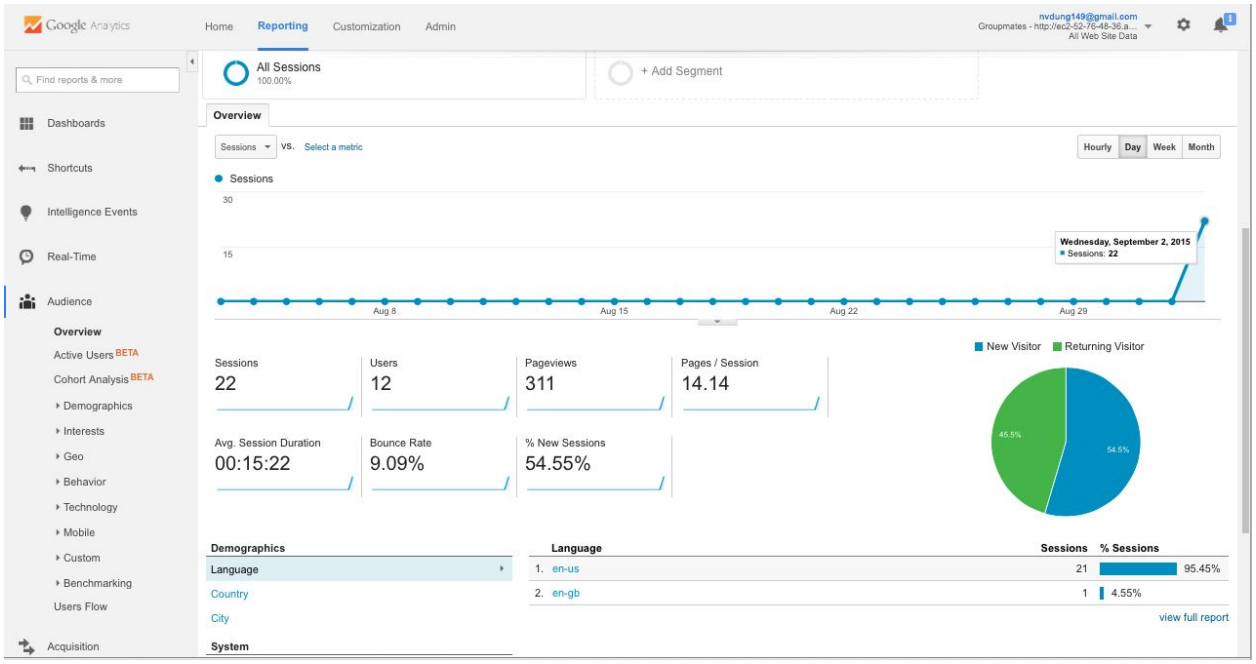
Aspiration 9: Your application should include the Like button for your users to click on. Convince us why you think that is the best place you should place the button.

We included the like button besides the share dialog for each review. With the same reason (as aspiration 8), as review is the root of the app. It also creates a way for user can express their agreement on the review.

Aspiration 10: Explain how you handle a user's data when he removes your application. Convince us that your approach is necessary and that it is the most logical. Do also include an explanation on whether you violate Facebook's terms and conditions.

We remove the user's profile data and modules, but we still persist the reviews. [TODO]

Aspiration 11: Embed Google Analytics on all your pages and give us a screenshot of the report.



Aspiration 12: Describe 3 user interactions in your application and show us that you have thought through those interactions. You can even record gifs to demonstrate that interaction! It would be great if you could also describe other alternatives that you decided to discard, if any.

- After logging in with Facebook account, users can go directly to profile page and connect to IVLE account. We originally require IVLE authentication in registration phase but removed it for better user experience
- Users can leave a review for other people, after that they will be redirected to the reviewee's profile page.
- Users can like or share the reviews on Facebook. The sharing link will lead to a single review view and can be accessed without logging in.

Aspiration 13: Show us an interesting DOM manipulation through jQuery that occurs in your application.

On the review pages, we store a template of review widgets. Every time a user scroll down to the bottom of the page, or clicking the down arrow for loading more reviews, new review data is loaded asynchronously and each review data item is mapped to new template instances. These

new review widgets are then inserted into the page through jQuery DOM manipulation functions.

Aspiration 14: Implement at least one Story involving an Action and Object in your application. Describe why you think it will create an engaging experience for the users of your application.

Aspiration 15: Describe 2 or 3 animations used in your application and explain how they add value to the context in which they are applied. *(Optional)*

Aspiration 16: Describe how AJAX is utilised in your application, and how it improves the user experience. *(Optional)*

We use AJAX to load reviews on a user's profile after the page is loaded. This has 2 UX benefits:

a. The page loads much faster
b. We can implement pagination like Facebook's news feed. We only load 10 at a time and load more and more once the user scrolls down. This prevents loading too much of data and slowing down the initial AJAX fetch.

Aspiration 17: Tell us how you have made use of any existing jQuery plugins to enhance your application functionality. Impress us further by writing your own reusable jQuery plugin. *(Optional)*

We are using Waypoint jQuery to trigger loading more reviews when a user scrolls down to the bottom of the page. Waypoint is a plug-in that monitors the position of "waypoint" elements in the page and triggers custom events when they hit certain positions on the viewport. It enables users to automatically view more reviews without explicitly making a request, and allows them to scroll down the page continuously without much interaction.