Milestones Report

Product Name: BOJIO
Team: Liu Zhiqun, Lai Cheng Yu, Stefano Chiesa

Milestone 1:

Standalone App

We chose to do a standalone app as we wanted to create a platform that taps on some of Facebook's social features but yet gives us the freedom to make use of other APIs such as GIPHY and other animations to bring life and character to our social platform!

Milestone 2: Explain your choice of libraries

Server side

We first contemplated the use of flask/Django; python-based server-side frameworks which were suggested by Stefano, who was the most experienced developer within our team. However, the rest of the team did not have extensive experience in python frameworks and were more familiar with JavaScript/Node.js-based frameworks instead.

Once we established that we were all comfortable with JavaScript/Node.js, we narrowed in on Express.js and MeteorJS (that Cheng Yu suggested as he had prior experience with it). We ended up dismissing MeteorJS rather quickly though, as Meteor comes bundled with MongoDB as its database (which was noSQL and non-relational) and would have needed extra work to work with mySQL (our choice of database management system).

Thus, we eventually decided on Express.js: a fast, unopinionated, minimalist web framework for Node.js. which had plenty of documentation for setting up with that made it easy to quickstart our backend. In addition, there were existing documentation for Express.js with our frontend and boilerplate choice (ReactJS). Also for connecting with the database, we use sequalize ORM because Stefano has used it before and the main consideration beside using other ORM is it is a Promise based framework which we are familiar working with.

Client side

For our frontend, we decided off the bat to go with ReactJS as our frontend rendering library and create-react-app for our boilerplate. ReactJS's state changes are easy to manipulate and components are easy to test as they update immediately with a webpack that hot reloads. Learning ReactJS is also easy compared to others with a less verbose/convoluted API. Additionally, we could install chrome extensions such as "React Developer Tools" that help to inspect and manipulate each component's state and props.

For our frontend framework, we chose semantic-ui-react as it was well documented, easy to use and had a really sleek design that was very suitable for BoJio! as it is an application for more casual purposes.

We also chose to use create-react-app as our boilerplate as it was officially supported by Facebook and was great for skipping all the build configuration.

Bonus Questions!

Pros and cons of each method of visibility control:

By using Javascript and CSS to control visibility, you can instantly sync rendered DOM correspondingly after fetching API from client side or after doing client side authentication like Facebook SDK without having to wait for a server reload, whereas using server-side rendering, we have to reload the whole DOM when something changed on the client which is a problem for single-page application.

Using server-side rendering is great when the render condition depends on sensitive information from the database that we want to keep from the user.

Javascript method vs Server-side method:

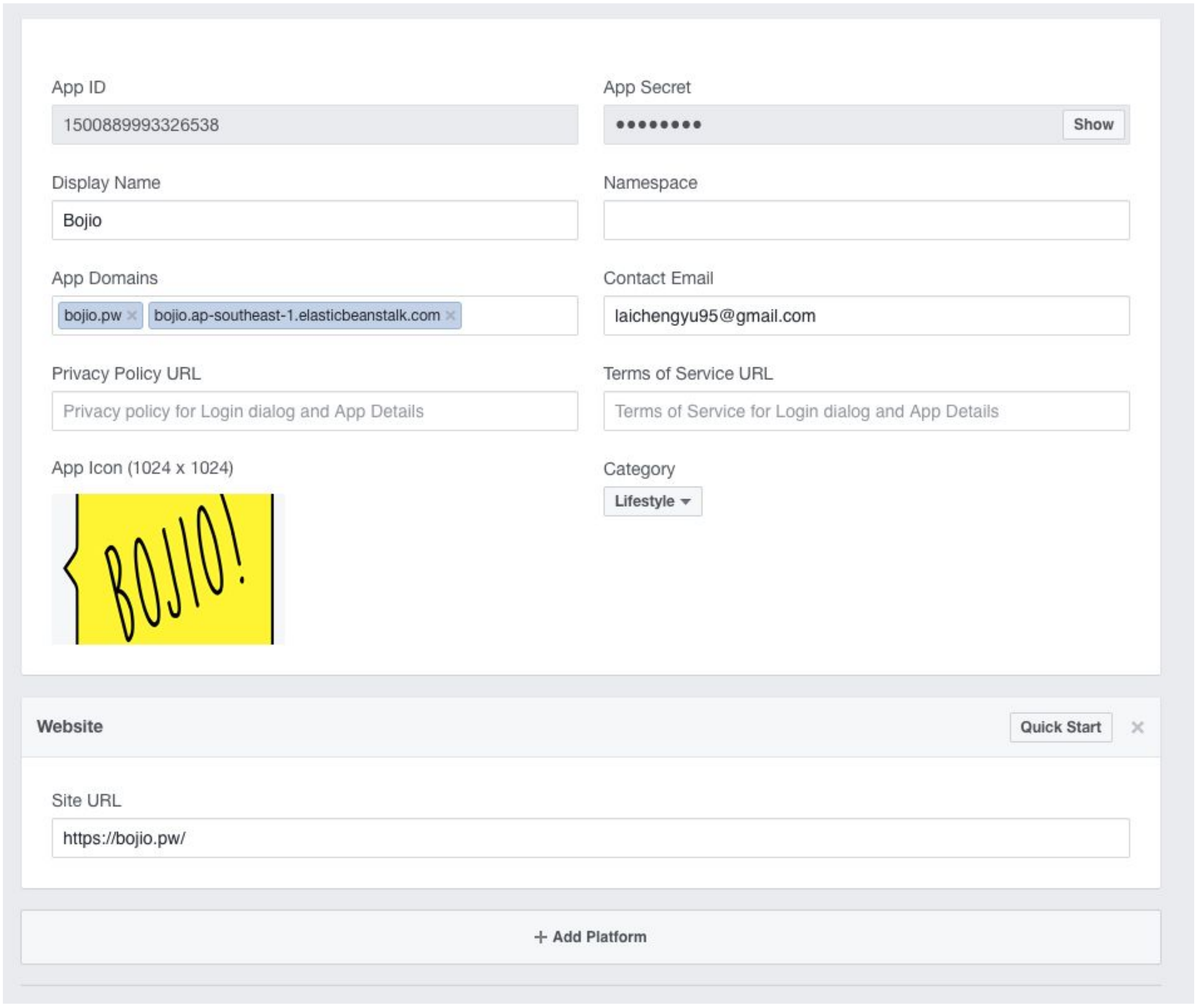
One should use Javascript method when they think they can take advantage of Facebook Javascript SDK UI components (login dialog, share dialog, feed dialog) . Using the Javascript method also gives users a better user experience overall by keeping the user in the application instead of being redirected to Facebook to login.

Server side method on the other hand redirects users to Facebook for authentication but able to simplify the server API params and also simplify enforcement of access control for preventing non-logged in users to access the API. It also allow heavy processing of facebook API call results to be done faster on the server.

What is the primary key of the home_faculties table? (0.5%)
A composite key of matric_no and faculty

Milestone 3: Screenshot of App details page!

Screenshot of our Dashboard!



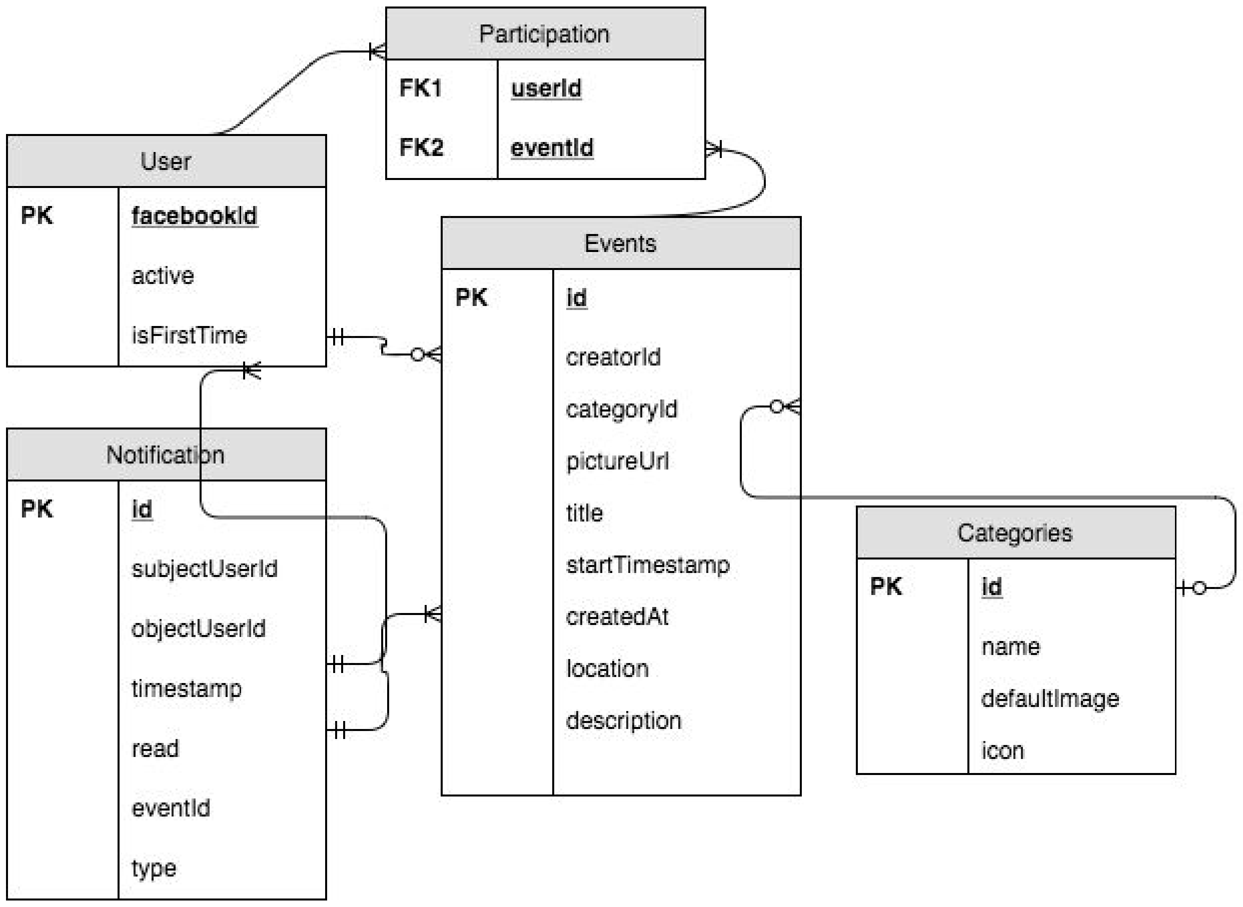
App ID	App Secret
1500889993326538	•••••••• Show
Display Name	Namespace
Bojio	
App Domains	Contact Email
bojio.pw, bojio.ap-southeast-1.elasticbeanstalk.com	laichengyu95@gmail.com
Privacy Policy URL	Terms of Service URL
Privacy policy for Login dialog and App Details	Terms of Service for Login dialog and App Details
App Icon (1024 x 1024)	Category
	Lifestyle

Website

Site URL

https://bojio.pw/

Milestone 5: Database Schema



Milestone 6:

Query for past events joined (sorted by most recently ended and hosted by an active user):

```
SELECT *
  FROM Participation, Events, Users
  WHERE Participation.EventId == Events.id
    AND Events.date > {currentTimestamp}
    AND Participation.UserFacebookId == {currentUserId}
    AND Events.creatorFacebookId == Users.facebookId
    AND Users.isActive == '1'
  ORDER BY Events.date DESC
  LIMIT 10;
```

Query for user created events (sorted by most recently created)

```
SELECT *
  FROM Events
  WHERE Events.creatorFacebookId == {currentUserId}
  ORDER BY Events.createdAt DESC
  LIMIT 10;
```

Query for fetching unread notifications of current users:

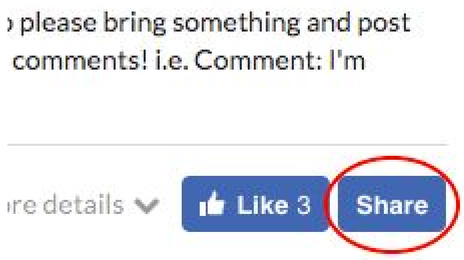
```
SELECT *
  FROM Notifications, Users
  WHERE Notifications.subjectUserId == {currentUserId}
    AND Notifications.objectUserId == Users.facebookId
    AND Users.isActive == '1'
    AND Notifications.read == ‘0’
  ORDER BY Notifications.timestamp DESC;
```

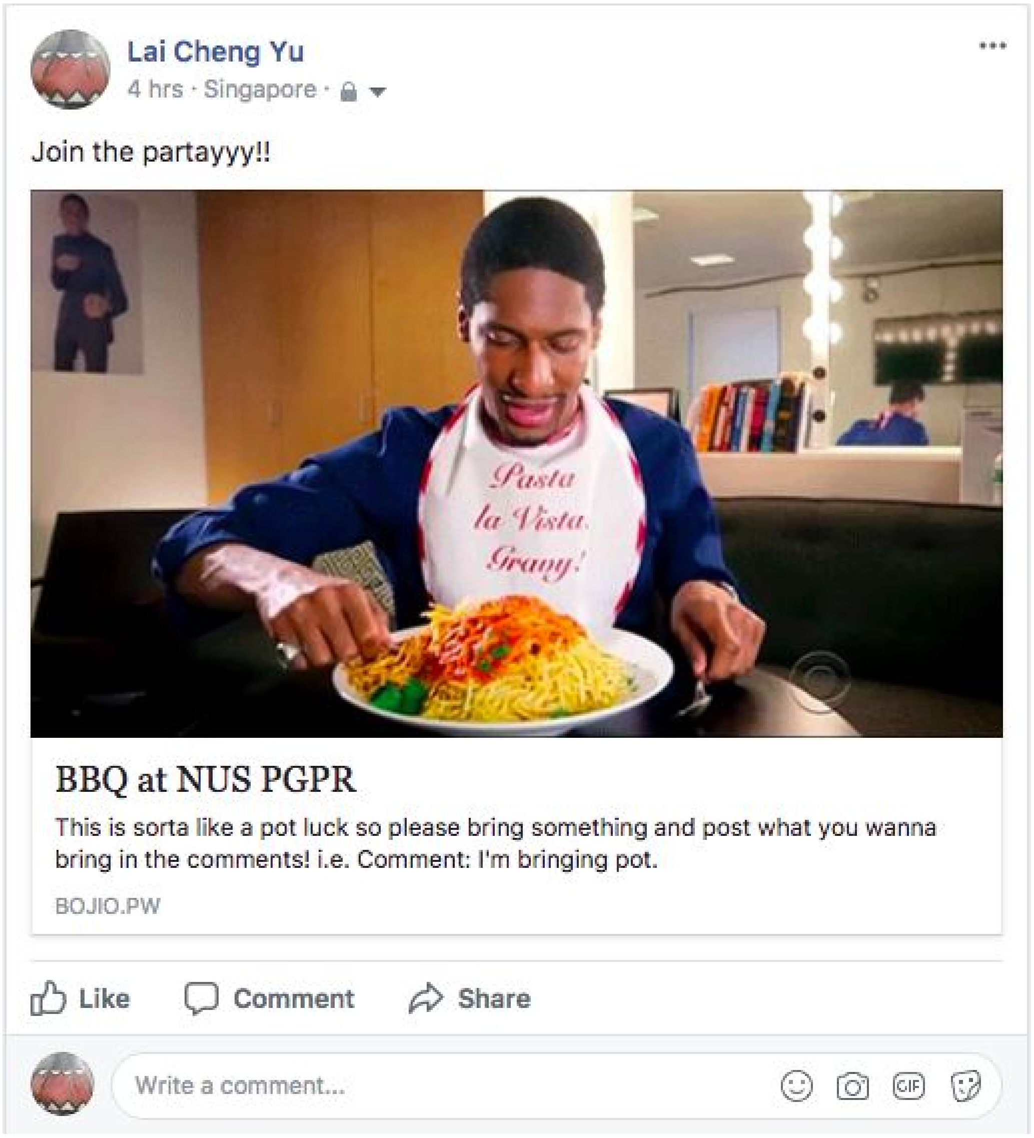
Milestone 7:

- In BoJio!, we need to add a feature for inviting friends to an event. We decided on creating a tokenizer with an autosuggestion. For the autosuggestion, we have to query graph api in order to find the list of friends.
  We do this by querying Graph API `/me/friends` with access token and
  `fields: 'id, name, first_name, last_name, picture'`
- We also decided to only store facebook id of users in our application. We do this in order to not store any personal information, fetch the latest information, and simplicity when users decided to deauthorize us. The trade off is the performance where we have to make an additional API call to `/{facebookId}` in order to fetch users information, but this is not a big of a deal for a small scale application and that’s to caching and events pagination that we did.

Milestone 8:

The feed we have implemented is the ‘Share’ button for each event, which posts the event’s individual page link with the event image, event title, event description and any (customizable) message in the post that the user wishes to add, onto his/her Facebook timeline and news feed.


› please bring something and post
comments! i.e. Comment: I'm
ıre details
Like 3
Share


Lai Cheng Yu
4 hrs · Singapore ·

Join the partayyy!!

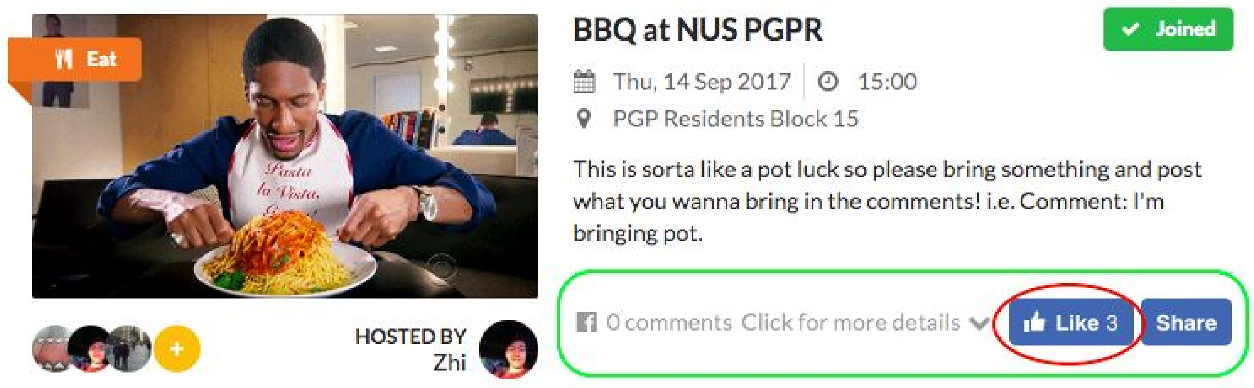
BBQ at NUS PGPR

This is sorta like a pot luck so please bring something and post what you wanna bring in the comments! i.e. Comment: I'm bringing pot.

BOJIO.PW

Our rationale behind implementing this ‘Share’ button is so that the user’s Facebook friends (especially those who are not on our app) are able to see the user’s activities (hosted/joined events) and be intrigued enough to click on the link in order to view the event or wanting to participate in the event, and thus try out BoJio!. The user’s message in the post can also act as advertisement as it would typically be invitations by the user to his/her Facebook friends to join the event. Inclusion of the event image/GIF into the post also helps to make the event look more lively and fun, which would be able to draw more interest from others.

Milestone 9:



BBQ at NUS PGPR

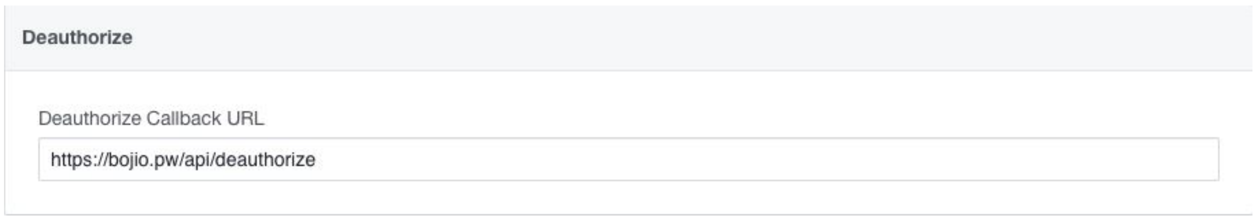
Thu, 14 Sep 2017 | 15:00

PGP Residents Block 15

This is sorta like a pot luck so please bring something and post what you wanna bring in the comments! i.e. Comment: I'm bringing pot.

Our Like button is placed next to our Share button, in-line with the Comments button in each Event block. This is so that it is intuitive for users to associate clicking the Like button with liking the event itself, and for all our Facebook interaction buttons to be grouped in the same section (bottom-right) of the Event block for better UI; i.e. users only need to look to the same, one section for all their Facebook interactions.

Milestone 10:



Deauthorize

Deauthorize Callback URL

https://bojio.pw/api/deauthorize

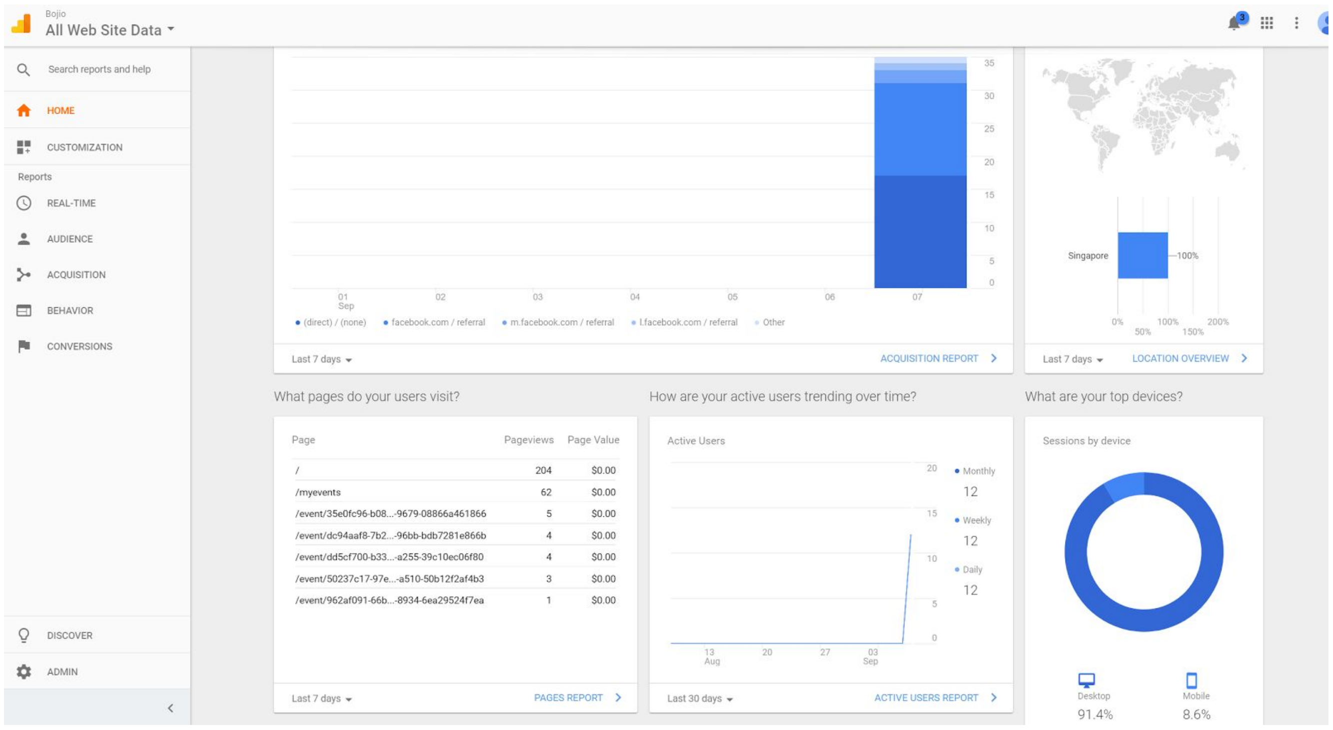
When users deauthorize us, we simply set users `isActive` column in `Users` table to false.

From active users point of view, deactivated users would no longer be seen in any events participant list and events created by deactivated users also would not be visible. When any deactivated user re-authorize us, their created events would be visible again and they will added to their previously joined events.

This approach comply with the Facebook's terms and conditions since we are not storing any information that is identifiable to an individual or in other words the data of deactivated users that we store are anonymous.

Milestone 11:

Google Analytics



Page	Pageviews	Page Value
/	204	$0.00
/myevents	62	$0.00
/event/35e0fc96-b08...-9679-08866a461866	5	$0.00
/event/dc94aaf8-7b2...-96bb-bdb7281e866b	4	$0.00
/event/dd5cf700-b33...-a255-39c10ec06f80	4	$0.00
/event/50237c17-97e...-a510-50b12f2af4b3	3	$0.00
/event/962af091-66b...-8934-6ea29524f7ea	1	$0.00

(Screenshot of our google analytics page)

We have added Google Analytics to our web-app and is thus able to track the page view counts and user's demography in the following pages:

1. Home Page
2. My Events Page
3. Individual Event Pages

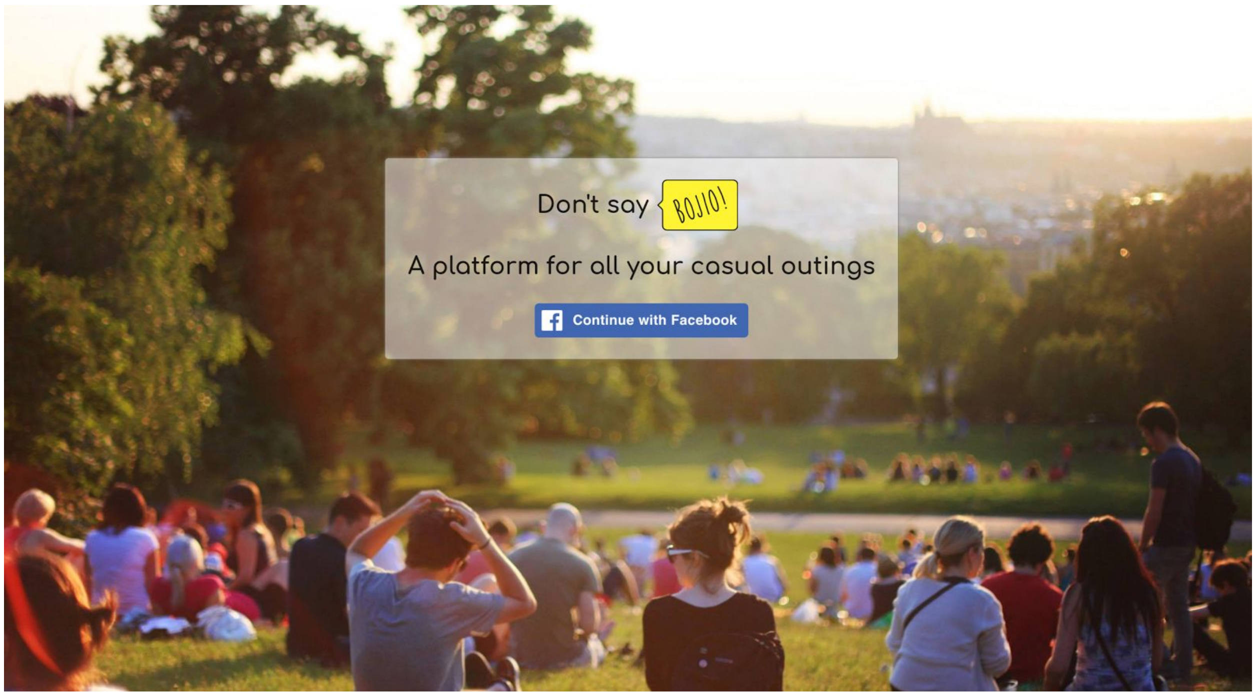
The analysis results will assist us in incremental improvements for the design of our UI and will also give us insights on the kind of events that our users prefer - which will be crucial in our marketing efforts.

Milestone 12:

3 User interactions

We want our app to reflect on the 3 values that we have recognized in organizing meetups. Fun, simple and interactive! As such, we decided to base our user experience design on those 3 crucial factors!

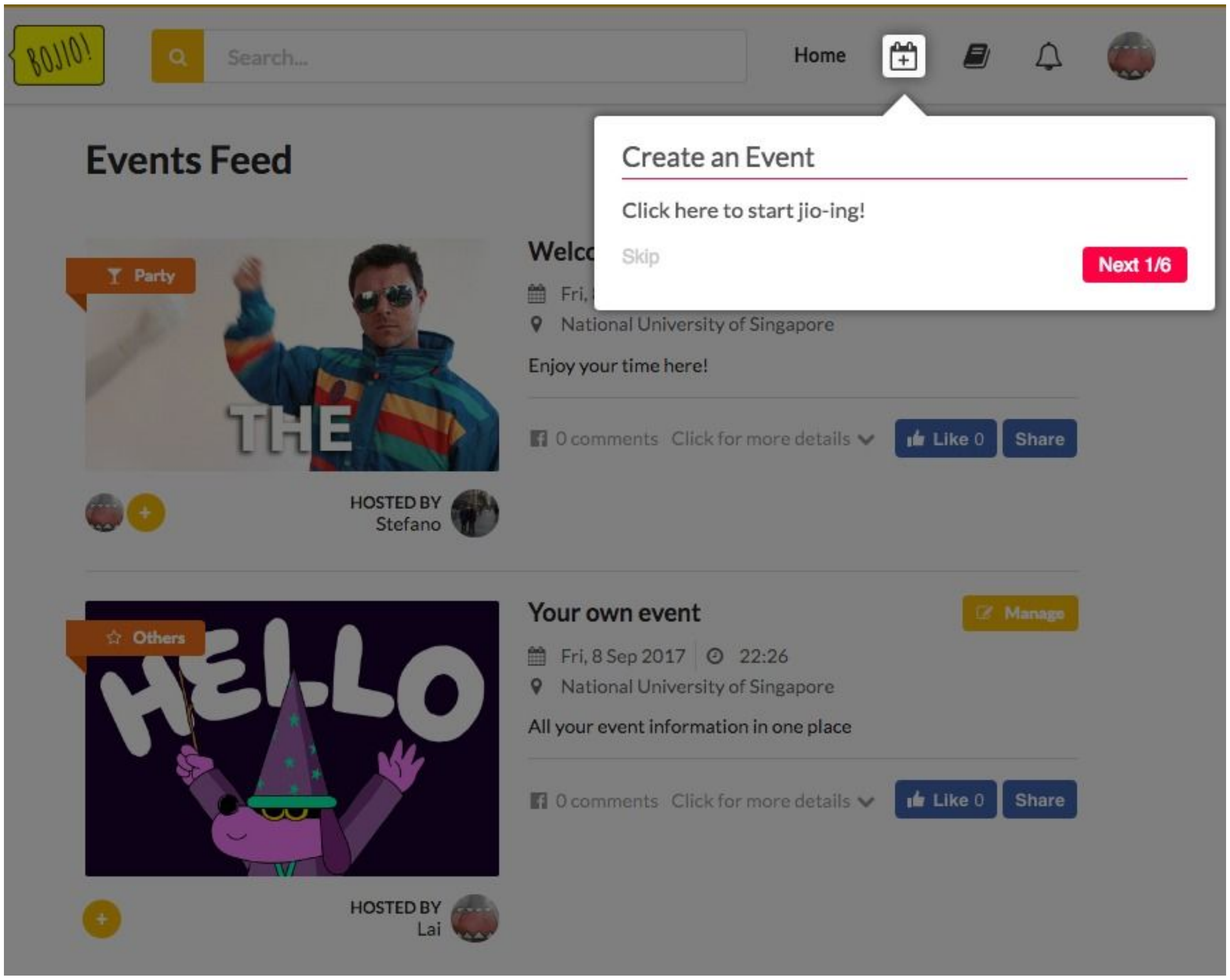
1. **INTERACTIVE onboarding flow**



(BoJio! Splash page)

Bojio feels that a good onboarding flow is crucial as it sets the first impression of the app. Thus, we scraped the initially boring and one-dimensional instructions based model to the currently more interactive one.

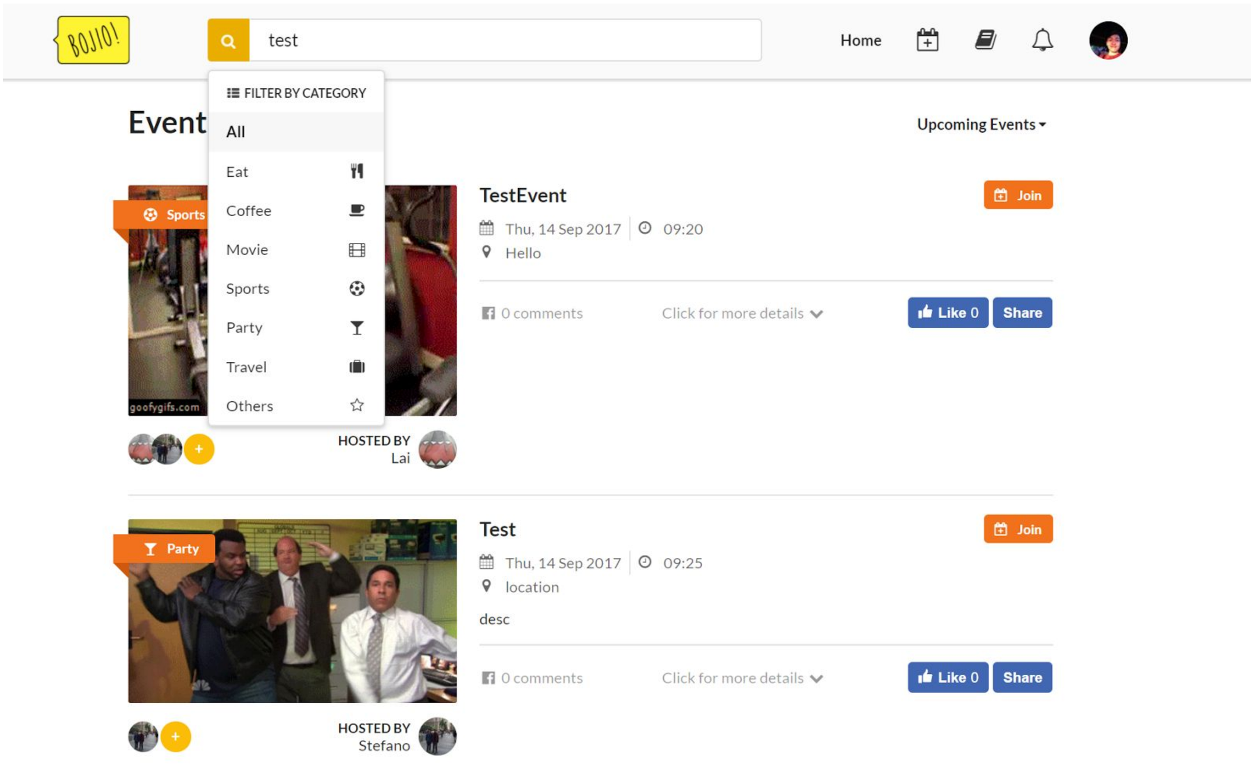
Our splash page is designed to be simple and straightforward with the intention of giving our users a sense of who we are and what we do. Thus, we added in our tag line; ‘Don’t say Bojio’ and our one-liner description with a simple call to action button - Connect with Facebook.


BOJIO!
Search...
Home
Events Feed
Create an Event
Click here to start jio-ing!
Skip
Next 1/6
Party
Welc
National University of Singapore
Enjoy your time here!
0 comments Click for more details
Like 0
Share
HOSTED BY Stefano
Others
Your own event
Manage
Fri, 8 Sep 2017 22:26
National University of Singapore
All your event information in one place
0 comments Click for more details
Like 0
Share
HOSTED BY Lai

(Interactive onboarding animation)

When our users have connected to Facebook and is greeted by our home/ events feed page, BoJio! Will then welcome users with an interactive animated guide which will introduce some of the key functions of our web-app including: Search, My Events, Create Events, Events Feed etc. This onboarding flow will then disappear upon completion and will not reproduce so as to not ‘annoy’ users in the future. This feature will be further elaborated in Milestone 15 - Animations.

2. **SIMPLE search feature**

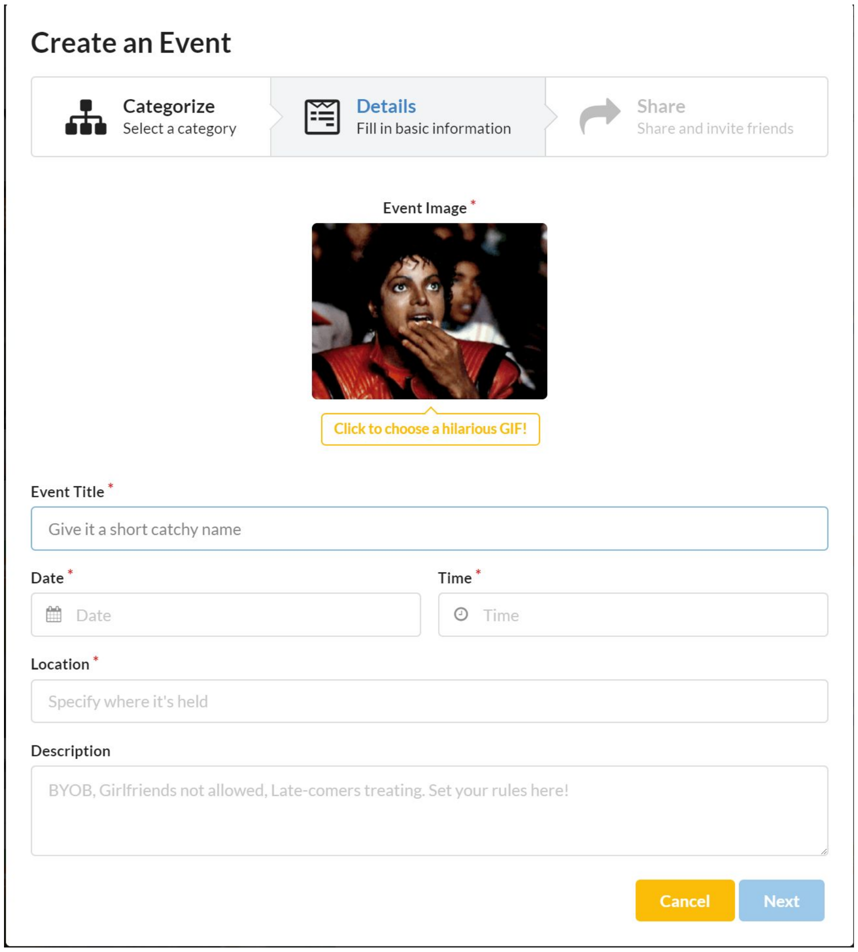


For an event organizing app, we feel that the search bar is one of the most important features as massive amounts of events may appear messy and intimidating to users. Thus, we aim to simplify the process of search through a no-click autofilter manner.

Users can simply search for a particular event's title on the events feed page, and the system will auto-populate the relevant events. All of these process with no 'entering' or clicking required!

We have also created a category filter drop down for users who knows which category their events belongs to. The simplistic design of the title and icon for each category (in our opinion) creates a soothing user experience which may come in handy especially during stressful last minute searches!

3. **FUN events creation**



Event Title *

Give it a short catchy name

Date *

Date

Time *

Time

Location *

Specify where it's held

Description

BYOB, Girlfriends not allowed, Late-comers treating. Set your rules here!

(Event Creation Pop-up - Details page)

Events creation has always been known as the most boring part of every event organizing sites, ie. Eventbrite, meetup.com etc. Bojio hopes to change that we our user experience design!

Firstly, we have decided to create a ‘pop-up’-ish interface to suggest or subtly hint to our users that the process is a quick and easy one! By displaying a 3-step guideline, our users will immediately understand how the entire process works and which step are they currently at.

Secondly, we minimize the number of input fields that our users have to run through and chose the most intuitive input formats that are available. For instance, instead of letting our users key in the date and time manually or choosing a timestamp, we separated the date from the time input to make things clearer. Additionally, we have blanked out any ‘past’ dates just in case our users misclicks to ensure accuracy in the data they have input.

Lastly, we try to make the process of event creation as FUN as possible by inserting fun and relatable placeholders and through adding in a GIPHY image feature!

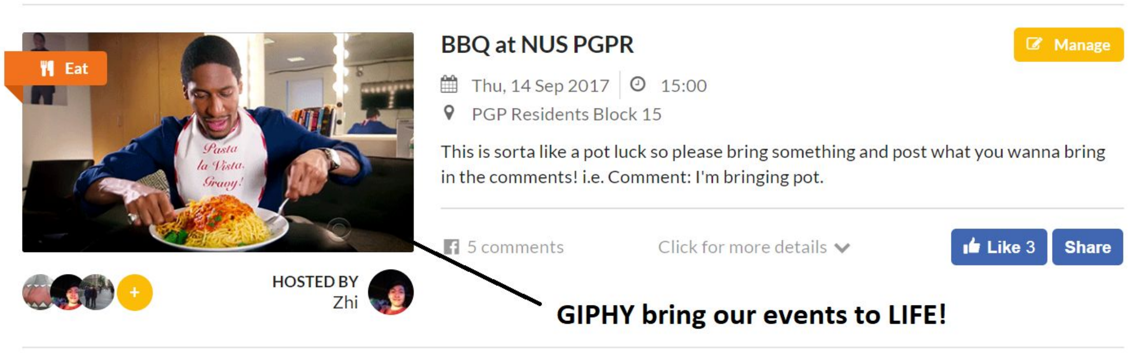
We will be elaborating more on this feature in the following milestone - Milestone 13!

Milestone 13:

Cool Feature showcase - GIPHY FEATURE

Being FUN is one of the key values in the Bojio team. As such we want to transmit our fun-loving attribute to our app!

Apart from the simplistic and **casual** user-interface design we have decided to incorporate a GIPHY feature to bring out the inner (FUN) personalities of our users and the events that they have!

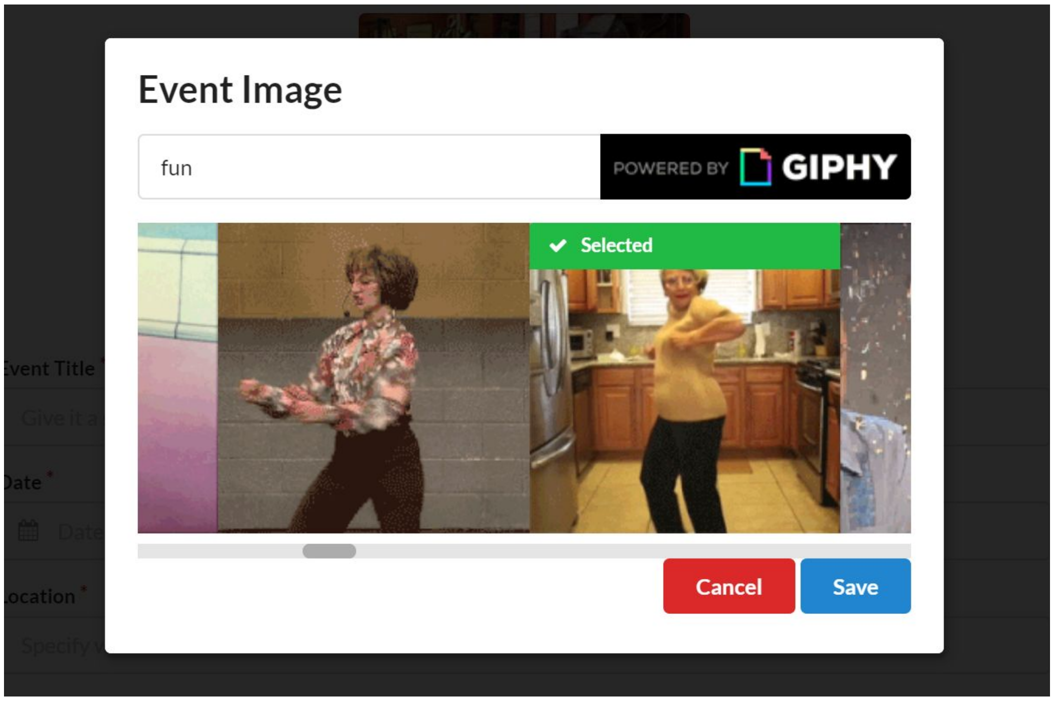


This is sorta like a pot luck so please bring something and post what you wanna bring in the comments! i.e. Comment: I'm bringing pot.

(Example of an event tile in our Event Feeds)

Through making use of the GIPHY API, our users are able to search for cool GIFs from the GIPHY database to effectively and animatedly express themselves and the event they are creating.

To make things simpler for our users, we have set a default GIF for each of the categories that our users are about to choose when creating a new event. In the case where a specific user intends to be more expressive, they have the option to choose a different GIF by making a search in the GIPHY database. He or she can then select from the row of cool GIFs that will be generated according to his or her search keyword!



(Ability to edit the GIF of your event - during event creation)

Milestone 14:

One of possible way to prevent CSRF is to use a token associated with a user that is generated by the server, this will be used as a hidden value on every state changing form on the website. This token will be send back to the server when we send a request to the server and if the token does not match, it will be rejected by the server. This approach protects the form from CSRF because the attacker will have to guess token to send in their forged request in order to send a valid request.

As for protection against XSS, we can do that by encoding untrusted data from the user before displaying it as data and make sure it is not executed as a code.

- For displaying HTML, we can convert:
    - " to &quot;
    - ' to &#x27;
    - / to &#x2F;
    - & to &amp;
    - < to &lt;
    - > to &gt;
- For URL, we use standard URL encoding
- For Javascript, we could escape each character into a \uXXXX (unicode format) except for alphanumeric characters.

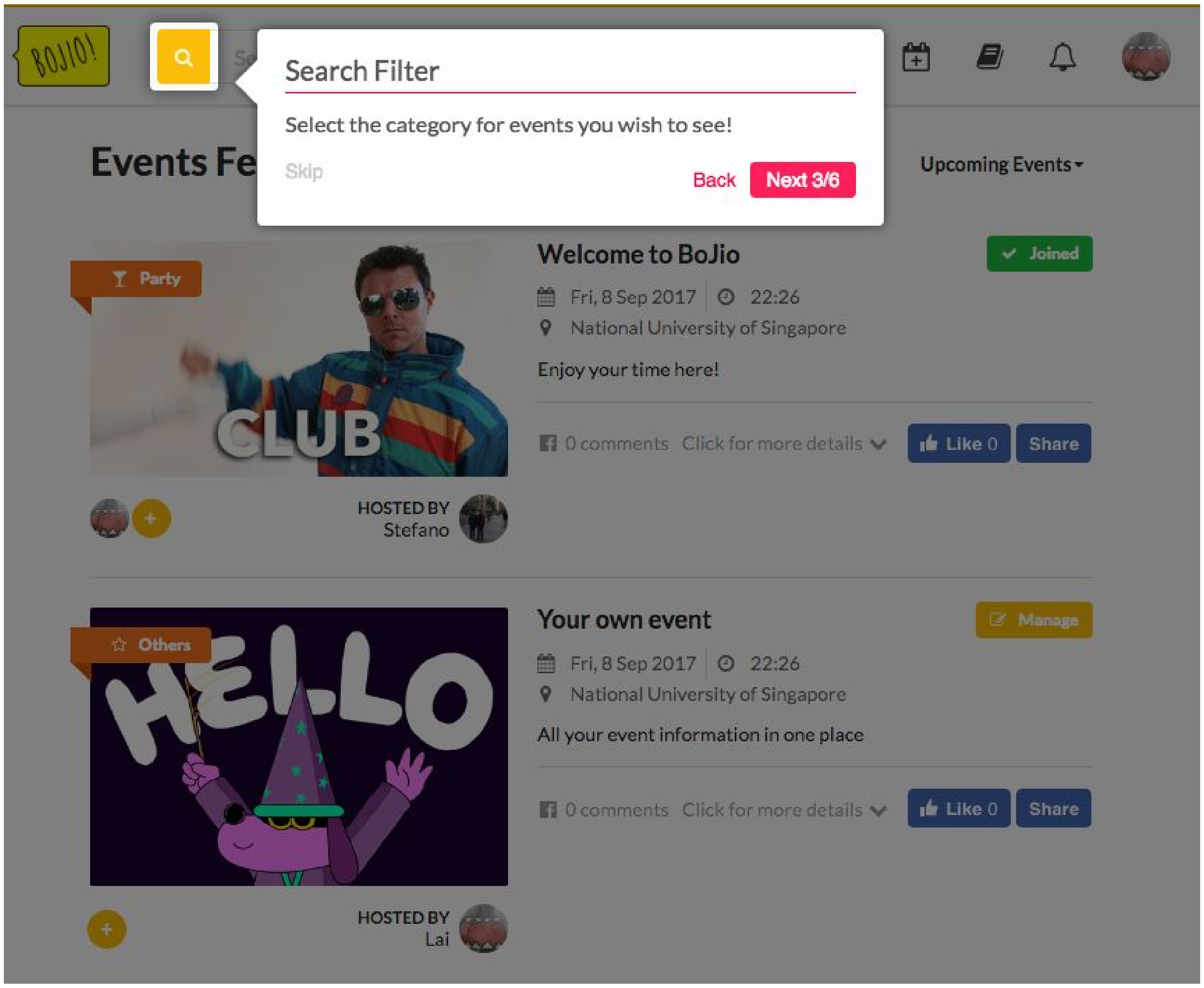
How is BoJio! protected from those attacks:

- **SQLi**
  One of the way to ensure your app is protected from SQLi is by using ORM and make sure to never execute a raw query using the ORM since they are well tested and always encode the argument value. BoJio! uses sequelize for ORM and we make sure to never use `sequelize.query` which is open to SQLi attack.
- **XSS**
  We always treat user textual input as a string and use is as part of the code logic.
- **CSRF**
  We use csurf (https://github.com/expressjs/csurf) for Express.js middleware which is well used to protect against CSRF**.** Since BoJio! Is a single page application, the server sends out a token after login to the homepage and use that token for every POST request to the server for verification.

Milestone 15:

BoJio! uses 3 animations, namely the throbber, onboarding sequence for new users and GIFs.

The throbber helps to let users know that BoJio! is currently retrieving data from the servers, and that their input has been received. This serves as a form of feedback for users, so that they would not be unsure as to whether their clicks have been registered or not (or if they even clicked correctly).



Search Filter

Select the category for events you wish to see!

Skip Back Next 3/6

Events Fe

Upcoming Events

Welcome to BoJio

Fri, 8 Sep 2017 | 22:26

National University of Singapore

Enjoy your time here!

0 comments Click for more details

HOSTED BY Stefano

Your own event

Fri, 8 Sep 2017 | 22:26

National University of Singapore

All your event information in one place

0 comments Click for more details

HOSTED BY Lai

The onboarding sequence helps to make the new user experience be seamless by introducing them to the basics they need to know to best enjoy BoJio!. In a sequential manner, the user is directed to the various icons in the page that pertain to BoJio!'s basic functionalities, along with guiding text that briefly explain what the icons do or what the user can do upon clicking them.

The GIF animations mainly help make the events look more lively and fun, and for further elaboration, please refer to Milestone 13.

Milestone 16:

Our main utilisation of AJAX is in our usage of window.fetch JavaScript polyfill in order to send AJAX requests to our server. This helps us to update the list of events we should be showing to users based on any input filters such as the search bar, current page viewed, etc all without re-rendering the page.

This helps to reduce lag for users as less objects are being rendered upon every interaction. It also improves user experience as re-rendering tends to indicate a change or update and to re-render a particular element in the page without any changes made to it would be misleading and confusing to users. This way, un-updated elements remain static in position.

Lastly, new notifications also appear for users without refreshing the page using the same methods. This helps to keep users engaged actively with BoJio! as they would be able to see their notification immediately as they are received rather than on refresh (which they may not do at all until they leave and re-enter BoJio!).