



Team 2 Milestones Submission

A0141138N

A0136070R

A0126172M

A0148038A

Apoorva Ullas

Lim Jia Yee

Won Jun Ru Daphne

Li Zihan

Milestone 1: Choose to do a "Facebook Web Games" application, a standalone application, or both. Choose wisely and justify your choice with a short write-up.

We believe that creating a standalone application will give us greater flexibility in developing our application. We want to keep the option of expanding our user base to other social media users, or those without Facebook accounts in the future. The feeds of our web-application will be hosted on Facebook and what our application offers is the anonymity of the user for both posting and replying.

Milestone 2: Explain your choice of libraries and what alternatives you have considered for your Facebook application on both the client-side and server-side. If you have decided to go with the vanilla approaches (not using libraries/frameworks for the core application), do justify your decisions too.

First, we used a full-stack web framework, because frameworks simplify the development of the app and let us focus on the high-level architecture rather than the low-level, basic, repetitive tasks like socket programming.

SERVER

We used the Laravel PHP framework for our server-side development. We do not want to choose microframeworks such as Flask and Sinatra as they are intended for smaller projects with specialized purposes. For example, for a static website, if we need a simple backend service for one or two forms, we can send the forms from client-side to a Flask backend.

In addition, we can expect more unnecessary labour in configuring microframework apps with the amount of non-trivial features we intend to expand towards. On the other hand, frameworks such as Laravel and Ruby on Rails support the philosophy of "convention over configuration". We can then focus our time on actually developing the app and leave the standards and specifications to the framework. This saves our time, and also helps with debugging and future handover. The community of both Ruby on Rails and Laravel are large, and with most of them sticking with the conventions, we can expect to debug our problems faster, without the distraction of configuration, setup, and variables.

As for the other frameworks like Django, NodeJS, and so on, we will get more flexibility from them compared to the aforementioned two MVC frameworks above. However, we foresee a steeper learning with these due to lack of familiarity. The MVC architecture, on the other hand, is understood well by the team.

In the end, we chose Laravel over Ruby on Rails and other PHP frameworks. PHP is easier to pick up for us as all of us are proficient with HTML, CSS, and JavaScript, but only one of us is familiar with Ruby. And as for Laravel, we appreciate that it emphasises “convention over configuration”. Such philosophy are not emphasised in other PHP frameworks, which could lead to us having problems to debug (less helpful answers on the internet).

CLIENT

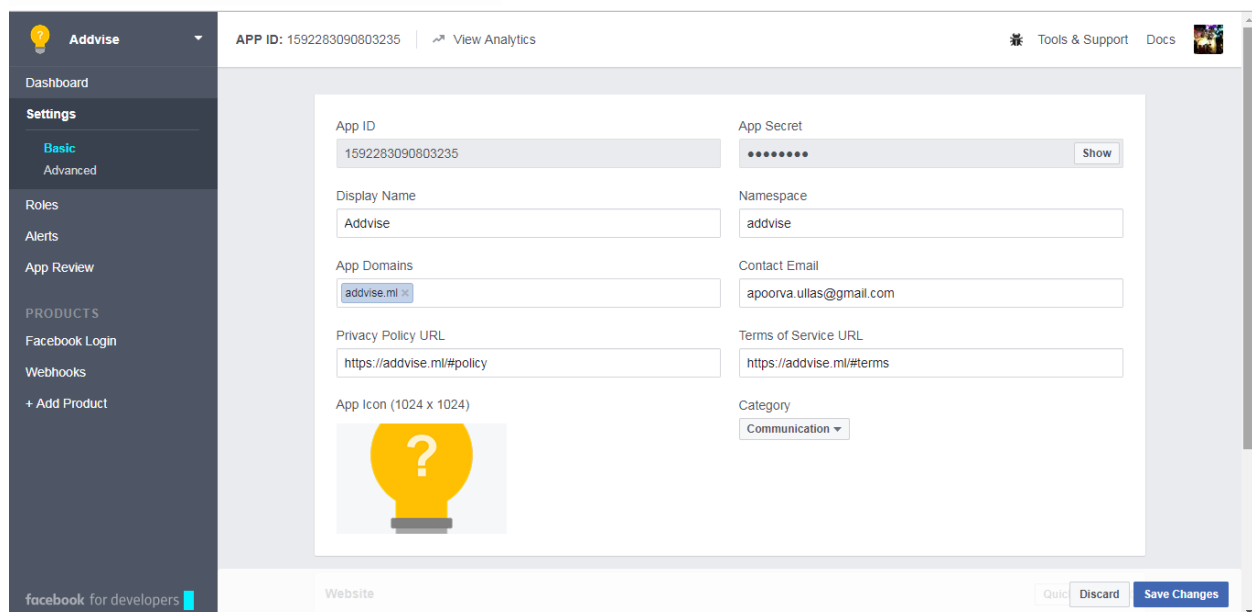
We picked bootstrap as it fits well with Laravel framework. It contains some templates based on HTML and CSS, which makes it efficient to design user interface and implement front-end development. The interface is really pretty and easy to pickup that we have already implemented.

Milestone 3: Give your newborn application some love. Go to the App Details page and fill the page with as much appropriate information as you can. And yes, we expect a nice application icon! Screenshot the dashboard fields for your midterm submission.

Icon: 

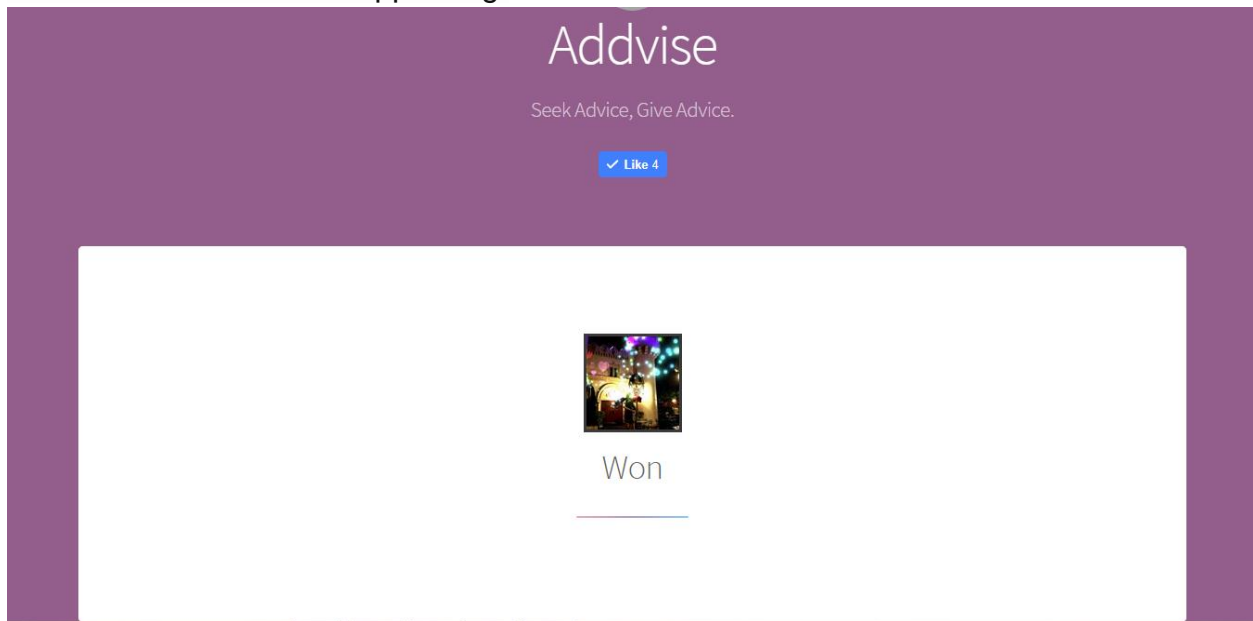
The icon in simplicity is actually a lightbulb with a question mark inside. The question mark implies that users can feel free to ask questions and seek for advice in this application. The lightbulb signifies the advice and tips given by other users, which could guide the life of those who request for advice.

Screenshot of latest Dashboard:

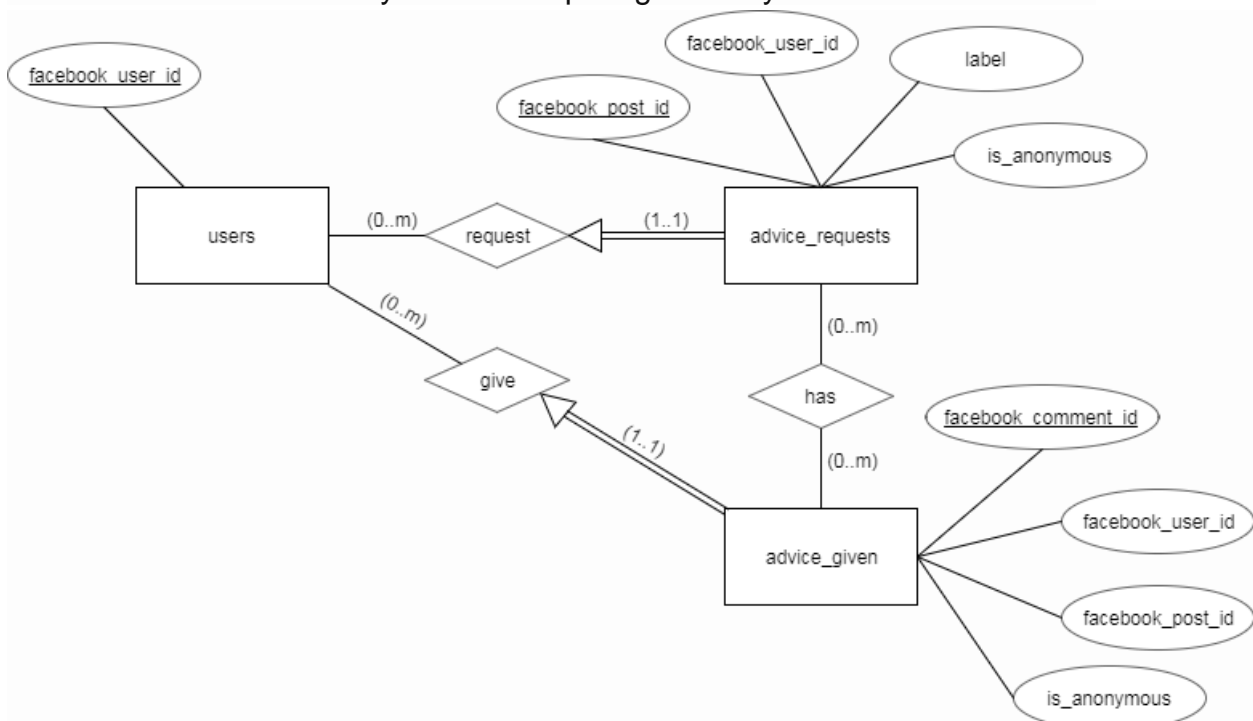


The screenshot shows the Facebook App Dashboard for an application named 'Advise'. The left sidebar contains navigation links: Dashboard, Settings (Basic, Advanced), Roles, Alerts, App Review, PRODUCTS, Facebook Login, Webhooks, and + Add Product. The main content area displays the 'App Details' page. At the top, it shows the App ID (1592283090803235) and a 'View Analytics' link. Below this, there are several input fields for app configuration: App ID, App Secret (with a 'Show' button), Display Name (set to 'Advise'), Namespace (set to 'advise'), App Domains (set to 'advise.ml'), Contact Email (set to 'apoorva.ullas@gmail.com'), Privacy Policy URL (set to 'https://advise.ml/#policy'), Terms of Service URL (set to 'https://advise.ml/#terms'), App Icon (1024 x 1024, showing a lightbulb icon with a question mark), and Category (set to 'Communication'). At the bottom right, there are buttons for 'Quick', 'Discard', and 'Save Changes'.

Milestone 4: Integrate your application with Facebook. If you are developing a Facebook Facebook Web Games app, then users should be able to visit your app and at least see their name (retrieved using the API) on the page. Similarly, if you are developing a standalone app, users should be able to login to your app using their Facebook account and see their own name appearing.



Milestone 5: Draw an Entity-Relationship diagram for your database schema.



Breakdown of tables

Users	
Description	To store user settings
PRIMARY KEY	<i>facebook_user_id</i>

Advice_Requests	
Description	To store the advice posts
PRIMARY KEY	<i>facebook_post_id</i>
FOREIGN KEY	<i>facebook_user_id</i> reference to table Users <i>facebook_user_id</i>
Remarks	We store the ID so that we can fetch the relevant and updated posts from Facebook quickly.

Advice_Given (comments)	
Description	To store the facebook comments given to the advice
PRIMARY KEY	<i>facebook_comment_id</i>
FOREIGN KEY	<i>facebook_user_id</i> reference to table Users <i>facebook_user_id</i>
FOREIGN KEY	<i>facebook_post_id</i> reference to table Advice_Requests <i>facebook_post_id</i>
Remarks	Comments can only be owned by one user and each comment can be only under one Advice.

Milestone 6: Share with us some queries (at least 3) in your application that require database access. Provide the *actual SQL queries* you use (if you are using an [ORM](https://www.wikiwand.com/en/Object-relational_mapping), find out the underlying query) and explain how it works.

1. Inserting advice request form values into advice_requests table:

```
INSERT INTO advice_requests (fb_post_id, fb_user_id, label,  
is_anonymous)
```

```
VALUES ($fb_post_id, $fb_user_id, $label, $is_anonymous)
```

2. Retrieve all details about requests from advice_requests table to show in “Give Advice” section

```
SELECT * FROM advice_requests;
```

3. Inserting given advice form values into advice_given table:

```
INSERT INTO advice_given (fb_comment_id, fb_user_id, fb_post_id,  
is_anonymous)
```

```
VALUES ($fb_comment_id, $fb_user_id, $fb_post_id, $is_anonymous)
```

4. Retrieve given advice from advice_given table to show in “Give Advice” comments section.

```
SELECT fb_comment_id FROM advice_given;
```

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

Example 01: Posting requests for advice on our Facebook Advise page

Example 02: Retrieving requests and comments from Facebook Advise page

Example 03: Posting comments for advice on our Facebook Advise page

Example 04: Displaying username and profile picture of user in “My Profile”:

```
Query for username: FB.api('/me', function(response) {  
  callback(response.name);
```

```
});
```

```
Query for profile picture:
```

```
FB.api('/me/picture?type=normal', function(response) {  
  callback(response.data.url);
```

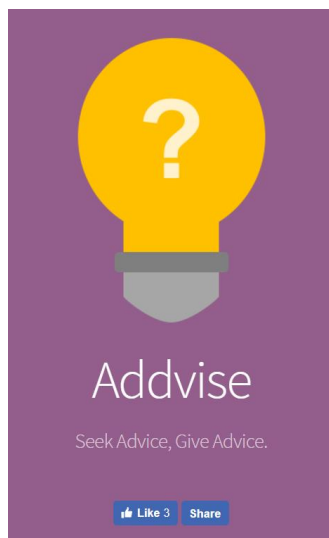
```
});
```

Milestone 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! Explain what feeds you implemented and your thought process for your feeds.

Our feeds are restricted to posts made on our Facebook Advise page by advice givers and advice seekers. We choose not to have any feeds published on our user's timeline as we provide a personalized "My Requests" and "Advice I Gave" section for users to track their personal requests and advises on our application. We also ensure there will not be spam posts by restricting a single user to maximum post 1 time every 5 minutes.

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

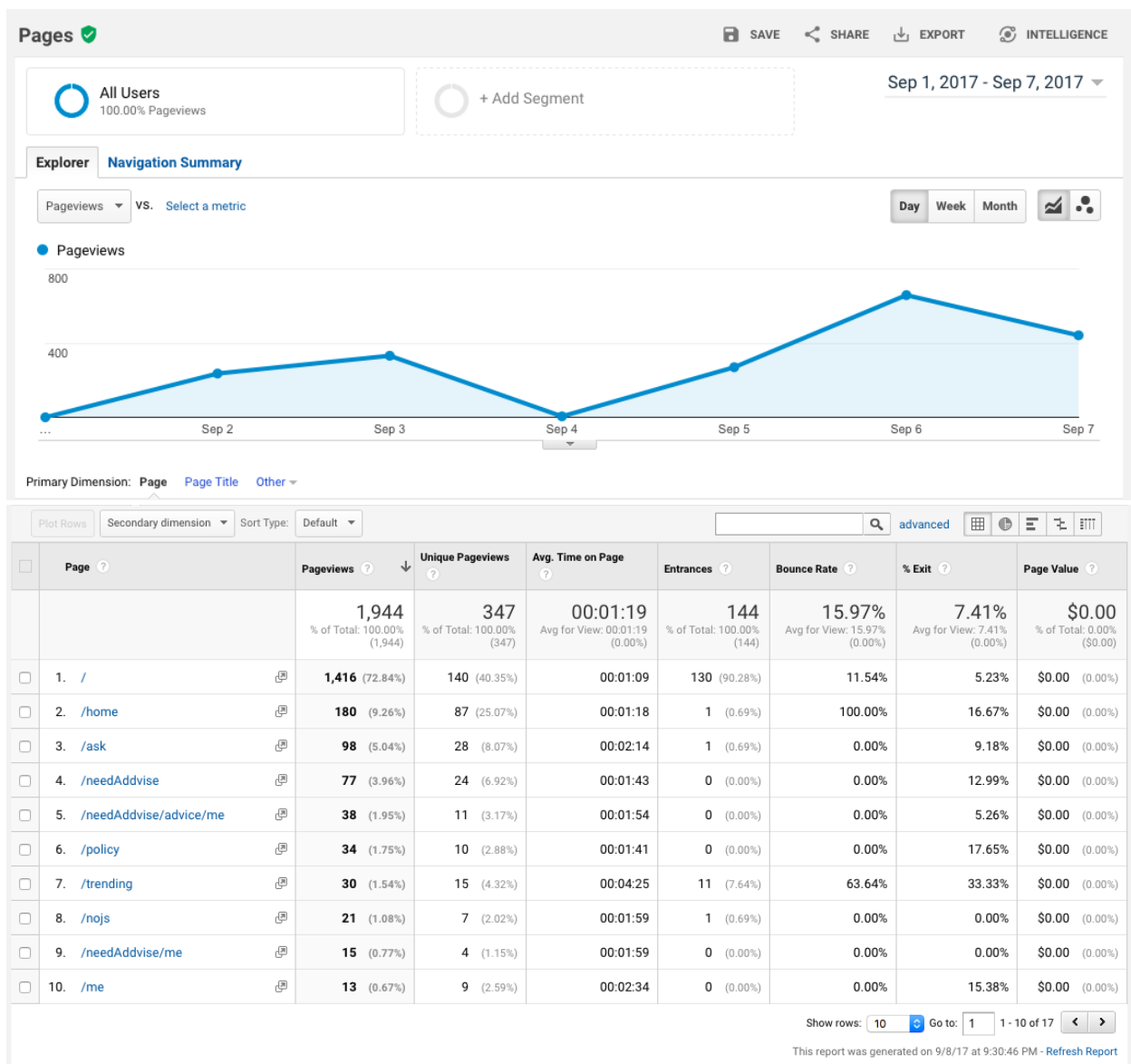
We placed our Like button below our logo. It follows the symmetrical concept of centering the button and this works well in both web and mobile version. In addition, we have to reiterate that the fundamental concept of our application is to post. Users will be able to see the like button the first thing when they visit our website.



Milestone 10: Explain how you handle a user's data when he removes your application and implement it. 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.

Our plan was to use Webhooks to remove a user as a 'member' and modify their permissions when a user removes our application. However, due to difficulties faced, we were unable to successfully integrate Webhooks into our application.

Milestone 11: Embed Google Analytics on all your pages and give us a screenshot of the report. Make sure the different page views are being tracked!

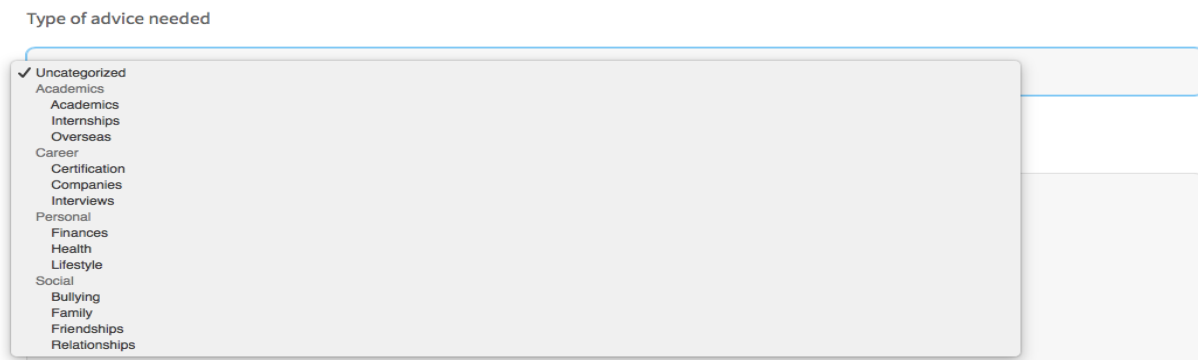


Screenshot above shows the analytics for all our different page views.

Milestone 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

1. **More input control provided to users requesting for advice. Dropdown has a few broad categories with further sub-categories to make it easier for users to select the appropriate category their request falls into.**

Ask for Advice



The screenshot shows a web form titled 'Ask for Advice'. Below the title is a label 'Type of advice needed' followed by a dropdown menu. The dropdown is open, displaying a list of categories. The first category, 'Uncategorized', is selected and marked with a checkmark. The list includes: Uncategorized, Academics, Internships, Overseas, Career, Certification, Companies, Interviews, Personal, Finances, Health, Lifestyle, Social, Bullying, Family, Friendships, and Relationships.

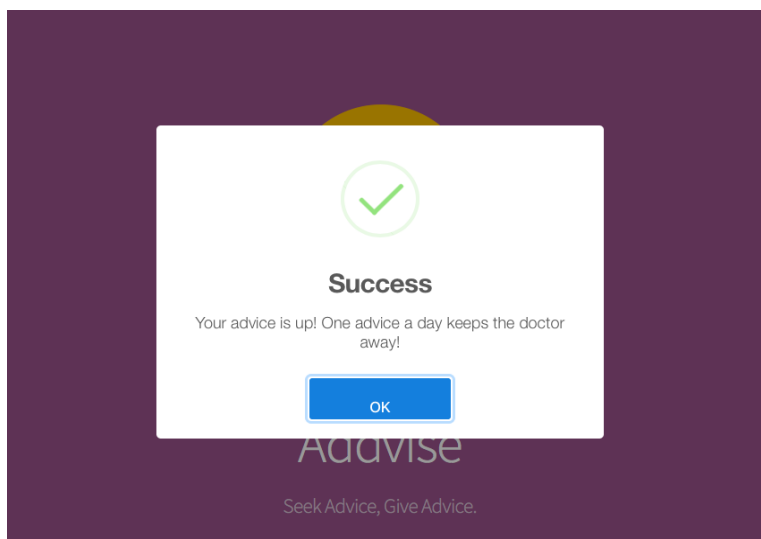
2. Hamburger navigation bar

Video showing responsive hamburger navigation bar:

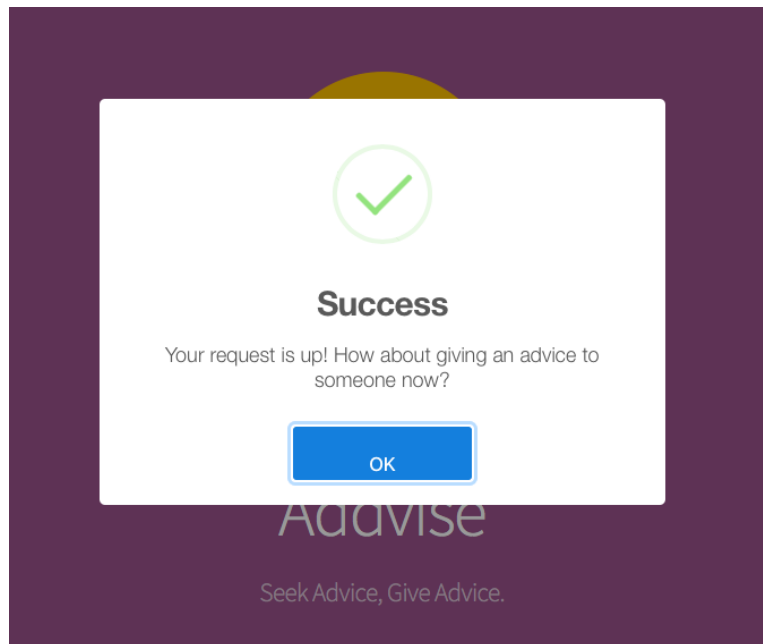
<https://drive.google.com/open?id=0BxGjlsC7FaauRkF4dXFFdFIfUEk>

3. Provide popup alerts to guide users on their actions:

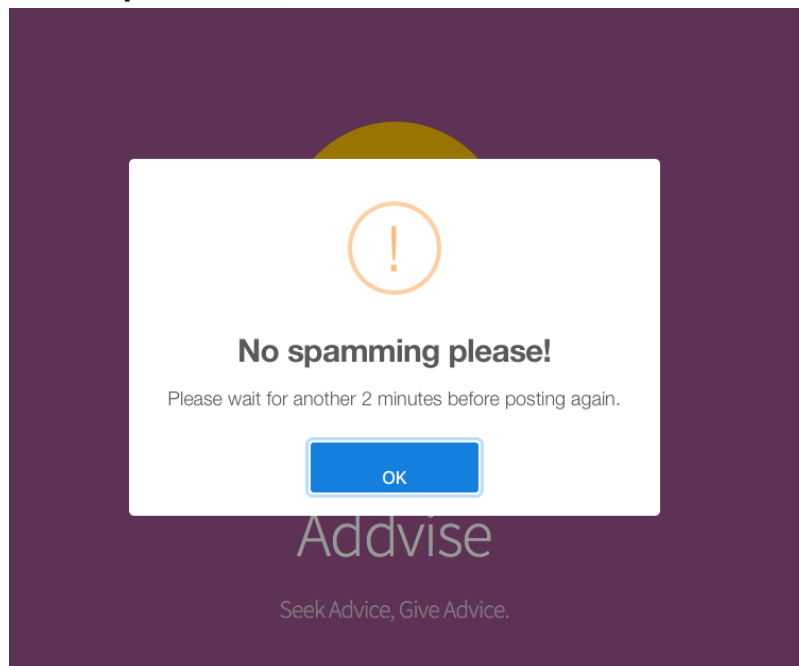
1. For people giving advice:



2. For people seeking for advice:

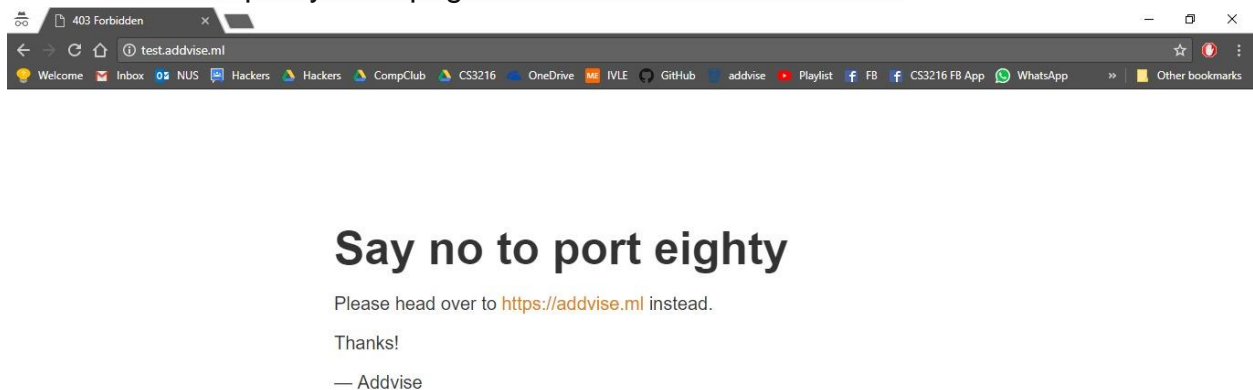


2. For spammers:



Milestone 13 alternative: In replacement of this milestone, we have bonus features:

1. **Redirecting all http requests to https.** If redirection fails somehow, we have created a pretty error page to handle manual redirection.



2. **Mobile Friendly.** Our hamburger navigation bar is auto-activated at a reduced screen size. Our text, screen size also auto resize to fit mobile devices.
3. **Personalized Posts data.** We created a “My requests” and “Advice I gave” page to give a more personalized page for the users. User can keep track of their posts without scrolling through hundreds of posts.

Milestone 14: What is the best technique to stop CSRF, and why? What is the set of special characters that needs to be escaped in order to prevent XSS? For each of the above vulnerabilities (SQLi, XSS, CSRF), explain what preventive measures you have taken in your application to tackle these issues.

Laravel’s Eloquent ORM uses PDO parameter binding to avoid SQL injection. Parameter binding ensures that malicious users can’t pass in query data which could modify the query’s intent. Without PDO parameter, a SQL query would look like this: `SELECT * FROM users WHERE id = 'jane' or 1=1`. Hence, a malicious user can input ‘jane’; drop table users; which would result in the following query: `SELECT * FROM users WHERE id = 'jane'; drop table users;` causing the whole database table user to be deleted. However, with PDO parameter, the resulting query will be: `SELECT * FROM users WHERE id = 'jane or 1=1'`, preventing any possible SQL injection attacks by escaping the quotes.

CSRF tokens are used to ensure that attackers cannot initiate a request impersonating the user. This is done by generating a token that must be passed along with the form contents. This token will then be compared with a value additionally saved to the user session. If it matches, the request is deemed valid, otherwise it is deemed invalid. Since we use the laravelcollective package to construct our forms, the CSRF token is automatically added to the forms.

Laravel's `{!!}` syntax will automatically escape any HTML entities passed along via a view variable, preventing XSS attacks. Hence, any scripts passed in through the form fields will not be executed, as the `<script>` tag will be replaced with `<script>`;

We also redirect all HTTP requests to HTTPS to ensure a more secure connection.

OPTIONAL

Milestone 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)

Animation 1: The logo

Link to view animation:

<https://drive.google.com/open?id=0BxGjlsC7FaauNWhMMEhxbW5kVHc>

Animation 2: Hamburger Icon

Link to view animation:

<https://drive.google.com/open?id=0BxGjlsC7FaauRkF4dXFFdFifUEk>

Animation 3: Pop-up alerts:

Link to view animation:

<https://drive.google.com/open?id=0BxGjlsC7FaauVDgyRGN3M19KUEk>