Le Xuan Manh (A0126501W)

Piyush Varanjani (A0105178H)

You Jing (A0133895U)

Zhang-yi-jiang (A0135817B)

CS3216 Milestones

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

We chose to create the app both as a standalone web application and a Facebook canvas app. This is because our game is a social game, which makes it suitable to be hosted through Facebook canvas alongside other games on the platform. We also make our game available outside of the Facebook canvas platform because the Facebook canvas platform has a much narrower reach than if the app was available outside of the platform.

Milestone 2: **Explain your choice of toolset 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.**

- Hosting: nginx on Digital Ocean Ubuntu instance
- Server-side stack: Django web framework on Python with gunicorn
- Client-side: Bootstrap framework, jQuery library, no front-end JS framework
- Asset pipeline: Gulp with Sass, Uglify and Browser-sync

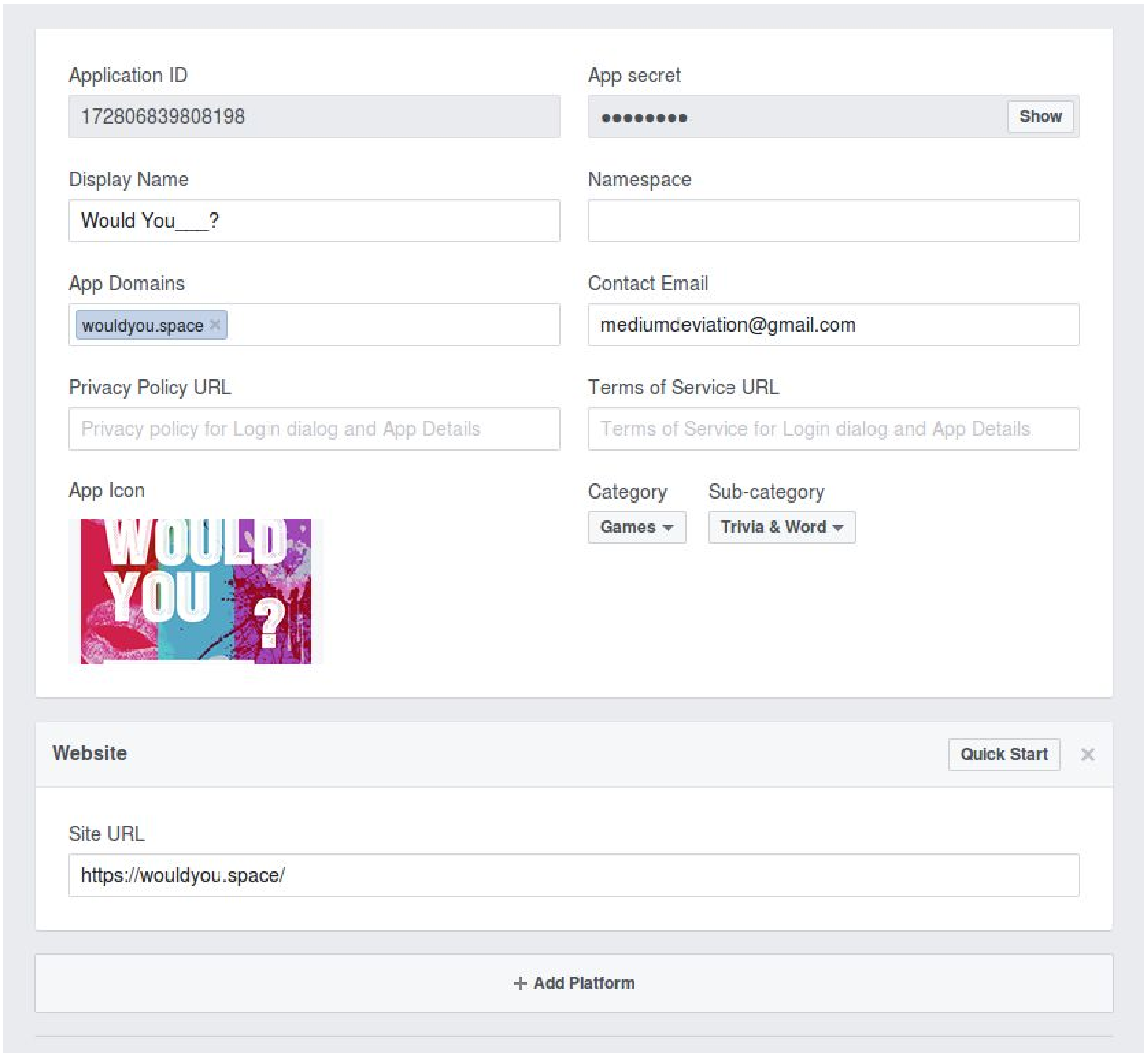
Hosting: this stack was chosen because it comes with sensible defaults out of the box and can be configured easily. Nginx is especially easy to configure compared to Apache, while Ubuntu achieves a good balance between stability and having newer versions of packages, as well as having wide community support, compared to other popular distros like Debian, CentOS and Fedora.

Server-side stack: The main alternative that was considered was Laravel on PHP. The table below highlights the main advantages of each choice. While one of the developers have extensive experience with Laravel, we chose Django, although neither developers are familiar with it, because we hope this would enable both developer to contribute more equally. This is based on the assumption that Laravel would be a more difficult framework to learn as the other developer has no experience with PHP.

Django is also good for rapid prototyping because the framework can automatically generate database migrations and comes with the CMS admin interface. Laravel does not have these, which means a significant amount of time have to be spent writing boilerplate database migration code and admin pages.

Django + Python	Laravel + PHP
Django Admin - CMS comes with framework Django migrations - automatically generated migration scripts Python - a much more pleasant language to work with than PHP Abstraction - higher level of abstraction than Laravel	Easy to deploy - both framework and non-framework assets runs through the same server, no need to keep long running process Familiarity - One of the backend developers have extensive experience with Laravel

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!**


Application ID
172806839808198

App secret
●●●●●●●●

Display Name
Would You___?

Namespace

App Domains
wouldyou.space

Contact Email
mediumdeviation@gmail.com

Privacy Policy URL
Privacy policy for Login dialog and App Details

Terms of Service URL
Terms of Service for Login dialog and App Details

App Icon

Category: Games
Sub-category: Trivia & Word

Website

Site URL
https://wouldyou.space/

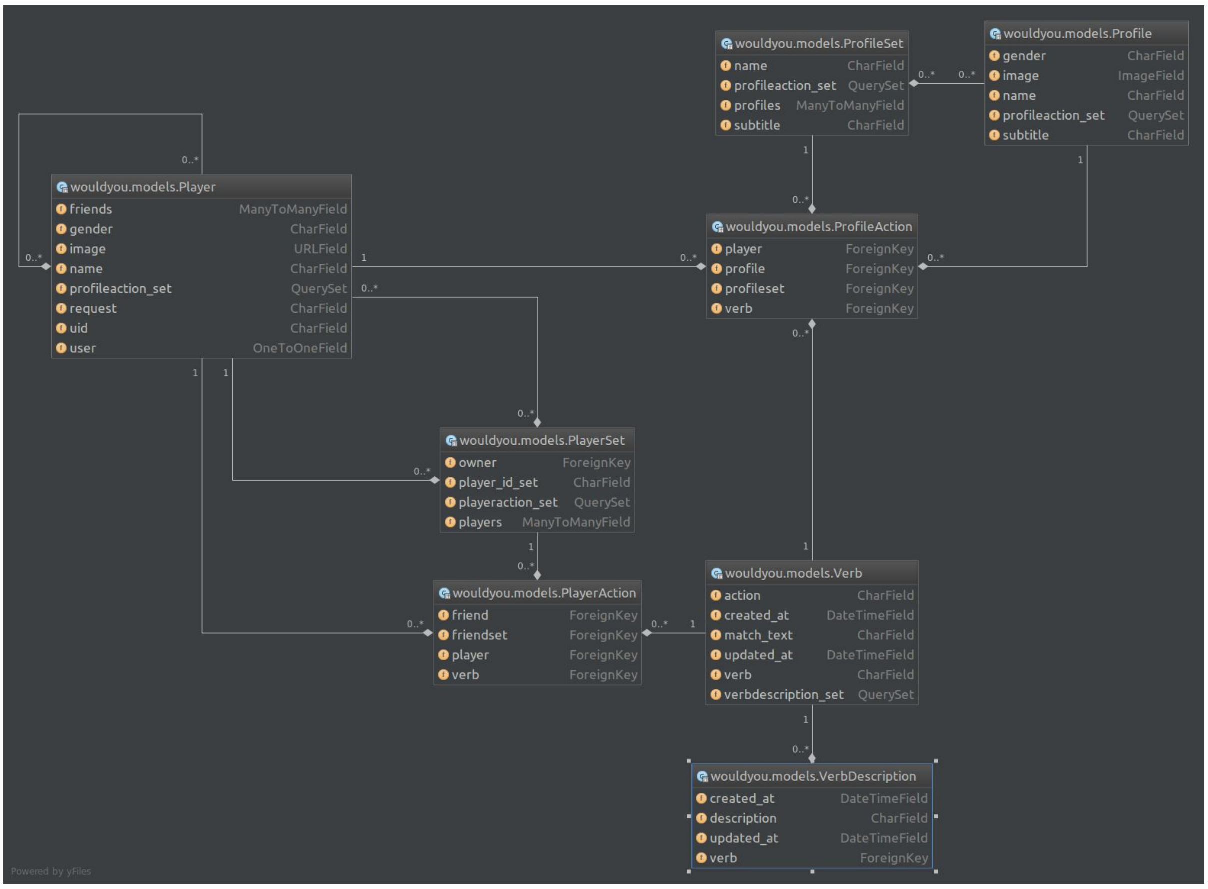
Logo:

Milestone 4: Integrate your application with Facebook. If you are developing a Facebook canvas 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.

Done! Please check here: https://wouldyou.space

Milestone 5: Draw the database schema of your application.



Full sized image can be found at http://imgur.com/eTvqqPC.png

Milestone 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.

All queries seen here are generated using Django ORM with a Postgres database.

SELECT "wouldyou_player"."uid" **FROM** "wouldyou_player" **INNER JOIN**"wouldyou_player_friends" **ON** ("wouldyou_player"."id" = "wouldyou_player_friends"."to_player_id") **WHERE** ("wouldyou_player_friends"."from_player_id" = 7 **AND NOT** ("wouldyou_player"."id" **IN** (**SELECT** V1."player_id" **AS** Col1 **FROM**"wouldyou_playerset_players" V1 **WHERE** V1."playerset_id" **IN** (**SELECT** U0."id" **FROM**"wouldyou_playerset" U0 **WHERE** U0."owner_id" = 7))))

This query is used to check if the current player has any friends he has never played a game with before. It is relatively complex as the schema was designed to allow the saving of partially played games (though it was decided in the end that the game will only proceed when either all three option has been selected or the player presses skip). That meant that the individual player choices was stored in a separate table from the potential choices the player could make, which added complexity to the database design.

```
SELECT "wouldyou_playerset"."player_id_set" FROM "wouldyou_playerset"
WHERE("wouldyou_playerset"."owner_id" = 7 AND "wouldyou_playerset"."player_id_set"
IN('1153166724726814,2096390553920391,662659760554647',
'1153166724726814,2096390553920391,956322294478826',
'10209976334031130,1153166724726814,956322294478826',
'1153166724726814,662659760554647,956322294478826',
'2096390553920391,662659760554647,956322294478826'))
```

This is the query used to look for duplicate player sets when generating new ones. A player set is a set of three players who are friends of the current user. To ensure each game is played with a unique set of players, we first grab a list of UID from all of a player's friends, then draw random sets of three and check if they already exist in the database. Since this query would be difficult to write against a fully normalized table (it would require three separate inner joins and aggregates), we decided to denormalize and have the model automatically store the three user IDs of the players in each player set in a column automatically when the player set is first created, and match the potential player set's player IDs against that to check for duplicates.

```
INSERT INTO "wouldyou_profileaction" ("created_at", "updated_at", "verb_id", "player_id", "profile_id",
"profileset_id") VALUES ('2016-09-02T13:22:05.191698+00:00'::timestamptz,
'2016-09-02T13:22:05.191780+00:00'::timestamptz, NULL, 7, NULL, 5)
RETURNING"wouldyou_profileaction"."id"
```

This is the insertion of a row into the profile action (player choice of kiss, marry, or kill against a celebrity) table indicating that he or she has 'skipped' the current set. When the schema was originally drawn up, this was not an anticipated action. Instead of adding another column or table to store this, we instead made the verb_id foreign key column nullable, with the null value indicating that the player has skipped the set. We felt that the faster development time was worth the slight tradeoff in data integrity.

Milestone 7: Show us some of your most interesting Facebook Graph queries. Explain what they are used for.

```
FB.api ('me/' + action, 'post', {
  celebrity: celebrity,
  access_token: userAccessToken,
})
```

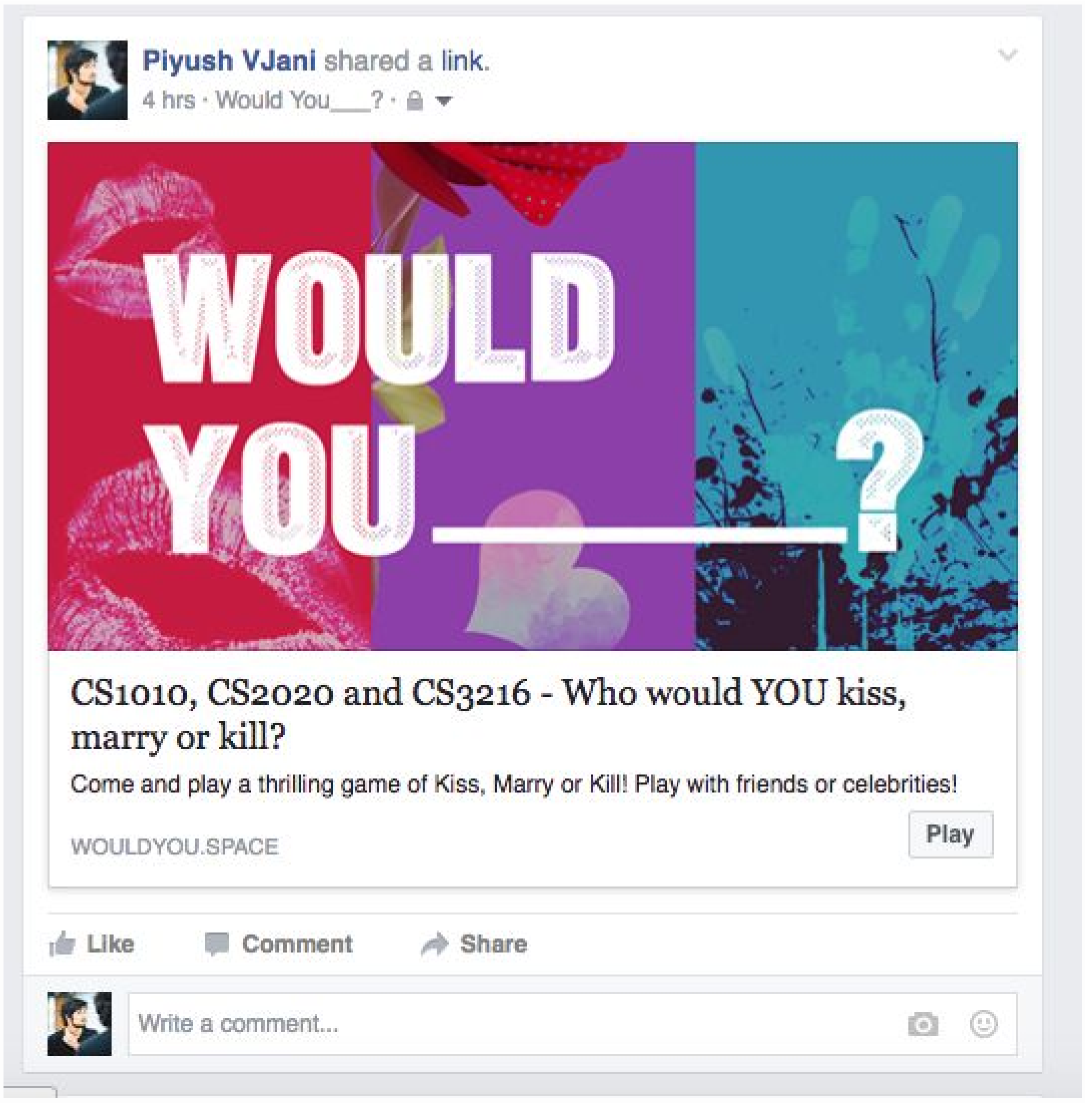
This is an open graph query made on the front-end to post custom user actions. The user's access token is passed from the backend to the frontend because the user authentication flow is done on the backend. Both the action and celebrity are variables pulled from the 'Post to Facebook' button itself, with action being our custom action (want to kill, marry or kiss), and celebrity as an URL pointing to a page containing only open graph metadata. A normal user to the same page would get a 404 error as we do not want to confuse them by showing them a 404 page, but the Facebook bot, identified by its user agent, is provided with this special page. The same technique is used to allow the sharing of sets of celebrities that players can play with, even these pages normally require logging in. The page itself only contains simple metadata, so no sensitive information is leaked.

```
FB.ui ({
  method: 'apprequests',
  message: message,
})
```

We use the Facebook SDK to let users invite their friends to play our game. Since the Facebook Graph API no longer supports returning all users, we abuse the invite system since after the invites are sent out, we get the Facebook User ID of all invitees, and using this we can query the Graph API's user endpoint for the data necessary for the game to continue.

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!

Users can share 'Profile Sets' (game instance) with their friends on Facebook. This intrinsic behavior is triggered when a user is addicted to the app, and wants to showcase it to their friends. Our app also pushed users to share as it provides users more value if more of their friends join the app, facilitating network effects.


Piyush VJani shared a link.

4 hrs · Would You___?

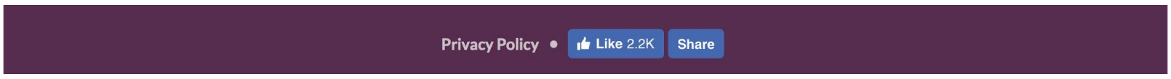
CS1010, CS2020 and CS3216 - Who would YOU kiss, marry or kill?

Come and play a thrilling game of Kiss, Marry or Kill! Play with friends or celebrities!

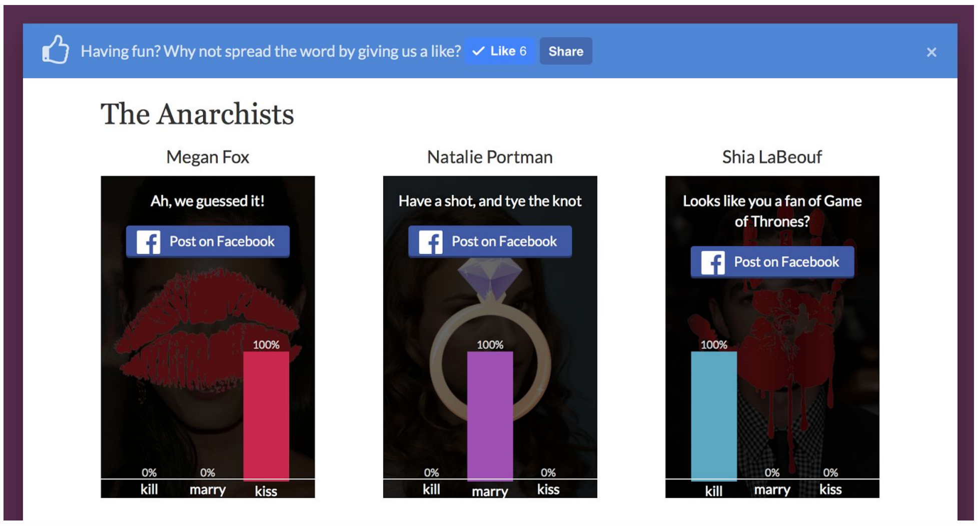
WOULDYOU.SPACE

Milestone 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 have positioned the Like button at two places. One is at the bottom of the page. That is a conventional place to put Like button that users can find easily and it does not get in the way of the user experience.



After the users have played five rounds, as shown below an alert will show up at the top of the game area to ask whether the users want to like our app. This is an obvious place for the users to see and it is not too annoying since we do not force users to perform the action and they have the freedom to choose whether to ignore it or continue playing.



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.

1. Handling the user’s data:

When the user removes our application from his Facebook settings, we will remove all data involved with this user(their data from Facebook (profile, name, user_id …), their actions that they made when using our application, and all actions from their friends that involved them.

2. It is necessary and the most logical:

First, we already put the Removal Information into privacy page to tell users that if they remove our application, then all their data will be removed. So users know what will happen if they remove our application from their Facebook settings.

Second, our application is a game and either users play or not play it. There is no deactivated mode (like Facebook/Viber….), then when user removed the application, we assumed that the user does not want to involve to our application. That’s why we decided to delete all user’s data and data that involved this user.

Third, our application using user’s friends to create games, so to avoid his friends make games involve him (when he is not using our application anymore), delete the these data is necessary and logical,and also easier for us to manage games between users.

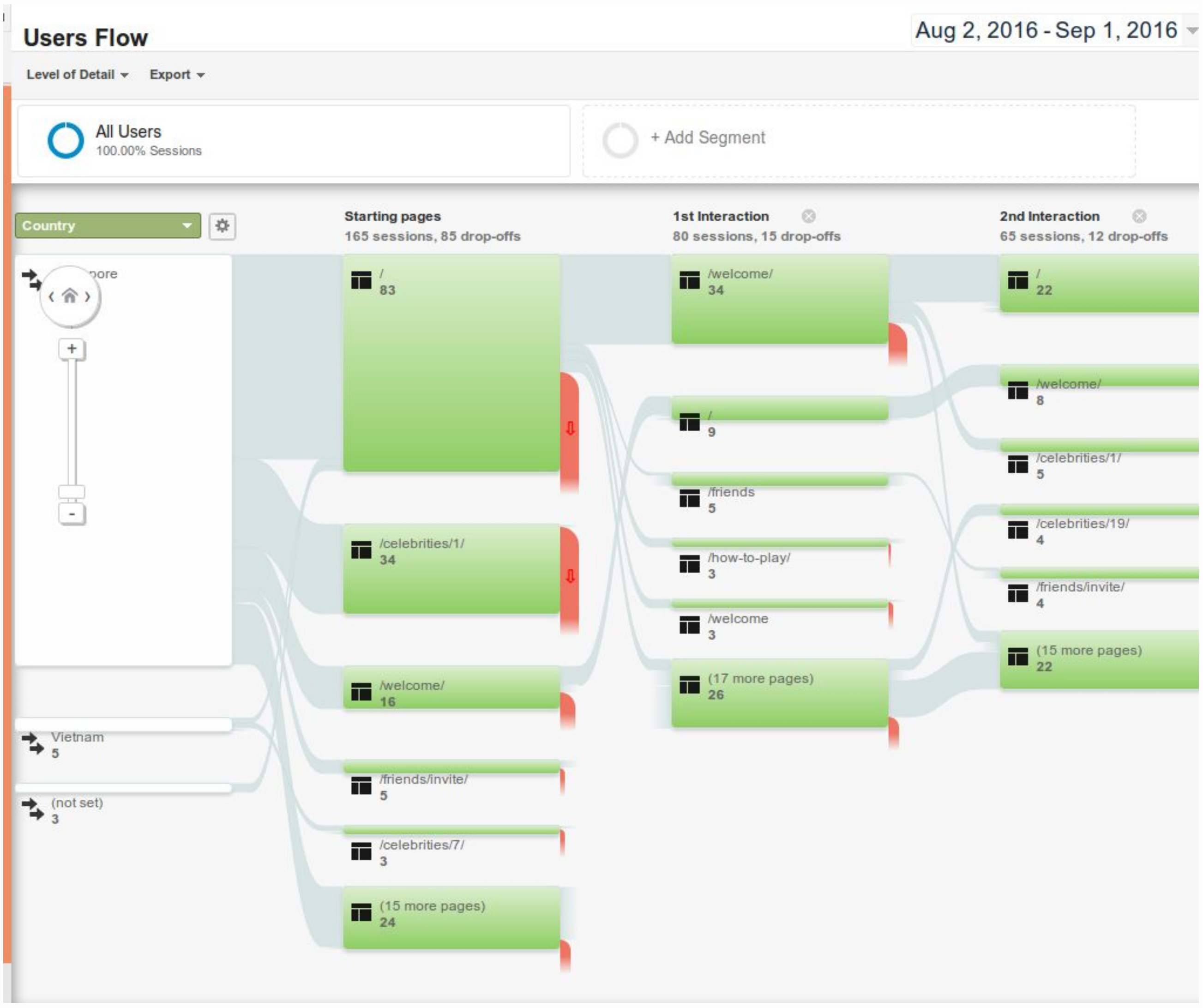
Fourth, we use Django+python, and Django does cascading deletes on foreign keys, so it is easy for us to delete user's data and data that involved this user.

3. **We do not violate Facebook's terms and conditions because:**

- When users use our application, they gave us right permissions to access their data on Facebook.
- When users remove our application, we delete all users' data from our application (not keep anything involved users). Then it is obvious to say that we do not violate Facebook's term and conditions.

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!**

We have implemented Google Analytics on all our pages. As it can be observed from the screenshot that the user is being tracked throughout all the different pages of the website during their session.

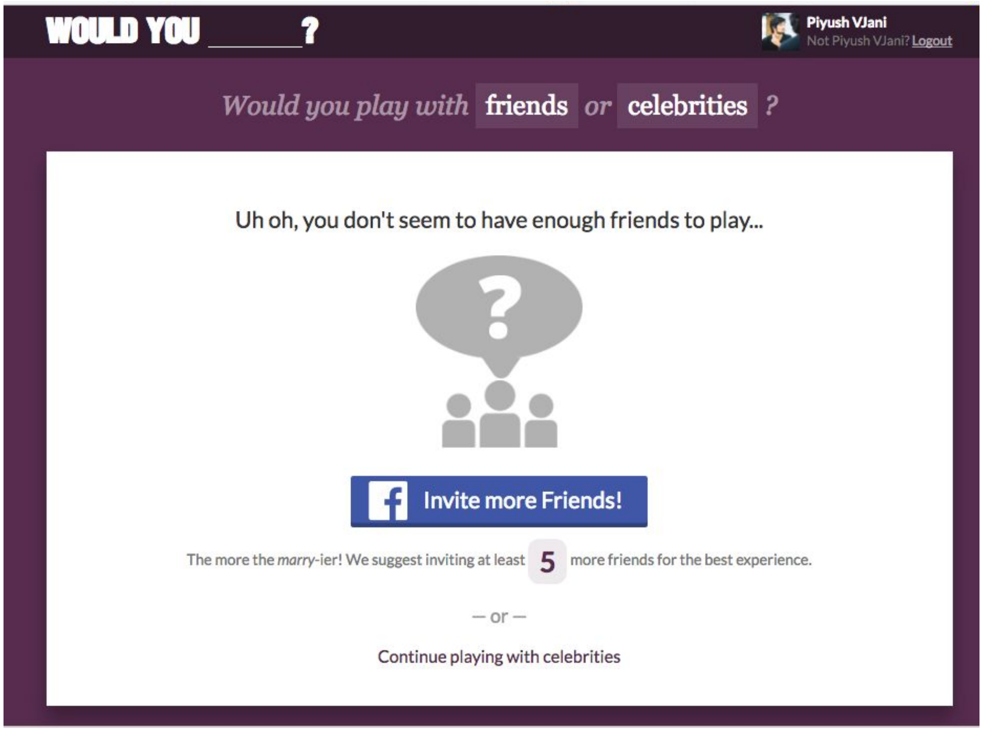


<span>Milestone 12:</span> Describe 3 user interactions in your application and show us that you have thought through those interactions. You can even record gifs to demonstrate that interaction! It would be great if you could also describe other alternatives that you decided to discard, if any.

At the heart of our application lies the surprise element of kiss/marry/kill.

Experience 1: Providing value from Day 1.

If the user doesn't have adequate number of friends, asking them to invite more while giving them an option to play with celebrities. This step makes sure that user gets value from Day 1, even if no friend has signed up on the app. So instead of frustrating the user, we delight them!



Uh oh, you don't seem to have enough friends to play...

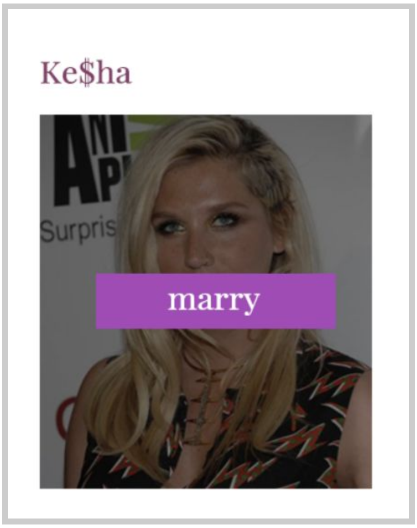
The more the *marry*-ier! We suggest inviting at least 5 more friends for the best experience.

— or —

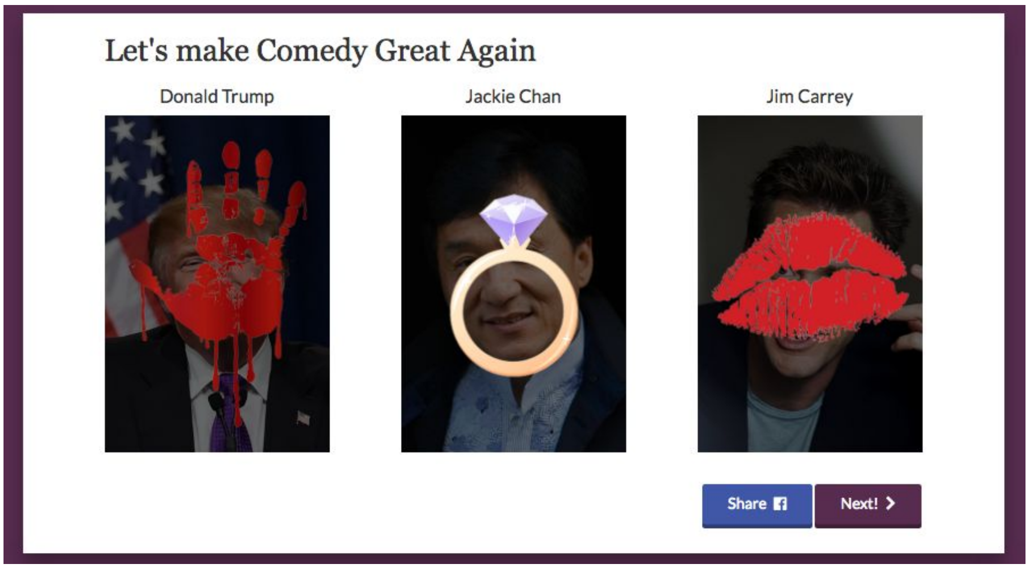
Continue playing with celebrities

Experience 2: Little delights!

Once a user selects any of the three options (Kiss/Marry/Kill), we used to show them their respective choices with the same phrase, as it is.



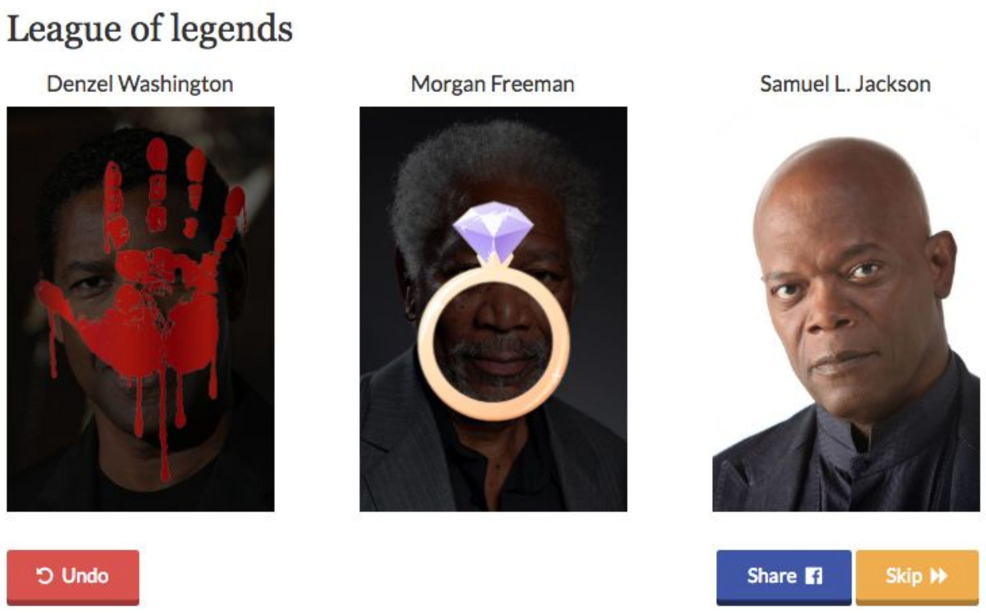
It was boring! So in the newer version, we wanted it to be more attractive, and so we got this:



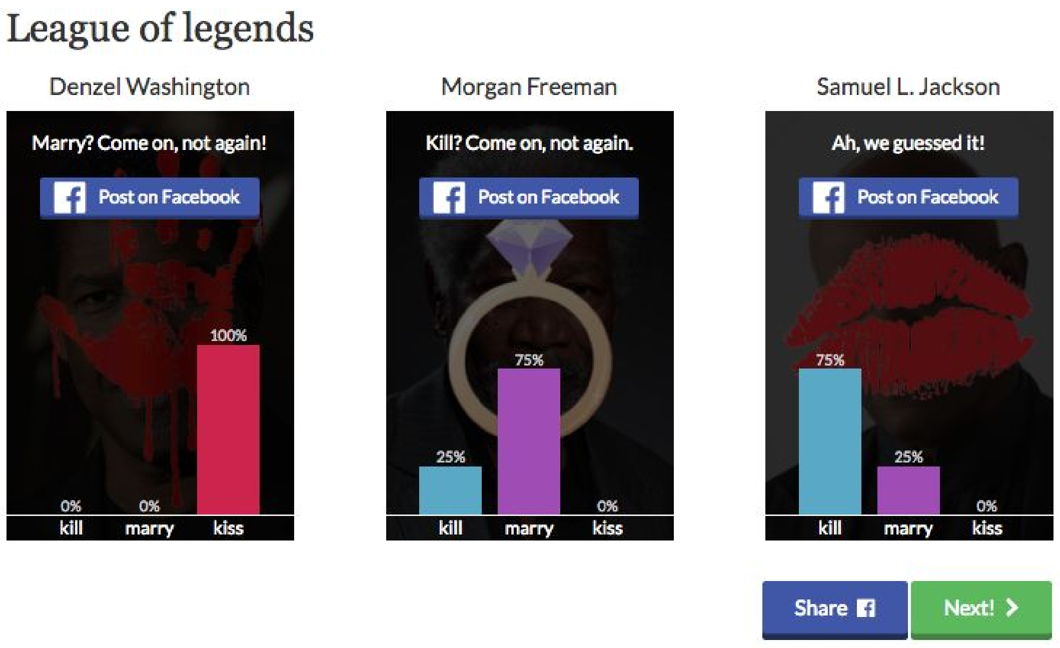
New version with icons as the chosen action

Besides that we wanted to make our users feel a part of bigger community and to indicate it has been played before, we added statistics to show what others think.

We had intense discussions on the desired behavior from the users so we made few more changes to the app: Once the user sees a particular set of choices, they will see a skip button on bottom-right but only once they choose all three choices it turns into a next button.



Unless all the three choices are made, users can’t select Skip. They can undo at this point.



Once the three choices are made, undo button disappears, and there is Next button instead of Skip button.

Experience 3: Confess your desire, secretly, and know theirs too.

By just playing this game, you can find out what you always wanted to ask your crush,

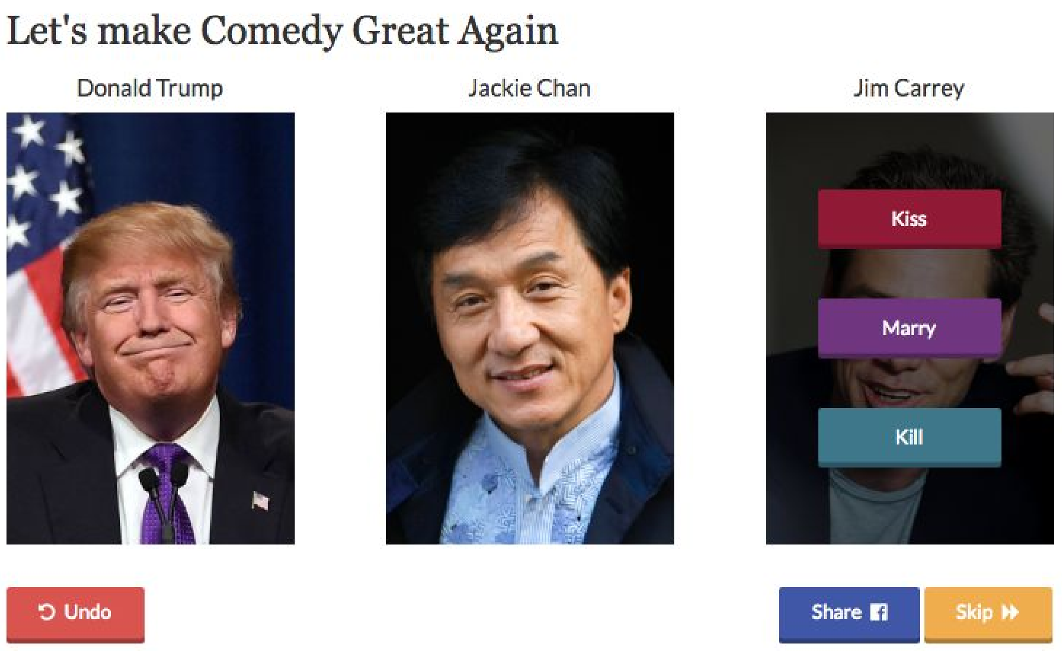
"*Would you?* kiss/marry/kill me?"

1. Once you log in, invite the friend who you want to kiss/marry/kill
2. Simply select your choice kiss/marry/kill
3. See the result of what they chose for you!

But here is the fun part, you don't get to see their choices if they selected a different answer than what you did for them. So if you both chose different options for each other, you will get a "That's not what (s)he said!" message but if you both chose the same option, suppose kiss, it will show "Bingo! Kiss Kiss Bang Bang."

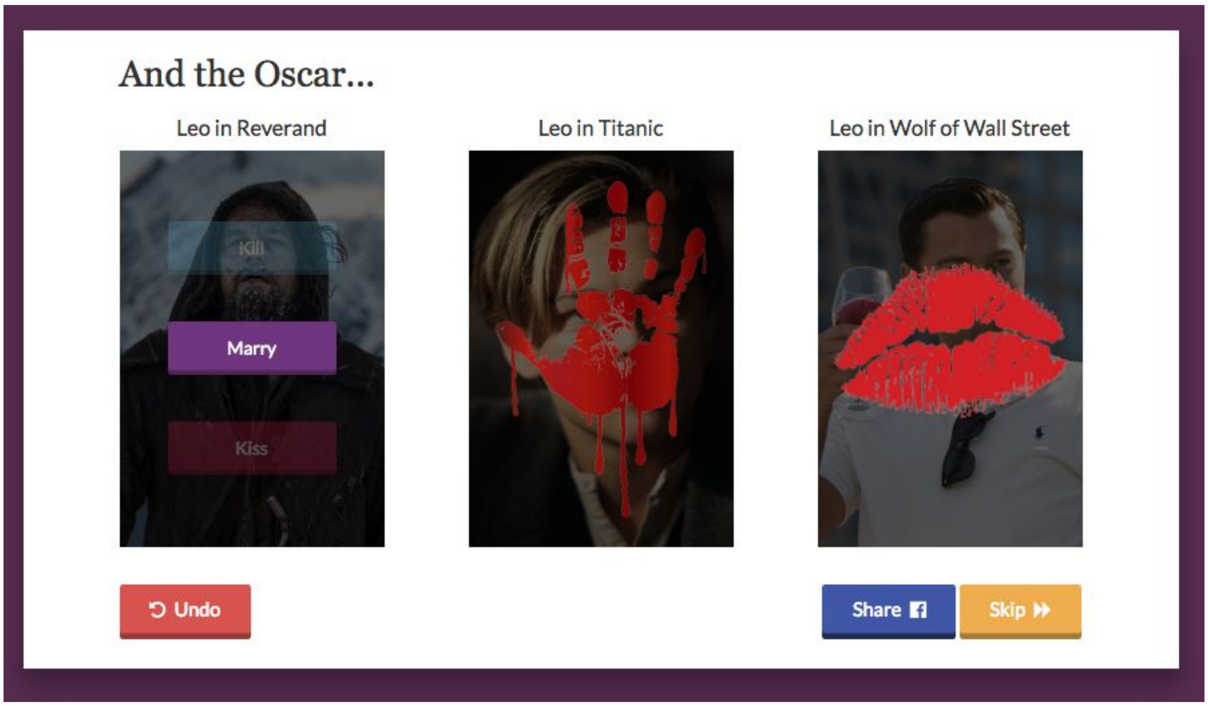
This way we ensure the user gets only the best response if it is reciprocated, and if not then maybe that's now what (s)he said!

Bonus: Carefully selecting the profile sets. Most of the times, other contemporary games (Chuck Marry Kill, etc.), have profiles of hot actors and actress, we spice it up a little bit, "Let's Make Comedy Great Again."

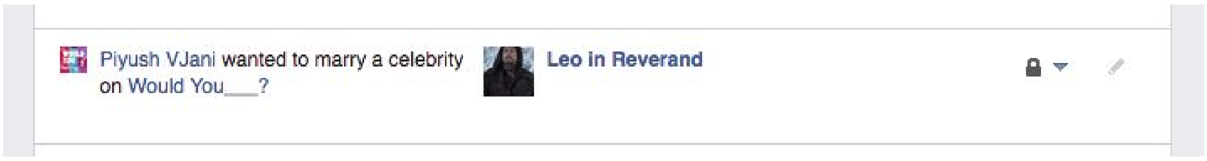


Milestone 13: Implement at least one Story involving an Action and Object in your application. Describe why you think it will create an engaging experience for the users of your application.

Facebook shows the Story in user's Activity Log. We implemented it to show user's choices during the game. So if I get a



And if I choose to ‘Marry’ Leo in Revered option, I can see the story on my activity log.


Piyush VJani wanted to marry a celebrity on Would You___? Leo in Reverand

<span>Milestone 14:</span> 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.

CSRF is best stopped using a unique session token, also known as a nonce, for each user that is embedded into every sensitive request. For forms this is usually done via a hidden input element.

XSS can be stopped by escaping the characters used to define HTML tags and replacing them with HTML entities. This means <, >, =, “ and ‘ should be replaced with &lt;, &gt;, and so on. Simply stripping out all tags is insufficient since in some cases even invalid tags will be parsed and rendered by browsers. In addition, Content Security Policies can also be used to further tighten security by disallowing inline JavaScript and prevent the loading of resources from untrusted domains.

For SQL injection, we use Django ORM for data persistence and interaction with the database. The ORM automatically escapes all user inputs that will be stored in the database, and uses parameterized queries, thus ensuring the application is not vulnerable to SQL injection.

For XSS, Django template automatically escapes all output by default. We also take care to always pass user input through functions like format_html when writing template tags and widget classes, which ensures that all parameters are escaped.

For CSRF, Django's CSRF middleware automatically rejects any non-safe (non GET, HEAD and OPTIONS and TRACE) requests that do not have the CSRF token defined. However, we have partially disabled this filter because Facebook's Canvas platform uses POST to fetch the pages instead of GET and display them inside an iframe. More specifically, using a middleware we disable the x-deny-iframe and CSRF verification when the 'signed_request' field is present in the request, then let the social authentication middleware that is further downstream to reject any requests with invalid 'signed_request' values.

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.

Landing page:

When users first reach the application, they will be directed to the interactive landing page. There are three color bars in the background (i.e. red, purple and blue), each representing the action of kiss/marry/kill respectively. When a user hover over one of the bars, it gets highlighted and zoomed in, and the corresponding text shows up as a spinner to fill in the blank. This creates the experience for the users to explore, raises their curiosity and reminds them of the three most important terms in the game.

Game page:

This page provides a game-like experience. When users hover over a profile, there will be a fade-in animation for the options to make the user experience smoother. Upon selecting on one, the corresponding icon shows up(a lip print for kiss, a ring for marry and blood for kill) with animations. And after all three choices are made, the results will show up one by one

Milestone 16: Describe how Ajax is utilised in your application, and how it improves the user experience.

Ajax is used to handle the invitation of new players into the game. After the invitations have been sent out, Facebook SDK provides the Facebook user ID of each invited player, which we send back to the server using Ajax to the server to create new player instances and check that we now have enough players to generate additional sets of friends to play with. Using Ajax to send the data back to the server improves the user experience as it provides immediate feedback if the player did not invite enough friends, instead of forcing him or her to sit through repeated page reloads.