TIKKACHU
NINTINDER

Overview

Our application takes the powerful matchmaking properties of the dating app *Tinder* and targets a whole new user base: gamers. Nintinder helps bring gamers together and make the most out of their gaming experience. Users will set up a profile that captures their primary interests and what they are looking for. Using this information, Nintinder will bring forward the best users who fit the description. Ultimately, users will be able to decide if they are interested in connecting with those suggested users or not. Nintinder will also feature a social feed where players can post status updates, share game results, post about gaming events, and much more. For all of your gaming needs, Nintinder is the one-stop solution for the optimal gaming experience.

Team Members

Johan Pereira, Raymond Liu, Daniel Zhang, Kush Patel, Oliver Pang, and Bhishma Pant

Github Repository

https://github.com/Philosobyte/nintinder

Design Overview

Data model: The two main tables are User and Game, where Game contains the static information like publisher and platform. Most other tables in our database, in general, reference these two tables in some way. The Interest table keeps track of which games each User is associated with. The Friend table keeps track of connections between Users, including whether or not certain users have “blacklisted” other Users from being connected. Achievement defines the possible achievements designated to each game, and EarnedAchievement associates users with the Achievements they have obtained. Event is a table of scheduled competitions, LAN parties, etc. and Participant holds users that *participate* in those events.
Important URL routes: Our main page, once called “feed” is now the “/nintinder/” home page. From the home page, we can reach “/nintinder/profile” (profile.html), then the sidebar button opens up to the Home page, Friends “/nintinder/matches” (matches.html), and Settings “/nintinder/settings” (settings.html)
UI views: Overall, each view is broken into three sections: the header bar, the sidebar and the main-content. The header bar and sidebar HTML is all contained in base_template.html. On the other hand, main-content contains the individual pages (e.g. profiles.html, achievements.html, etc.) with the dynamically generated information.
***Disclaimer:** We purposely made the site choose one random user from the database every time we reload the page because we want to see if everything was working as intended and show dynamic retrieval from the database constantly. This also means that every associated attribute/friends/achievements/etc will be linked to the specific User. Naturally, this will be removed once authentication is added to the website in Project 3.*

Problems/Successes

Successes: One of our biggest success in our website was that we were able to dynamically generate HTML <div> tags using python code and in addition to that, we also learn how to dynamically generate each user profile picture in the matches.html inside the HTML file.
Problems: One of the implementation issues was not being able to view/modify our mock data. After some research, we realized that our error was directly modifying the database file using SQL queries and a visual database browser. This problem was solved by dumping the current database into a temp file and then reloading the entire database again using django.

Part 4: Individual Write Up

(18%) Johan Pereira

For this project, I created the data models in models.py. This involved working out a relational schema with the group (with really good suggestions from Raymond --who also made the diagram). From there, I worked on models.py (and the creation/running of migrations with each major step). At the same time, I also created the admin classes for the models as well, allowing data entry/editing to be possible through the Django admin site. Speaking of the admin page, I created the superuser as well. Frankly speaking, I had fun doing this (because I really did do what I did for fun over break and I enjoyed it) Other than that, I made sure to help out elsewhere where I could :-)

(17%) Raymond Liu

After Johan created the models, we came up with a few different ideas/pivots to the model (as well as outlined a few fixes that needed to be made). For those I tweaked the models per our group discussions. This involved adding __str__ methods to the models, and fixing two slight mistakes with Friend statuses and Event. I finished modifications to the logo which I started in Project 1, and I reprocessed all our profile images in Lightroom so they are the same aspect ratio and resolution, and so their names correspond to usernames in our mock data. I created the data model diagram.

(15%) Daniel Zhang

My main job for project 2 was to finalize our base_template.html. This include adding the our logo to the html file, implementing the sidebar feature (e.g scroll locking the page when the sidebar is out, making the sidebar overlay the page) and making the template responsive. In addition to that I helped Gaurab move over some of mock UI HTML to the django application. Overall, I am really proud of the team and the work we put in to our website!

(15%) Kush Patel

For this project, my contributions involved populating the database with mock data to be used in the app. Specifically, I helped populate data for the Users, Friends, Events, and Participants. At first, this was done by directly modifying the database using SQL queries, however, this caused errors in actually being able to use the data. As a result, the data was then added creating the appropriate objects through shell commands, a method which Oliver had found.

(15%) Oliver Pang

I helped populate the database with mock data for Game, Achievements, EarnedAchievement, and Interest, making sure they all linked to their proper data columns and are usable for mock data. I also researched a way to script in tuples with django orm when direct sql inserts caused errors. I also linked the matches page with the database, for both friends and friend's games. Overall great work by the team. Bit rushed at the end but I think we did good.

(20%) Bhishma (Gaurab) Pant

I got the wonderful honor of linking our database to our view and front-end with Daniel, creating our paths in urls.py, and creating some methods in views.py to dynamically create most of the data displayed on our html pages. I took all of the html pages we had created from Project 1 and got it to work with the server/moving the static files over. Django syntax was difficult at first, but definitely manageable with time. The most difficult task, I feel, was dynamically for looping (in the Achievements.html) through the Achievement and EarnedAchievement objects and seeing which one's the User has actually completed. It was definitely rewarding to see everyone's hard work all come through in the end to create a working product!