* Download the repo
* “Good Vibes” is the Unity project folder
* Most of the main networking scripts and functionality are under PlayerController.cs
* Functionality is split into regions, such as a region for sending vibes, receiving vibes, reporting vibes, etc. and they generally follow the structure of private instance variables functions to call networked code coroutines, forms to send specific data back and forth, and receiving functions and coroutines to display data to the user
* The script for connecting the authoritative server and network client is NetworkHelper.cs
* Most of the UI is controlled through MainMenuController.cs
* An android build is made for the client (although windows builds and really any other platform is viable as a client)
* A Linux build with server mode checked (headless) is made to run on the server, which in this case held our database as well
* Most of the back-end is done via PHP scripts on a digital ocean droplet that won’t be available to future developers, so the scripts have been taken from the server
* PHPMyAdmin was used to create and manage the database
* There’s an old create database script on the github, CreateGoodVibesDB.sql and ERD to match it, but the structure of the database changed during development, so screenshots of the new structure are on the repo in order for it to more easily be recreated on a future developer’s server
* Without a direct way to move builds from a local computer to the server, we relied on zipping the build files, uploading it to google drive, copying the share link, opening the link, copying the link from the “download” button, and using wget url on the linux server’s terminal to download the zip archive, often using tinyurl.com to simplify the wget url command
* After unzipping, chmod +x (name of build file) has to be done in order for the build to be executable
* Running this command: ./ buildexecutable.itsextension -logfile log.txt would create a log file that when opened, can be refreshed to get a constant log
* Using our provided PHP files, if you want to get error reporting, such as bad syntax or incorrect data input, error log files would show for us under an Ubuntu linux structure such as var → log → apache2→ error.log
* The server build connects to localhost, aka the php files that interacted with the database, if the database was on the same server as the linux server build, which was the case for us. Localhost php files on the server need to be located under var → www → html
* UNet was used to connect the android client to the linux server build. The NetworkHelper.cs script would automatically host a match if the script was running on a server, and if running on a client, would automatically join the match, via the same password used in the network string.
* We didn’t run into CCU bottlenecks. This is concurrent users for a match, which for a free version of Unity is 20 users, however in our tests, Unity was not counting the number of clients connected to the match for whatever reason