Raden Mathieu Almaden 101104851
Dave Mckenney
COMP 2406 WINTER 2020
Assignment 5 (Final)

README.

How to setup the system:
1. Make sure to have mongodb installed and some way to look at the mongo database. An example is the MongoDB Compass app which you can connect to the local user once mongod is up and running..
2. Afterwards, make sure to run mongod in the background with a database setup in the folder of this project.
3. Next open up a command prompt within the directory containing mainServer.js
4. Before starting the actual server, you'll need to install some modules. In the command prompt type "npm install express mongoose express-session mongodb crypto pug"
5. After all this is done, you can initialize the database with the command "node database-initializerEDIT.js"
6. Next, you can start the server by typing the command "node mainServer.js"
7. Afterwards, you can go on your browser and head to localhost:3000 to open up the website.
8. At this point, you can start testing!

NOTE:
Something to immediately take note of so you don't waste you're time. There will Unfortunately be no real-time updates, I wasn't able to fully implement that.
Also, you can only trade 1 for 1 cards. Again, I did not have the time to complete being able to trade no cards, or subsets of cards.
(If you also try adding same usernames, there will be a mongo error message on the console as I initialized the database to only have unique usernames. These errors will just be text and warnings, they don't affect the program)

TESTING:

When you initially load the page, you'll be able to register or login. You will need to first register 3 users, with different usernames.
You can test trying to register with the same username, it will not let you.
Afterwards, you can log in on either or on 2 on separate browsers.

There is a logout option if you decide you'd also like to log out near the top of the page.

Once logged on, you'll be sent to your profile page. Here you will see your own cards, and you'll be able to add friends through a search bar.

After searching for a user, you'll be directed to their user profile and see their cards and have the option to request to add them or decide to head back to your profile page.
Once you send a request, the other user will either have to refresh their profile page or login to see the request.
If you are the other user, you have the choice to accept or reject or not answer. If you decide to search for the user who sent a request, and try to add them. It will not let you. It will also not let you add the same friend twice.
Otherwise, if you add the user, they'll show up in your friends list. Which you can then look at their cards or request a trade.

When requesting a trade, you'll need to choose one of your cards, and one of their cards to request a trade otherwise you'll be told that you're unable to request the trade.
After requesting a trade, you can go on a third account and request a trade with one of the two users and try and trade for the same card. Once any of these trades are completed, if a user decided to trade one of their cards that is also in another trade,
once the recipient of the request replies to said trade, if either one, or both don't have their respective card anymore. The trade will cancel.

All these trade and friend requests will show up on your profile page once refreshed after being sent a request. Within this page, you will be able to reject or accept trade/friend requests one at a time.

DESIGN,IMPLEMENTATION,WHY?

Now how does everything work?

To start off, there is 4 collections in the database. Cards,Trades,Users,Friendships.
The cards collection is their to begin with.
The users collection is where all the user data is held. (Because we'll obviously need to hold onto their cards,user,pass information)

(I decided to use relationships so I can easily check for relationships between friendships and trades to then easily pass the information to the server and then to the webpages)
The trades collection is to hold a relationship between two users and their cards they're trying to trade and to see if there's a request to begin with.
The friendships collection is to hold a relationship between the two users to see if they're friends or if a request is waiting.
(You can easily look at the schemas for this to understand how it works)

(I decided to keep the login and registration page in the same area to save creating more pages and it's more efficient because to login, you first need to register. It is also the main page so its easier to register)
For registering and logging in, there is a post method once you submit user information on the registration user and password boxes. Once this occurs, that information is then saved to the user collection alongside randomly generated cards.
As for logging in, there is a session id keeping track of the user that is currently logged in. Same method where there is a post method to the server once the login information is given and submitted.

(The reason for redirecting to a new page is so you can have your friends user information with the username you provided, and also see their cards and then decide whether to add this person or not)
As for friendships, with the search bar. It redirects you to a new page containing both user information. Within this page, there is a post method that will send to the server containing both users and create a friend relationship. (Request)

(The reason for again redirecting to a new page is so you can see both your cards and your friends' cards, you friends information shouldn't show up in you're profile page. That's only supposed to contain your personal info)
As for trades, once you've added a friend, you can request a trade and it'll redirect you to another page containing again, both user data and their cards. Here there is a post method that is sent to the server to then create a trade relationship (request)

Once you've received requests to trade or add a friend, these will show up on you're profile page.

(The friend requests only show up depending on the status between the two users)
For friend requests, you just see who it's from and there you can decide to add or reject (if you add, it just sets a friend status for the relationship between the two users which can then be obtained in the database to look at anytime
,if you reject, it just deletes the relationship)
To note, the add and reject sends a post to the server to then add or reject said friend accordingly with the server functions.

(Similar to the friend requests, these only show up depending on the status of the trade)
For trade requests, you will see who it's from, as well as the cards being traded. You can then again decide to add or reject.
(Again adding or rejecting send a post to the server to run certain functions accordingly and set the trade status depending on the user input).

For this assignment, I also decided to not user routes as it felt more tedious to do so and keeping everything in the main server is simpler.