Design Document

The following document explains the design and flow of my application in detail. It also includes the procedure to successfully run and test the application for all features.

Setup:

The first step is to install all the dependencies for my node application. Download the project files to a folder. Open the command prompt to that folder then and open a5 folder. Run the following command:

npm install

This will download the node modules to the a5 folder. The database initializer needs not to be run because the database will be automatically initialized when the server first runs. To clear out the previous database, default database initializer can be used.

Run the application by running the following command from the same directory where the modules were installed:

npm run dev

After running this command, the console should print “Listening on port 3000”. Open a browser (preferably Google Chrome) and enter the following address in the bar:

localhost:3000

This should render the main page i.e. index.ejs

Folders:

I have multiple folders in the project to save files and classify them based on functionality.

The basic backbone files of the application are stored in the **src** folder which includes the main app.js file, the db folder and the middleware. The files in db folder are used to define schema and models for Users and Friends using mongoose.

All the routes of the application are defined inside the routes folder. All get and post requests including login, signup, friend request, friend addition etc. are routed by the files in this folder.

The views folder includes our ejs files for the main pages i.e. login, signup and index. Ejs was chosen as a templating language as it is quite similar to html and provides some functionalities for easier templating.

The css and remaining js files are defined inside the public folder which are imported in the other files to be used.

Design:

Upon running the application, the user lands on index page first which includes the search bar for searching friends and displaying cards etc. The design is kept this way intentionally so that any user that lands on our website may be able to have an idea of the functionalities that the application offers. But, the feature of search is only available to the logged users. As soon as the user clicks on the search bar and tries to enter a letter, he/she is redirected to the login page instead of showing results. The user can also go to login or signup page from the index main page.

The login and signup pages are very conventional. They include a form of 2 fields each i.e. username and password. Each of these fields should be of minimum 5 length to be considered a valid entry. This was included in the design to ensure valid entries in the database.

A logged in user can search for users in the database by their user names and send them a friend request. A user can only send a friend request to a user who is not already in the friend list and is not have a pending request from the same user. The search bar, friend requests and the list of friends to show cards and trade with them are kept on the same page for the sake of ease. A user may be able to analyze his/her trade strategy by viewing each of his friends' cards interchangeably.

Testing Features:

All features have been implemented comprehensively and can be tested for error catching and handling.

- On signup page, enter a username that has already been registered. The page will not allow registering multiple users in the database with same username.
- On login page, enter incorrect username or password. The page will not let the session activate without authentication. The password and username are both case sensitive.
- Without logging in, try to search a username in the search bar. You will be redirected to login page without showing any results. The application won't allow accessing specific details without logging in to the system such as friends list and their cards.
- Create a user by registration and then view its friends and cards after logging in with the same credentials. The user will initially have 10 random cards and no friend.
- Search for friends in the search bar after logging in. Try searching your own name. The search results will not suggest your name. Try searching your friends' names too. The search results will limit you to send friend request to your current friend or to such a user whose friend request is already pending.
- Using the search bar, send friend requests to any user. This 'real-time' functionality can be tested by logging in as another user on another browser or the same browser in incognito mode. Running the application will require the same localhost address there too. The friend request will be received by the other user in no time.

- Try accepting a friend request that is received from some user. The friend list will be updated in real time on both sides.
- If you choose to reject a friend proposal, the request will be deleted from the database. The presence or absence of request can be verified from the database using any software providing access such as MongoDB Compass.