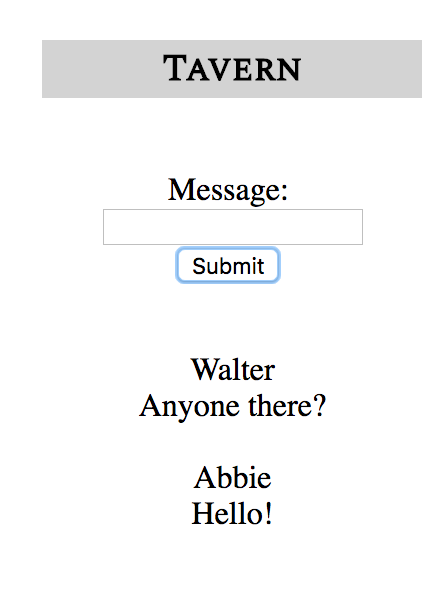
Walter Donnellan

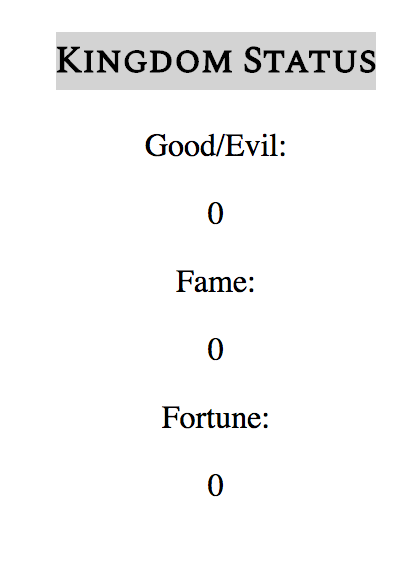
URL: https://m2cfinalproject-txdxyngvdl.now.sh

Rubric Point Description

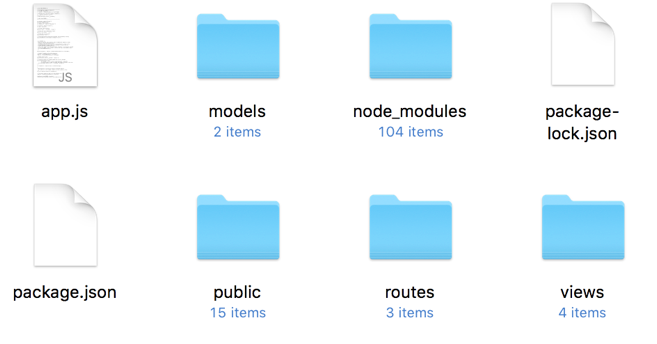
1. I have created a single-player/multiplayer game for my distributed application. Users need to create an account and log into the account in order to participate. The users have a list of choose-your-own-adventure text-based games, and the decisions they make in the games affects the kingdom all the players exist in. The kingdom’s stats determine what adventures are available to be played, and in order to access other adventures the players need to make decisions that change the kingdom’s stats.

Some user interactions include the ability to message one another in-game:

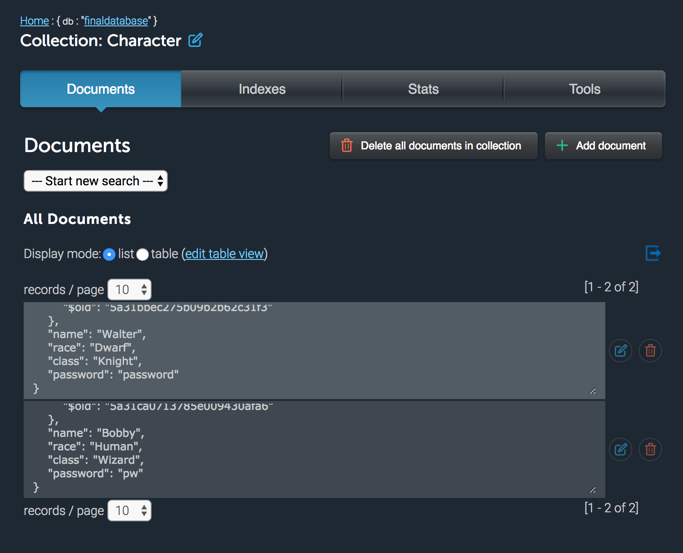
1. The main way in which the content displayed on the client is updated based on the server is in what adventures are displayed. As the kingdom’s stats change, some adventures appear while others disappear in the adventure section of the main page. Another way the client content is updated by the server is through the ‘Tavern’ (a basic message feature where players can chat with one another), where the messages from other players appear. Below is the kingdom status that is updated as the players make decisions in the adventures:



1. I use WebSockets via socket.io on this project. The server and clients emit events back and forth to update the Kingdom Status and the available adventures.
2. The server-side application interacts with other systems by coordinating live collaboration with other users. This is mostly seen in the users collaborating to change the Kingdom Status.
3. I use an MVC architectural style.



1. I use passport sessions to manage web app state information.
2. I store user/character information (name, race, class, password) persistently in a server-side database.



8 & 9. App functions the same on both mobile and desktop.

1. App is deployed and working in the cloud via Zeit Now and /\_src is available.
2. I keep HTML, CSS, and JS separate except for in new\_character.html where I ran into trouble having my HTTP Put in a separate file, so I included it in a script tag.
3. As seen above, I use a Put method in new\_character.html.
4. I have a good coding and commenting style throughout. I did my best to make everything neat.
5. I employed authentication and authorization in this app. /town and /town/index.html cannot be accessed until the user is logged in as a user in the database. The user can also make a new account and then log in with that account.
6. PUBLISH TO GITHUB WITH LICENSE AND README.md