Rabbit Hole documentation:

The concept of The Rabbit Hole requires a bit of explanation. It is sourced from a sci-fi tabletop RPG a friend of mine has made. The rabbit hole is a database where information is both currency and commodity. To be able to get any information out of the rabbit hole, you must first enter information of equal or greater value. Of course, writing a deep learning algorithm simply for this project is a bit out of scope. For this project, the commodity of information is words. Users may enter their 'facts', which the server stores. Each word in a fact is given a value based upon the frequency the server has seen that word, and a fact is worth the sum of the values of the words in that fact and an additional value based on the length of the fact. When users submit facts, they are given coins based on that fact's worth. Users are also able to request the fact catalogue. The client will display related facts and their prices, and if a user has enough coins they may pay and see the fact itself by clicking on the name.

The API handles storage of words and facts, as well as assessing the value of facts. It generates GUIDs for every fact added. It also pulls specific parts of each fact object depending on the get request. Users can add facts, update facts, view the catalogue of facts, and purchase any fact they can afford. They can also request a Head request from the fact catalogue.

String formatting using RegExp was crucial to this project functioning. GUID generation needs to be consistent, and wayward punctuation can mess up both the server when indexing facts and the client when creating fact purchase links. While I haven't accounted for every ASCII/ Unicode character, I have accounted for every punctuation mark a regular American keyboard is capable of producing without special character sequences (i.e. alt-164 for ñ, etc). My implementation of RegExp went very well, thanks to the guide provided by the JavaScript MDN document here.

Nothing much 'went wrong', although I was somewhat pressed for time when programming this app.

Going forward, I would like to add a language library like RiTa JS to detect words (as the current implementation does not, it simply splits on spaces) and perhaps even adjust the value of each word based on what part of speech it is. Adding the ability to search keywords would be a nice change, rather than scroll through every fact. I could optionally increase the security of transactions by requiring keys in the head of requests, and possibly even add user accounts to store coins rather than the local storage. There is also a quality of life change I could implement to keep track of each fact a user has purchased already, either in a basic form in local storage or a more complex library using a user account. The GUID generation could likely be made a little sounder as well.

For above and beyond, I personally believe the scope of this project was more complex than most of the examples I saw in the lecture. I also expanded my knowledge of using RegExp to format strings to help prevent several issues that occur when numbers and punctuation find themselves in some strings and document IDs. Not sure if this next one counts for much, but this documentation is something the server can provide to the user!

3/5/21:

- Ported code from Web API assignment II to serve as a codebase for this project.

3/7/21:

- Updated code to take names and facts.
- Added a word bank object to store and track word usage.
- Words are extracted from facts and counted.
- When updating a fact, the system removes any words that were in the old fact.
- Calculates coins to reward based on words entered.
- Calculates the value of each fact based on the words used.
- Now returns just name and price for each fact.
- Clicking on a fact's name returns the fact itself.
- Facts can have names with spaces.

3/12/21

- The rabbit hole now determines a value of each fact and returns that to the client, allowing the user to store coins in the local storage.
- The rabbit hole now charges users the proper number of coins to be able to access a fact.

3/14/21

- The rabbit hole was given a substantial CSS overhaul.
- The rabbit hole now indexes facts based on GUID instead of name.
- Using RegExp statements to format strings to prevent internal server errors.
- Words now ignore punctuation.
- Improved GUID generation.
- Coin algorithm tweaked to scale values more consistently.
- Documentation is linked in the 'about' link in the page's footer.