The following code dump is relatively straight forward. It is a single page application that is driven by a python Flask server, and a Vue.js front-end. These two are separate entities and therefore are located in the Server and Client folders respectively.

Taking a look at the Server first, there are four items in this folder, app.py, custom\_parser.py, env, and assets. Assets simply contains the different text documents that are being analyzed. If needed other corpuses of text could be added to this folder. The next easier item is the env folder, this is created when install libraries with pip. In our case I used python’s natural language toolkit library, flask, flask-cors, and others.

Taking a look at the actual code, the base app – app.py is the root of the server. When you run python app.py, this file will first import and run a function from the custom-parser.py file which I will address in the next section. After running this function, this Flask server imports the CORS library so that cross origin access requests are allowed on this server. This server has only two routes, the first getAnswers expects to receive a sentence of text, and in turn send back a list of answers that most closely match this result. The other route, getParagraph, is also given a sentence of text – however, this is returning 10 sentences that are the closest to the sentence that is given to it.

The custom\_parser.py module is doing most of computation of the server. The first thing the server does when it is started, is run the parser.sentence\_and\_word\_tokenizer() function. This function takes the corpus of text that was given to us and goes through it parsing it into sentences, paragraphs, and word maps that are then returned. These three objects are indexed and then used in the different routes and other functions to return results. I chose maps for these functions because they have O(1) access time which means it doesn’t matter the length of the corpus you are searching, you should receive results at the roughly the same speed every time. This custom\_parser file relies heavily on the NLTK library for python. This library allows for the tokenization of words and sentences which makes indexing this dump much more accurate.

Now looking at the Client folder, this is mostly all boilerplate that gets created as part of a new Vue.js project. To dive into the parts that I have added go into the src folder. Index.html and App.vue are the two root files of the client. They are also mostly boilerplate and do not really contain much new code other than the importing of libraries, and different files in the index. The important pieces of the Client code are the Home.vue and the Answer.vue components that are contained in the components folder of src. These represent reusable components that make up the functionality and UI of the application. Home.vue shows the landing page and it utilizes the Answer.vue component to create the list of answers that gets returned.

Also, in the router folder I initialized a vue-router which just makes it so that when people visit the home page it takes them to the Home.vue component.