Documentation for File Q&A
The first step involves obtaining and setting up your OpenAI and Pinecone APIs. These keys should be stored within the .env and .yaml files. Next, the source code can be downloaded from this website: https://github.com/openai/openai-cookbook. On the server side, we start by launching a virtual python environment. Then, the *openai* module (using *pip* and *npm*) as well as the dependencies located within *requirements.txt* are installed. The following line should be run in the command prompt to complete the server setup:

```
python .\app.py
```

As for the client, we again use a virtual python environment. The *openai* module is installed with both *pip* and *npm*. Afterwards, the process is completed by typing the following commands into the command prompt:

```
npm install
npm run dev
```

The application can be reached from this url http://localhost:3000, where you can simply drag and drop files to have your questions answered.

Documentation for web-qa.py
The initial setup will require an OpenAI api that should be stored within the .env file. Additionally, we will be using a virtual python environment. A description of the code is given below:

This code is a Python script for crawling hyperlinks from a specified domain. It imports necessary libraries including *Requests*, *BeautifulSoup*, and *deque*. It starts by defining a class to parse the HTML and get the hyperlinks. A function is created to get the hyperlinks from a URL, and another function is created to get the hyperlinks from a URL that are within the same domain. The main function crawls the website by first parsing the URL and getting the domain, then creating a queue to store the URLs to crawl, and a set to store the URLs that have already been seen (no duplicates). It continues crawling until the queue is empty or a maximum number of pages have been crawled. While crawling, it saves the text from each URL to a text file, and the hyperlinks to a CSV file. The crawling process is done in four steps, including the import of necessary libraries, defining functions to get hyperlinks, creating a class to parse HTML, and the main crawling function.

Documentation for traffic_controller.py
This code is written in Python and is using the PYNQ library to interact with a field-programmable gate array (FPGA) board connected to the computer running the code. The code controls a traffic light system that uses LEDs connected to the FPGA board.

The first line of code imports the sleep function from the time module, which is used to create a delay in the execution of the program. The second line of code imports the Overlay class from the PYNQ library, which is used to load a hardware design onto the FPGA board.

The third line of code loads a hardware design file called "ps_gpio_kv260.bit" onto the FPGA board using the Overlay class. This file defines the input/output connections for the board.

The next few lines of code create instances of the GPIO class from the PYNQ library to control the LEDs and button connected to the FPGA board. The GPIO class takes two arguments: the first argument specifies the pin number for the input/output, and the second argument specifies whether the pin is an input or output.

After defining the GPIO pins, the code sets the initial state for the traffic light system. The three LED objects (led1, led2, and led3) represent the red, yellow, and green lights of the traffic lights. The ledswitch object controls which set of traffic lights is active. The count1, count2, and count3 objects control a binary counter that is used to determine which set of traffic lights should be active. The button object represents a physical button connected to the FPGA board that is used to trigger the change in the traffic lights.

The changecounter function updates the binary counter to determine which set of traffic lights should be active. The function takes the current value of the counter as an argument and returns the updated value. The function uses a series of if statements to update the counter based on the current value.

The while loop is the main part of the program that controls the traffic lights. The loop runs continuously, checking the value of the button object to see if the button has been pressed. If the button is pressed, the code runs a sequence of instructions to change the traffic lights. The sequence of instructions uses nested for loops to create a delay between each change in the traffic lights. The changecounter function is called after each loop to update the binary counter. If the button is not pressed, the code runs a sequence of instructions to keep the traffic lights in their current state.