Objective

The objective of this assignment is to utilize what you have learned so far in this course (i.e., HTML, CSS, JavaScript, and jQuery) in the design and creation of an artificial intelligence (AI) chatbot-powered tool. Operations such as perceiving, synthesizing, and inferring information are all examples of how machine-based intelligence, or AI can be exhibited. There are numerous applications of artificial intelligence, such as recommender systems (e.g., Amazon, Netflix, YouTube), human speech recognition (e.g., Siri, Alexa), and human-computer conversation tools (e.g., ChatGPT). A chatbot is a human-computer interactive entity that attempts to replicate human conversations via text and/or voice. An AI chatbot (such as ChatGPT) integrates machine intelligence when responding to users. Using client-side technologies such as HTML, CSS, and JavaScript, you will develop an interactive browser-based conversation tool that uses OpenAI's API to synthesize and infer information.

Brief Overview

You may already be familiar with chatbots, which provide a way for users (or even other machines) to interact with each other through a conversation. Modern chatbots include AI components that are capable of carrying on a conversation with a human user in natural language while to a great extent simulating the way a human would behave in typical human-to-human conversations. Recently, OpenAI, an AI research company founded in 2015, developed ChatGPT, a generative AI chatbot. Google recently introduced its own AI chatbot called Bard. These AI chatbots utilize a variety of learning models in order to simulate human conversations and can be tailored to specific applications. Other known chatbots include Poe (developed by Quora), Tay (originally released by Microsoft), and Ernie Bot (developed by Baidu).

Building a tool that maintains a conversation between an end-user and a chatbot requires the development of a software entity that is capable of maintaining users' inputs while communicating with a generative chatbot on a back-end system (e.g., a web server) to respond to users' inquiries or inputs. Developing a chatbot is beyond the scope of this course. However, we can use existing AI chatbots such as ChatGPT from OpenAI to support the development of an interactive conversational tool that we plan to build.

Using HTML, CSS and JavaScript to Build Interactive Messaging or Conversational Tool

Throughout the quarter so far, you have learned multiple tools such as HTML, CSS, and JavaScript (including jQuery). These technologies are all client-side, which means they reside on the client machine. However, we can also use JavaScript, for example, to communicate with back-end technologies, such as ChatGPT, to relay messages between end-users and back-end systems. To communicate with a back-end chatbot, you can use web services. That is, a conversational tool generates a service request through JavaScript to a back-end chatbot with user inputs as parameters, and the chatbot processes the request to generate a response that contains the response information. Using JavaScript, we can then parse the response, apply our own presentation style, and append HTML tags within an HTML document to make this tool real-time or interactive. In Module 3, you learned how a client browser can be used to communicate with a web service (slide 24, lecture 7, Module 3 GitHub example, and tutorial 3). We will utilize this knowledge to build this conversational tool.

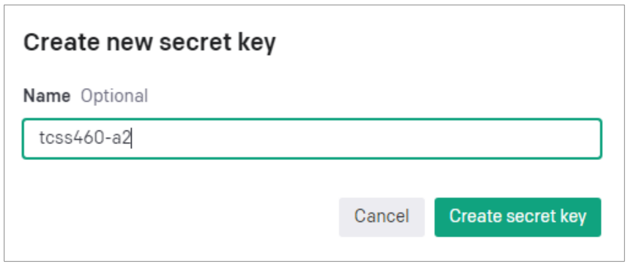
Part A: Using ChatGPT and Developing Conversational Tool using HTML, CSS and JavaScript

Step 1: Acquire an API Key from OpenAI

As a first step, we need to sign up with OpenAI to obtain a valid API key that will allow us to communicate with the ChatGPT system. ChatGPT accepts user input in the form of text and emits a response in the form of text. OpenAI maintains other systems such as GPT-4, which is a multimodal system that can accept images and text as inputs. DALLE-E 2 is yet another AI system that can create realistic images and art based on a given user's input. To get started and for the purpose of this assignment, we will use ChatGPT. However, usage of such systems is often moderated and can be quite expensive. That is, every request that you perform as a developer is logged, and one has to pay for each request they generate. In fact, the OpenAI price model is based on tokens (think of them as words or chunks) and not on the number of requests you make.

Thankfully, OpenAI provides a $5 free trial to try their APIs, including ChatGPT. There are a number of models that we can use, such as text-davinci-003, curie, ada and babbage. The trial credit should be sufficient to complete the assignment with extensive testing and/or troubleshooting using these models. To get started, you need to obtain a key.

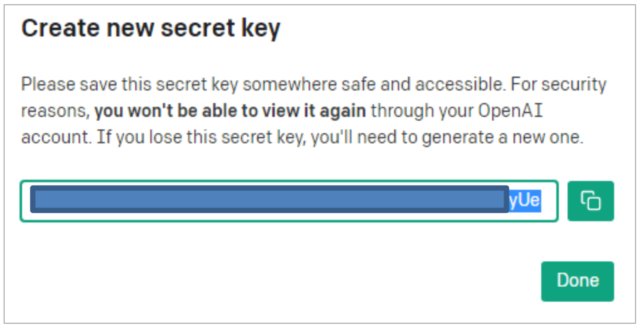
1) Using a browser, go to https://openai.com and click on the **Sign Up** link (top-right corner) to register for a personal account.
2) Login to your OpenAI account. You should see a welcome screen. On the top-right corner, click on **Personal | View API Keys**
3) Click on the button "**+ Create new secret key**" and provide a name (e.g., tcss460-a2). Click the "Create Secret Key" green button to generate a new key.


Create new secret key

Name Optional

tcss460-a2

4) A pop-up window will appear with the generated key. **Ensure to copy the API key, as you will not be able to retrieve it again.** You will need the API key for future steps.


Create new secret key

Please save this secret key somewhere safe and accessible. For security reasons, **you won't be able to view it again** through your OpenAI account. If you lose this secret key, you'll need to generate a new one.

[redacted]yUe

5) You can always keep track of the usage in the usage section of your account: https://platform.openai.com/account/usage
6) Once you have obtained an API Key, navigate to Canvas | Assignments | Assignment 2 and download the ***a2.zip*** file, which contains the required startup code for this assignment. The code is documented and provides links to open-source code resources. Therefore, **ensure that you read the code and documentation to comprehend how it operates.**

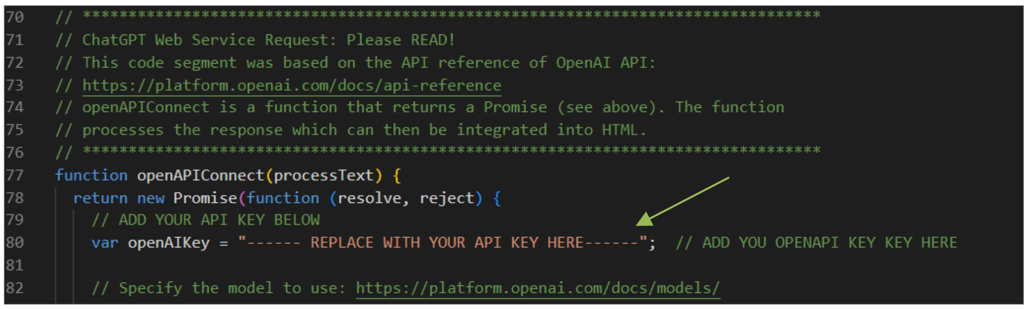
Step 2: Execute the Code

Chrome has been used to test the startup code. You can try other alternative browsers. However, Chrome is preferable because some of the natural language processing tools we will use are generally only compatible with and/or perform optimally on the Chrome browser. The sample code should allow you to execute queries to ChatGPT immediately once you add your OpenAI API key (Part A | Step 1) as outlined below.

1) Extract the a2.zip file into a local folder named 'a2'. The code contained within these files is based on open-source examples that are provided on the web (see documentation). The compressed zip file contains three files, including:

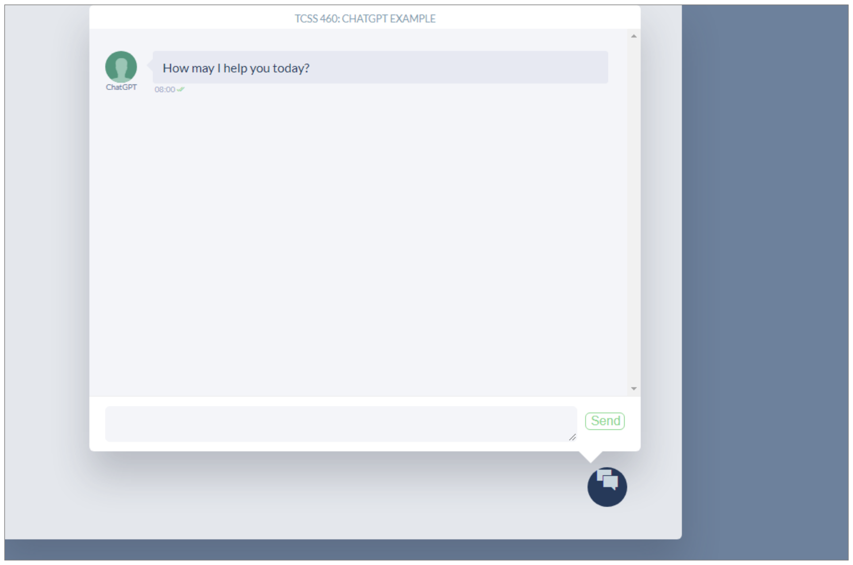
-	**index.html**	→ main html file that develops the interactive conversational tool
-	**script.js**	→ JavaScript file that communicates with OpenAI API for ChatGPT
-	**style.css**	→ stylesheet that controls the design and style of HTML elements

2) Now that you have all the files in a single folder, we need to add the API key obtained in Step 1. Open the "script.js" file using your favorite source code editor and replace the following string with the API key you obtained (line 80). Save the file.



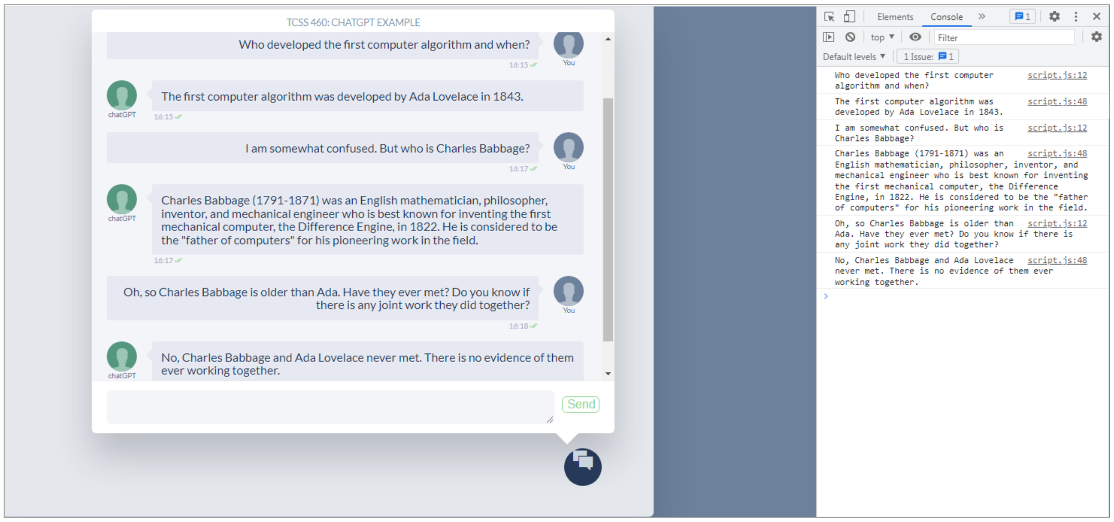
```
70    // ****************************************************************************
71    // ChatGPT Web Service Request: Please READ!
72    // This code segment was based on the API reference of OpenAI API:
73    // https://platform.openai.com/docs/api-reference
74    // openAPIConnect is a function that returns a Promise (see above). The function
75    // processes the response which can then be integrated into HTML.
76    // ****************************************************************************
77    function openAPIConnect(processText) {
78      return new Promise(function (resolve, reject) {
79        // ADD YOUR API KEY BELOW
80        var openAIKey = "------ REPLACE WITH YOUR API KEY HERE------";  // ADD YOU OPENAPI KEY KEY HERE
81
82        // Specify the model to use: https://platform.openai.com/docs/models/
```

3) Now, double-click on "index.html" to open it in a browser. You should see the following interface, or similar. You may wish to access the developer tool and check the console section to troubleshoot any errors.



4) Ask ChatGPT a few questions and analyze how it works. Note that you may encounter a slight delay (a few seconds). This is due to the conversational tool sending requests and awaiting responses from the OpenAI API. OpenAI API servers take time to respond, and this may vary depending on many factors such as the availability of resources, the number of concurrent users, and the type of account (e.g., trial vs. paid), among others.

Below is an example of a conversation with ChatGPT.



Who developed the first computer algorithm and when? 16:15

The first computer algorithm was developed by Ada Lovelace in 1843. 16:15

I am somewhat confused. But who is Charles Babbage? 16:17

Charles Babbage (1791-1871) was an English mathematician, philosopher, inventor, and mechanical engineer who is best known for inventing the first mechanical computer, the Difference Engine, in 1822. He is considered to be the "father of computers" for his pioneering work in the field. 16:17

Oh, so Charles Babbage is older than Ada. Have they ever met? Do you know if there is any joint work they did together? 16:18

No, Charles Babbage and Ada Lovelace never met. There is no evidence of them ever working together.

```
Who developed the first computer        script.js:12
algorithm and when?
The first computer algorithm was        script.js:48
developed by Ada Lovelace in 1843.
I am somewhat confused. But who is      script.js:12
Charles Babbage?
Charles Babbage (1791-1871) was an      script.js:48
English mathematician, philosopher, inventor, and
mechanical engineer who is best known for inventing
the first mechanical computer, the Difference
Engine, in 1822. He is considered to be the "father
of computers" for his pioneering work in the field.
Oh, so Charles Babbage is older than    script.js:12
Ada. Have they ever met? Do you know if there is
any joint work they did together?
No, Charles Babbage and Ada Lovelace    script.js:48
never met. There is no evidence of them ever
working together.
>
```

Congratulations! You have successfully developed a conversational tool using an AI chatbot.

Part B: Adding Speech to Text Recognition using the Web Speech API

In Part A, you implemented a conversational tool that relies on the user entering text in a textbox (e.g., a message form field). Wouldn't it be more efficient if one could speak inputs and questions into the textbox? Obviously, it would. Consequently, we will add an extension to Part A that integrates speech so that you can speak your own questions or inquiries into the conversational tool. This may sound simple, but technically, performing this simple step in a browser can be quite complicated for a variety of reasons.

Supporting speech requires an API capable of capturing audio, identifying words, and generating equivalent words and complete sentences. Such an API must be trained to recognize accent variations, support multiple languages, can intelligently relate best matching words within a speech, and other factors. There may be variations in the speech input support (e.g., speech-to-text) between browsers. In addition, speech-to-text support requires microphone access, meaning that a user must allow a webpage (e.g., an HTML file) to access a microphone, which is generally restricted by default on many browsers.

When a web page requires access to a microphone, you may frequently receive a notification requesting permission. We will integrate speech-to-text into our conversational tool through the addition of an additional JavaScript file. Thankfully, an experimental standard within browsers allows developers to process voice input from users. Chrome has a Web Speech API, whereas Firefox has a Speech Recognition API, both of which enable building voice-enabled apps. Hence, we will use a speech API to make our conversational tool voice-enabled! This handout uses Chrome's Web Speech API for this assignment. For a demonstration of the Web Speech API, click here. For information on other alternative speech APIs, click here. For troubleshooting the microphone on Chrome, click here.

Step 1: Download Speech Recognition JavaScript

1) Go to the following URL: https://github.com/zolomohan/speech-recognition-in-javascript
2) Click and download the file named "speechRecognition.js"
3) Using a source code editor, open the file and save it as **speech.js** in the same folder as Part A.
4) Examine the code to understand how it works. Let's examine the first line:

```
if ("webkitSpeechRecognition" in window) { ... }
```

This line attempts to identify the speech API if it is supported by the browser. Note that this code file was developed to match an HTML file that contains elements that are non-existent in the HTML file we developed in Part A. (e.g., “status” element, “select_dialect”, “final”, etc.). Hence, we need to modify the file and our index.html file to integrate this code into our conversational tool correctly.

5) Update the following line (should be line 7)

```
  speechRecognition.lang = document.querySelector("#select_dialect").value;
```

to

```
  speechRecognition.lang = 'en-US';
```

6) Modify the following line (should be line 31) – we are not using final element, but **message**:

```
    document.querySelector("#final").innerHTML = final_transcript;
```

to (message is a textbox and does not have innerHTML. Hence, we use **.value** instead:

```
    document.querySelector("#message").value = final_transcript;
```

7) Comment the following line (should be line 32)

```
  //document.querySelector("#interim").innerHTML = interim_transcript;
```

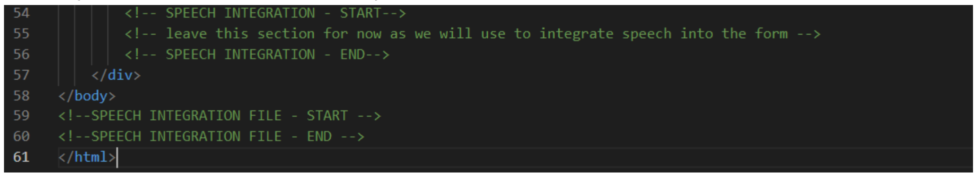
8) We need to clear the value of final transcript every time we access the start button. Hence, add the following line of code (line 36 shown below as **final_transcript = "";**) after line 35:

Line 35
Line 36
Line 37

```
document.querySelector("#start").onclick = () => {
    final_transcript = "";
    speechRecognition.start();
```

Step 2: Updating index.html

We need to update the index.html from Part A to include the buttons start and stop and a paragraph tag that will be used for showing the status of speech usage. Open the **index.html** file from Part A and locate this section (lines 54-56, lines 59-60):



```
54              <!-- SPEECH INTEGRATION - START-->
55              <!-- leave this section for now as we will use to integrate speech into the form -->
56              <!-- SPEECH INTEGRATION - END-->
57          </div>
58      </body>
59      <!--SPEECH INTEGRATION FILE - START -->
60      <!--SPEECH INTEGRATION FILE - END -->
61      </html>
```

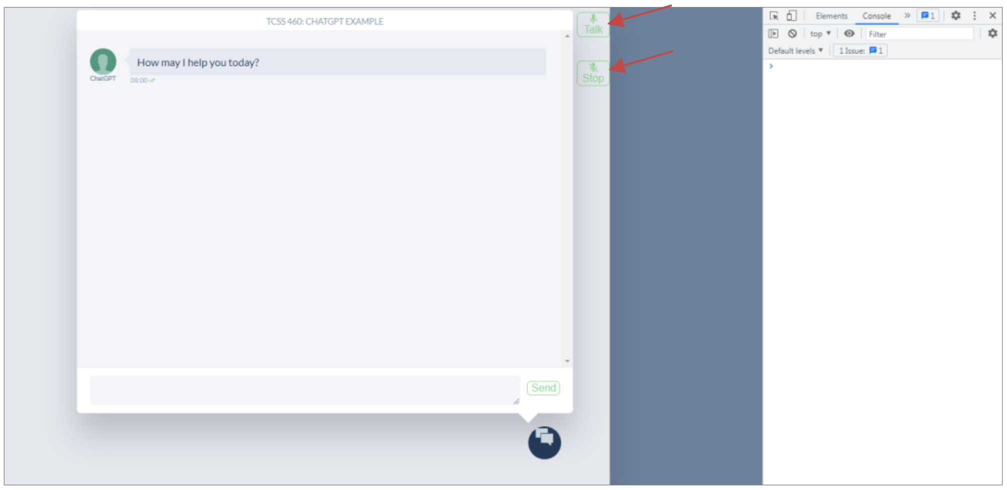
Then, **add** the following lines of code represented by yellow color / highlight:

```
             <!-- SPEECH INTEGRATION - START-->
            <!-- leave this section for now as we will use to integrate speech into the form -->
            <div class="mt-4">
              <button class="btn btn-primary ion-android-microphone" id="start">Talk</button>
              <br><br><br><br>
              <button class="btn btn-secondary ion-android-microphone-off" id="stop">Stop</button>
              <p id="status" class="lead mt-3 text-light" style="display: none">Listening ...</p>
            </div>
            <!-- SPEECH INTEGRATION - END-->
        </div>
</body>
<!--SPEECH INTEGRATION FILE - START -->
<script src="speech.js"></script>
<!--SPEECH INTEGRATION FILE - END -->
```

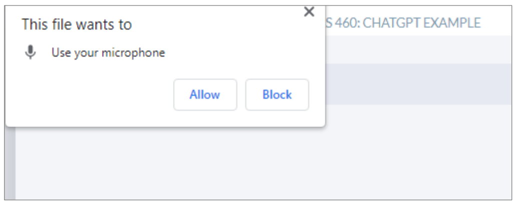
Step 3: Using Speech to Text

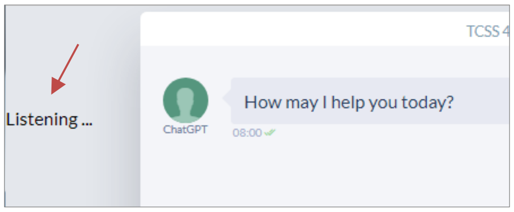
Ensure that your computer (PC, laptop, tablet, etc.) has access to or is equipped with a microphone to be able to complete this step.

1) Open the index.html file from Part A, and access the developer tool | console and ensure there are no errors. You should now see **Talk** and **Stop** buttons on the top-right of the conversational tool window:

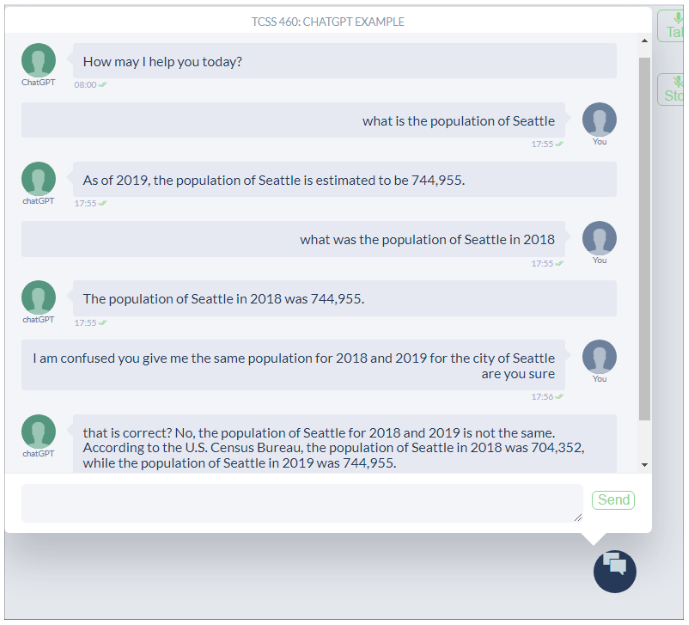


2) Click on the **Talk** button. You should see a notification request from the browser requesting to use the microphone. Click **Allow**. Then, express your voice-enabled request or input. Note that a paragraph tag to the top-left will appear with the keyword "**Listening**". This represents the status element that we added in Step 2 to the index.html.





3) Once finished speaking, click on the **Stop** button. Then click the Send button to chat with ChatGPT. Alternatively, you can wait after you speak your input (idle time is about five seconds), and then clicking on the stop button will cause the input to be sent to ChatGPT automatically. On the right is an example of asking questions using speech about the population of the city of Seattle (in 2018 and 2019), where the data response was the same!



TCSS 460: CHATGPT EXAMPLE

ChatGPT: How may I help you today? 08:00

You: what is the population of Seattle 17:55

chatGPT: As of 2019, the population of Seattle is estimated to be 744,955. 17:55

You: what was the population of Seattle in 2018 17:55

chatGPT: The population of Seattle in 2018 was 744,955. 17:55

You: I am confused you give me the same population for 2018 and 2019 for the city of Seattle are you sure 17:56

chatGPT: that is correct? No, the population of Seattle for 2018 and 2019 is not the same. According to the U.S. Census Bureau, the population of Seattle in 2018 was 704,352, while the population of Seattle in 2019 was 744,955.

Important Note that Deserves Mentioning Before Creating Your Conversation!

Is the Web Speech API really client-side?

Although the W3C Community Group that developed this speech recognition experimental feature specifies in the Web Speech API specification that it "defines JavaScript API to enable web developers to incorporate speech recognition and synthesis into their web pages." Chrome does not perform speech recognition locally on your machine. In fact, you will no longer be able to send and receive data from the API if you load index.html and then turn off your Wi-Fi or Internet connection. Since the Web Speech API does not run on the client machine, as one might expect, it indicates that browsers share your data through a third party when using it! According to the Google Privacy whitepaper, the Web Speech API" uses Google's servers to perform the conversion." Click here to learn more about Google's Speech to Text privacy. Microsoft Edge provides a similar note on the same topic in their Edge privacy whitepaper. Aside from privacy concerns, having the conversion of the API on the server side also raises a fundamental question: **What bandwidth and data transfer implications would this have if you were to develop a mobile application that uses Web Speech API-enabled browsers on mobile devices?**

Task: Using Conversational Tool for Storytelling or Interesting Conversation

Tips while using ChatGPT:

- It may take a few attempts to generate useful responses using the conversational tool effectively. Therefore, do not be discouraged if your initial attempts to obtain answers from ChatGPT are ineffective or unsuccessful.
- ChatGPT is trained to answer questions, but its responses depend on how precisely the questions are posed. That is, you should always aim for clarity in your questions in order to receive relevant responses from ChatGPT.
- Take notes as you create a story so that you can connect the ideas and have engaging conversations with ChatGPT.
- Be mindful of your limited API usage credit. Modify your tasks (e.g., inquiries to ask, magnitude of conversation, etc.) accordingly to ensure you can complete it with the available free credit. Create a conversation that is professional, fluent and natural for this assignment.
- You may wish to prepare notes for your conversation and discussion questions prior to interacting with ChatGPT.

ChatGPT is an extremely powerful tool. As a developer, you can use HTML, CSS, and JavaScript to create an even more powerful AI tool that can be used for a variety of applications. This tool can be used for a variety of purposes, including storytelling, smart assistants, question-and-answer systems, searching for answers, and more. In this regard, your task is to record a 3- to 7-minute video (you do not have to appear in the video) demonstrating a conversation between you and ChatGPT. This conversation should support both text-based and voice-enabled questions. The ratio between manual and voice-enabled is up to you.

The conversation may take the form of a story, requesting assistance in creating characters for a story or a game, or the generation of ideas for a task you wish to complete. The conversation should be interactive, with your questions related to or based on the responses provided by ChatGPT. In other words, your conversation should lead to an enlightening story, a coherent result, or a conclusion, and your questions should not appear to be isolated or unrelated. That is, use the HTML, CSS, and JavaScript-created conversational tool to tell a story or carry on a useful or fruitful conversation with ChatGPT. Personal narratives, jokes, legends, favorite sports, consensus, propaganda, misinformation, coding, and programming practices, among others, may be included in this conversation. A good conversation or story typically has a beginning, middle, and conclusion. As a software developer, think of ways that can test the limits or scope of ChatGPT from a technical perspective and identify ways on how the system is capable of addressing technical issues or contradicting opinions or thoughts.

Need some ideas: This link provides a YouTube video on having ChatGPT Interviewed on TV. Here is another interesting conversation on YouTube with ChatGPT with respect to software development.

Deliverable (upload to Canvas → Assignments → Assignment 2) the following files:

Place your assignment in a folder with the named ***assign2_username*** (for example, assign1_ealmasri). Place all materials used or required to evaluate the assignment, including CSS file, JavaScript files, and images in this folder. Upload your compressed folder into a zip file to Canvas.

☐	**assign2_username.mp4** containing recorded short video of your conversation with ChatGPT.
☐	**assign2_username.pdf** containing a brief self-reflection on the technical merits of ChatGPT (e.g., usefulness, accuracy, clarity, consistency, etc.). In addition, describe your experience in the development of the conversational tool using technologies such as HTML, JavaScript, and CSS. What are your thoughts on the use of artificial intelligence chatbots in education? What were the benefits of completing this task from a client/server programming perspective? Which portion of the assignment was successful, in your opinion? Which portion of the assignment did not meet your expectations? What are your thoughts on privacy-related issues when it comes to utilizing server-side services like Web Speech API? Would you recommend utilizing AI tools like ChatGPT in other TCSS or programming courses? What contribution has this assignment made to your understanding of course topics or concepts? You may also provide any additional comments regarding the overall assignment. You are not required to answer each and every question listed. These questions are provided to make writing the self-reflection simpler to accomplish. The self-reflection is limited to **400 words maximum**.

Grading Rubric (15 marks)

☐	**AI Conversation/Storytelling (assign2_username.mp4)** - [2 marks] Conversation/Story was understandable with a good storyline - [8 marks] Conversation/Story was creative and original - [2 marks] Conversation/Story was fluent with natural or occasional pauses (e.g., good flow) **Reflection (assign2_username.pdf)** - [1 mark] Thoughtfulness (presents some thoughts and effort) - [1 mark] Analysis/Insight (provide more than a simple description of the experience)\