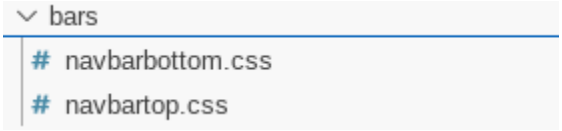
1. This part of the code is complex because I had to compile and organize many different Javascript files in order for the code to remain neat and organized. While I could have included everything in the html file itself, by having each button as its own class with its own function. By organizing it in this way, if there was an issue with a button all I would have to do is to open the button's file, rather than having to search for it within hundreds of lines of code. It also allows for better optimization, as instead of having to rewrite the button instance within the html file multiple times, it and all of its functions could be written a single time within the js file, and only calls to its functions and creation of object types would have to be written within the main code. Within each button are also comments to explain its function, and each folder expresses the area in which the buttons go on the Graphical User Interface.



2. Similar to the Javascript files, I created multiple Cascading Style Sheets in order to organize the way that my Graphical User Interface would work. I decided to do this because it was easier to visualize within my head, and that each part of the Graphical User Interface would require unique design to fit within the overall structure of the code. For example, navbartop, which is the navigation bar situated at the top of the screen, required areas to fit dropdown menus, while the navigation bar at the bottom would hold buttons for the playback feature. Each part of the Cascading Style Sheets were unique in how they required to be designed, contributing to my decision to put them within different files. Furthermore, it makes editing the code much easier, as I could simply edit within the file, without worrying how it affects other aspects of the code. This makes minor tweaks very easy, as I could open the live website in another tab and watch how the changes affect it live.

```
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7
  <link rel="stylesheet" href="../bars/navbarbottom.css" />
  <link rel="stylesheet" href="../bars/navbartop.css" />
  <script type="text/javascript" src="../bars/editbuttons/cutlength.js"></script>
  <script type="text/javascript" src="../bars/editbuttons/equalizer.js"></script>
  <script type="text/javascript" src="../bars/editbuttons/pitchadjust.js"></script>
  <script type="text/javascript" src="../bars/editbuttons/tempo.js"></script>
  <script type="text/javascript" src="../bars/filebuttons/upload.js"></script>
  <script type="text/javascript" src="../bars/filebuttons/save.js"></script>
  <script type="text/javascript" src="../bars/playbackbuttons/pause.js"></script>
  <script type="text/javascript" src="../bars/playbackbuttons/play.js"></script>
  <script type="text/javascript" src="../bars/playbackbuttons/restart.js"></script>
```

3. This code is complicated because it required me to require an upload of all Javascript and Cascading Style Sheets before running the rest of the code. This was due to the fact that I wrote all the functions and classes of the buttons, as well as the layout and design of the Graphical User Interface, in separate files. As a result, when the HTML file itself is run, none of the style of the Graphical User Interface nor the functionality of the buttons will load with it. As a result, preloading of both of these things is required. This can be done using <script> and linking the file within the repository to be run with the HTML file. As evidenced in the image, along with the <source> tag, the source of the file also needs to be written down for it to be imported. This can be done by typing it up as shown. Furthermore, if I wanted to utilize Cascading Style Sheets written in a separate file, they will also need to be linked, but in a different manner than Javascript, which is shown in the picture above.

```
<div class="navbar">
  <div class="dropdown">
    <button class="dropbtn">File
      <i class="fa fa-caret-down"></i>
    </button>
    <div class="dropdown-content">
      <button id="upload" style="width:100%"
      <button id="save" style="width:100%" c
    </div>
  </div>
  <div class="dropdown">
    <button class="dropbtn">Edit
      <i class="fa fa-caret-down"></i>
    </button>
    <div class="dropdown-content">
      <button id="pitchadjust" style="width:
      </a>
      <button id="tempoadjust" style="width:
      </a>
      <button id="cutlength" style="width:10
      </a>
      <button id="equalizer" style="width:10
```

4. This code is complicated because it shows the actual utilization of how the different buttons and menus are coded into the HTML. This picture focuses on the coding of the navigation bar and the dropdown menus, not the buttons. This is complicated because I have multiple different navigation bar styles and dropdown styles. I had to make sure I was applying the right navigation bar class, because one would be situated at the top and one would be situated at the bottom. This can be seen by the declaration of the class at the very top. In this picture, it would be the top navigation bar. Within the of the navigation bar, the buttons and dropdown menus would be coded. This is because these elements occur within the navigation bar, so they would have to be contained within the to stay within the same line. This line of thinking is the same for including buttons within the , as without being in the same they would not be included within the dropdown menu.

```
<div class="dropdown">
  <button class="dropbtn">Recorc
    <i class="fa fa-caret-down">
  </button>
  <div class="dropdown-content">
    <button id="record" style="v
    <button id="pause" style="wi
    <button id="stop" style="wic
  </div>
</div>
```

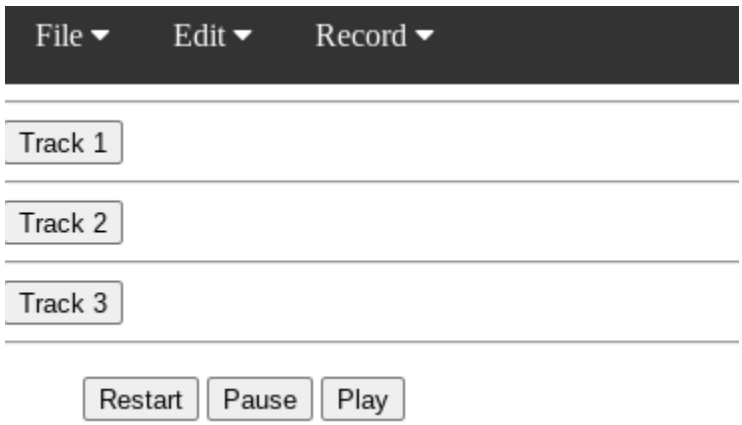
5. This code is complicated because it shows the utilization of the buttons and their functionality. Because each Javascript file has already been imported at the beginning of the code, each button has been loaded with unique functionality. As a result, as that needs to be done is to classify the button with the correct id. This can be seen within the code using the <button> tag. After declaring that the element is a button, an id is assigned matching the correct button Javascript file. However, the correct button id is required to ensure that the corresponding button on the Graphical User Interface has the correct functionality, else the code will not work.

```
#nav {
    background-color: #262626;
    width: 100%;
    height: 50px;
    box-shadow: 0px 1px 50px #5E5E5E;
    position: fixed;
    bottom: 0px;
}

.title {
    color: #EDEDED;
    font-family: verdana;
    font-size: 25px;
    width: 350px;
    margin-top: 6px;
    margin-left: 150px;
    font-weight: bold;
    float: left;
}
```

6. This code is complicated because it shows the formatting of the Cascading Styles Sheets. Specifically, it shows the code for the bottom navigation bar. As seen, the color, font size, margin, and size parameters all need to be coded for. However, the code is virtually the same

as the top navigation bar, except for the code that allows it to be floating while scrolling, the box surrounding the buttons, and the position of the bar itself. Because the space on the top has already been occupied by the top navigation bar, the only other space that the other navigation bar could logically occupy would be the bottom.



7. This code is complicated because it shows how the Cascading Styles Sheets, Javascript, and HTML come together to form the GUI and its functionality. For the top navigation, it holds the dropdown menus with file editing functionality, microphone functionality, and editing functionality. The track buttons allow for the upload of audio files to be edited, and the playback navigation bar at the bottom allows for the user to hear how their edits have effected the audio files.