```
        <div id="myModal" class="modal fade" role="dialog">
          <div class="modal-dialog">

            <!-- Modal content-->
            <div class="modal-content">
              <div class="modal-header">
                <button type="button" class="close" data-
                dismiss="modal">&times;</button>
                <h4 class="modal-title">Give Melissa The Audio File</h4>
              </div>
              <div class="modal-body">
                <form method="POST" enctype="multipart/form-data"
                action=""><input type="file" name="myfile" /><br/><input
                type="submit" value="Submit"/></form>
              </div>

            </div>

          </div>
        </div>
</body>
</html>
```

First you include all the necessary scripts and CSS files for adding the functionality and styling, respectively. Then comes the `canvas` element, which displays the waveform simulation of the sounds detected by the microphone. The other `canvas` element represents the waveforms of the recorded audio from which you want to create the WAV file.

You then have the `controls` section, which contains a button to toggle starting and stopping recording the user's voice. The next button saves the WAV file that has been recorded to your system. The third button opens a bootstrap modal dialog that gives the user the option to upload the file and submit it to Melissa. As soon as the file is uploaded, the back-end code runs `Python main.py audio_file.wav` in the terminal using the `os.system()` function. The WAV audio file is then sent to Google Speech Recognition for conversion of the speech to text. The rest of the machinery works exactly as it did before.

Although I haven't included the static files here, I would still like to discuss the `audiodisplay.js` file:

```
function drawBuffer( width, height, context, data ) {
    var step = Math.ceil( data.length / width );
    var amp = height / 2;
    context.fillStyle = "silver";
    context.clearRect(0,0,width,height);
    for(var i=0; i < width; i++){
        var min = 1.0;
        var max = -1.0;
        for (j=0; j<step; j++) {
            var datum = data[(i*step)+j];
```