

PROJECT SPONSOR NATION INSTITUTE OF HEALTH, NIDCD.

THIS IS A 2018C RELEASE IT INCLUDES ONLY PHP AND DOES NOT INCLUDE
NODEJS OR REACT. THOSE WILL BE COMING IN THE FUTURE RELEASE OF
THIS DOCUMENT.

THIS WORK IS SUPPORTED BY THE NATIONAL INSTITUTE ON DEAFNESS
AND OTHER COMMUNICATION DISORDERS, OF THE NATIONAL INSTITUTE OF
HEALTH (NIDCD/NIH). R01DC015436: "A REAL-TIME, OPEN, PORTABLE, EX-
TENSIBLE SPEECH LAB" TO THE UNIVERSITY OF CALIFORNIA, SAN DIEGO.

[HTTP://OPEN SPEECHPLATFORM.UCSD.EDU](http://openspeechplatform.ucsd.edu)

WEB-APPS USER GUIDE

OSP TEAM

Copyright © 2020 <http://openspeechplatform.ucsd.edu>

PUBLISHED BY OSP TEAM

OPENSPEECHPLATFORM.UCSD.EDU

Copyright 2017 Regents of the University of California

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

First printing, June 2020

Contents

Introduction 9

<i>1.1</i>	<i>What is in this guide?</i>	9
<i>1.2</i>	<i>What's the bigger picture?</i>	9
<i>1.3</i>	<i>What's under the hood?</i>	10
<i>1.4</i>	<i>OSP web-app Landing Page</i>	11
<i>1.4.1</i>	<i>Testing the OSP web-app Landing Page</i>	11
<i>1.4.2</i>	<i>Helpful tools and tutorials</i>	11
<i>1.4.3</i>	<i>Under the hood of the OSP web-app Landing Page</i>	12

The Researcher Page web-app 15

<i>2.5</i>	<i>Testing the Researcher Page</i>	15
<i>2.5.1</i>	<i>Testing Amplification Parameters</i>	15
<i>2.5.2</i>	<i>Testing Noise Management Parameters</i>	16
<i>2.5.3</i>	<i>Testing Feedback Management Parameters</i>	16
<i>2.6</i>	<i>Researcher Page Documentation</i>	18
<i>2.6.1</i>	<i>Amplification Parameters</i>	18
<i>2.6.2</i>	<i>Noise Management Parameters</i>	20
<i>2.6.3</i>	<i>Feedback Management Parameters</i>	21

4 Alternate Forced Choice (4AFC) web-app 23

<i>3.1</i>	<i>Testing the 4AFC web-app</i>	23
<i>3.2</i>	<i>4AFC Documentation</i>	24

<i>AB Test web-app</i>	28
4.1 <i>Testing the AB Test web-app</i>	28
4.2 <i>AB Test Documentation</i>	29
<i>Environmental Momentary Assessment (EMA) web-app</i>	31
5.1 <i>Testing the EMA web-app</i>	31
5.2 <i>EMA Documentation</i>	32

List of Figures

- 1.1 Diagram of OSP system. Everything in the white square of this diagram has to do with the embedded webserver (EWS) in this guide. The EWS uses PHP to communicate to the RT-MHA. The EWS literally hosts a webserver, on which users can access webpages or web-apps. 9
- 1.2 Diagram of the file-tree of the EWS, and the specific files involved in the rendering and functionality of the researcher page controlling amplification parameters. 10
- 1.3 Diagram of the file-tree of the EWS, and the specific files involved in the rendering and functionality of the researcher page controlling amplification parameters. 11
- 1.4 This is code from the web.php file, located in the Routes folder. You may notice that the code lines in the screenshots do not match the code lines in your download. Do not worry, this is simply the result of comments being added to the code, or things being moved around. 12
- 1.5 Screenshot of the controller for this web-app. This bottom portion that you see contains the functions which render many of the web-apps. The top portion of this screenshot features the code for pushing the last row of the Amplification page table. Notice that there are several web-apps which receive commands from this controller.
13
- 1.6 This is code from the welcome.blade.php file, located in the Resources>Views folder. It contains the HTML skeleton of the web-app, as well as styling elements. (Note: You can also follow along in your own development environment, browser or Finder. (note: these screenshots of code are taken in a development environment called Atom. If you download Atom for your computer, you can go to File>Open> and select the directory that you have this code saved in. In other environments, there may be no colors) 13
- 2.7 Researcher page web-app, in the amplification parameters section. The url of this page is /researcher/amplification Note that the URLs of the noise and feedback management pages are different. 15

- 2.8 Researcher page web-app, in the Noise Management section. The url of this page is /researcher/noise. 16
- 2.9 Researcher page web-app, in the feedback parameters section. The url of this page is /researcher/feedback 17
- 2.10 This is a screenshot of the start of the table in the bottom of the amplification parameters page. You may notice that the code lines in the screenshots do not match the code lines in your download. Do not worry, this is simply the result of comments being added to the code, or things being moved around. 18
- 2.11 This is a screenshot of the code for the researcher page routes found in web.php Notice that the route for the amplification parameters page is the first line under the comment "Researcher Routes" 19
- 2.12 This is a screenshot of the controller code for this web-app. You can see 20
- 2.13 Screenshot of the code of the feedback.blade.php file, which contains the content that is used to visualize the feedback management web-app, also called the view. 20
- 2.14 Researcher page web-app, in the amplification parameters section. The url of this page is /researcherAmplification Note that the URLs of the noise and feedback management pages are different. 22
- 3.15 This is code from the 4afc1.blade.php file, which shows some of the JavaScript functionality, which contributes to the buttons being able to update with each question. . 24
- 3.16 This is code from the 4afc1.blade.php file, which shows some the skeleton of what the user sees on the page, including the progress bar, and the button containing the first word. 25
- 3.17 This is code from the word_set.JSON file, located in the Public folder. 26
- 3.18 This would be the first segment of word_sets.json would look like if you performed step 3. 26
- 3.19 This would be the first segment of word_sets.json would look like if you performed step 5. 27
- 4.20 This is a screenshot of the part of the view code, which codes for the A and B buttons, as well as the option buttons you see in this web-app. 29
- 4.21 This is a screenshot of the part of the Route file, web.php. Notice at the top, you can see the function which returns the view of the AB web-app. Note that this is not done by calling a controller. 30
- 5.22 This is a screenshot of the ema.blade.php, or the view file for the EMA web-app. Notice the differences between this top part of the view, and the following picture, from later in the same file. 32
- 5.23 This is a screenshot of the further down the ema.blade.php, which shows the HTML elements you see on the screen. 33

- 5.24 This is a screenshot of the web.php file, which shows the route for the EMA web-app 33
- 5.25 This is a screenshot of the ema.json file, which holds all of the content that will be read into the web-app. This is the file you should edit to change the contents of the app. 34
- 5.26 This is a screenshot of the ema.json file, with the last question changed to ask about the participant's age 34
- 5.27 This is a screenshot of the EMA web-app, which shows the updated question 35

Introduction

1.1 What is in this guide?

This version corresponds to Release 2018c. We present instructions on making sure the Embedded Web Server (EWS) is working properly. We provide steps for:

- Testing each web-app to make sure it is working as expected.(Additionally, we provide some documentation on the web-apps.)
- How to customize them

The guide is organized by web app, so that you can easily find the information relevant to your task.

Note: Please refer to the Getting Started Guide to install the EWS software before following the instructions in this guide: <https://github.com/nihospr01/OpenSpeechPlatform-UCSD>

1.2 What's the bigger picture?

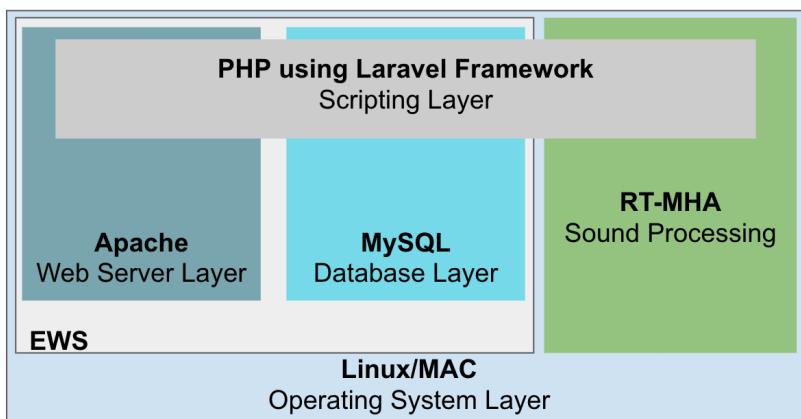


Figure 1.1: Diagram of OSP system. Everything in the white square of this diagram has to do with the embedded webserver (EWS) in this guide. The EWS uses PHP to communicate to the RT-MHA. The EWS literally hosts a webserver, on which users can access webpages or web-apps.

This guide is about the web-apps of the EWS. There is information about testing the web-apps, as well as documentation to help you

make new web-apps. The web-apps themselves are hosted by the EWS, and they use PHP to communicate with the other elements of the OSP. For example, the web-apps use PHP to send data and receive data from the mySQL database. In addition, the web-apps use PHP to communicate parameters to the RT-MHA, which results in real-time audio processing. The web-apps enable researchers and end users to affect the real-time engine, by simply clicking or tapping on elements of a webpage. The web-apps can be accessed from any browser-enabled device. If you are running OSP locally, like on a MacBook, you can use the web-apps by typing in the URL localhost:8000 However, if you are accessing the web-apps from a different device than the one hosting the EWS, say the portable or wearable systems, you must type the IP address of the hosting device into the URL.

1.3 What's under the hood?

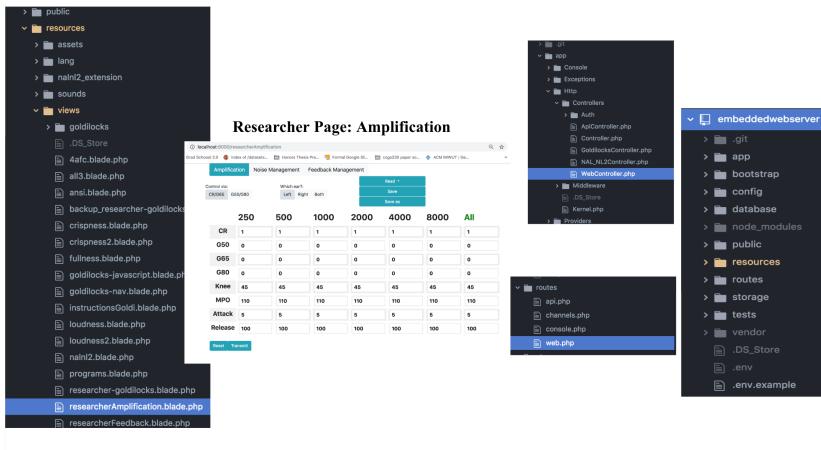


Figure 1.2: Diagram of the file-tree of the EWS, and the specific files involved in the rendering and functionality of the researcher page controlling amplification parameters.

This researcherAmplification.blade.php file located inside of the Resources>Views folder, contains the HTML skeleton of the web-app. It includes aspects of the page such as the tabs at the top, the table, and the labels of the table . This web.php file contains the routing instructions for the web-app. This file outlines which page is associated to which link, and points each page to associated controller. In this file we defined this page link as /researcherAmplification. This WebController.php file contains all of the functions available for this page. This includes the function to view the page, which is called from web.php This file includes the functions which send parameter information to and from the RT-MHA and mySQL databases.

The localhost in the URL refers to this app being served locally. If used remotely, this would be the IP address.

Please be aware that line numbers in code screenshots may not exactly match line numbers in the code provided. Please refer to comments and specified section.

1.4 OSP web-app Landing Page

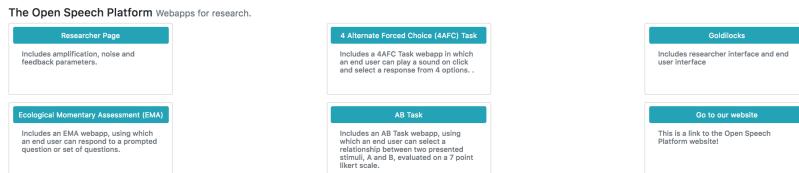


Figure 1.3: Diagram of the file-tree of the EWS, and the specific files involved in the rendering and functionality of the researcher page controlling amplification parameters.

The OSP web-app landing page is the place you end up if you simply type the URL `localhost:8000` or `IPAddress::`. It contains links to all of the different web-apps that we have provided for OSP. It also has a link to our website.

1.4.1 Testing the OSP web-app Landing Page

In order to make sure the OSP web-app landing page is working properly, simply click on the blue "Researcher Page" button. You should be redirected to the researcher page, and see a table of amplification parameters. If the link works, everything is running as it should be and you can proceed.

1.4.2 Helpful tools and tutorials

Most of the time, coding includes Googling a lot of stuff, so we have compiled our most visited websites when creating these web-apps, and have included them below:

1. To get more familiar with Laravel, the framework we are using to enable simple communication between the EWS and the RT-MHA, follow along with these simple tutorials:
2. To get all of the details on Bootstrap, the styling front end framework we are using to create sleek styling which supports multiple form factors, visit this website. Use the menu bar on the left to explore what bootstrap has to offer. As you go through this guide, you can visit this helpful website to learn more about any element used: Note: this particular link will take you to the section about the way in which Boostrap organizes the layout of pages, enabling you to customize content placement. It is good to be familiar with this grid system of rows and cols, so take a few minutes to read this page.

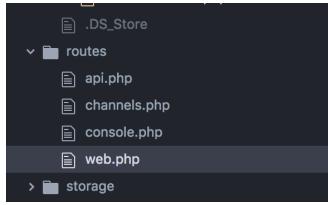
Laravel Tutorials: <https://laracasts.com/series/laravel-from-scratch-2018>

Bootstrap Documentation: <https://getbootstrap.com/docs/4.1/layout/overview/>

3. To learn more about HTML5, the front end skeleton all of the web-apps use, visit this helpful site: (This includes things like the <head> or <style> tags.)

HTML5 Documentation https://www.w3schools.com/html/html_basic.asp

1.4.3 Under the hood of the OSP web-app Landing Page



```

13
14 Route::get('/', function () {
15     return view('welcome');
16 });
17
18 Route::get('goldilocks-nav', function(){
19     return view('goldilocks-nav');
20 });
21

```

Figure 1.4: This is code from the web.php file, located in the Routes folder. You may notice that the code lines in the screenshots do not match the code lines in your download. Do not worry, this is simply the result of comments being added to the code, or things being moved around.

As we discussed in the under the hood section, this page can be broken down into three components:

- The view, which contains the HTML skeleton and some styling elements are located in the Resources>Views>welcome.blade.php file.
- The web.php file, located in the Routes folder, and contains information about the link to this page.(Reference Figure 1.4 and notice lines 14-16.) This code specifies that the interface in the welcome.blade.php file is accessible via the / link. This is the default home direction if no URL is specified. If you click on one of the other web-apps, you notice that the URL reads localhost:8000/nameof.blade.php page, or IPAddress:nameof.blade.php page.
- this page does not have any live updating functionality. It is actually just a list of links to different pages. The controller responsible for this page is called WebController.php, and is located in the app>Http>Controllers folder. The only function that it performs for this page is actually rendering the view. In figure 1.6, pay attention to lines 97-99, which shows the controller code that actually renders the view.

Reference figure 1.5 to look inside the welcome.blade.php file. This excerpt of code that makes the part of the landing page which has the Researcher Page tile. We use Bootstrap 4, an open-sourced front-end framework, which provides a library of stylized elements for designing web apps. For example, on line 55 in figure 1.5, you can see that a class called "card" is called. This is a bootstrap component, which can be easily customized for diverse tasks.

The screenshot shows a file browser interface with a sidebar on the left listing project files and a large text editor window on the right containing the code for `WebController.php`. The sidebar includes sections for `embeddedwebserver`, `app`, `Http`, `Middleware`, `Kernel.php`, `Providers`, `Controllers` (which contains `Auth`, `Goldilocks`, `NALNL2`, `ApiController.php`, and `Controller.php`), and `WebController.php` (which is selected and highlighted in blue). The code in the editor is a PHP script with several methods defined, including `researcherPage`, `researcherAmplificationPage`, `researcherFeedbackPage`, `researcherNoisePage`, `welcomePage`, `ansi`, `load4afc`, and `loadEWA`.

```

Project
└── embeddedwebserver
    ├── .git
    └── app
        ├── Console
        ├── Exceptions
        ├── Http
        │   ├── Controllers
        │   │   ├── Auth
        │   │   ├── Goldilocks
        │   │   ├── NALNL2
        │   │   └── ApiController.php
        │   └── Controller.php
        └── WebController.php
    ├── Middleware
    └── Kernel.php
    ├── Providers
    │   ├── GoldilocksListener.php
    │   ├── GoldilocksLog.php
    │   ├── GoldilocksProgram.php
    │   └── GoldilocksResearcher.php
    ├── Log.php
    ├── User.php
    ├── bootstrap
    ├── config
    ├── database
    ├── public
    ├── resources
    └── routes

WebController.php
61             (int)$request->input('attackTime')[5]
62         ],
63         "releaseTime" => [
64             (int)$request->input('releaseTime')[0],
65             (int)$request->input('releaseTime')[1],
66             (int)$request->input('releaseTime')[2],
67             (int)$request->input('releaseTime')[3],
68             (int)$request->input('releaseTime')[4],
69             (int)$request->input('releaseTime')[5]
70         ]
71     ];
72
73     $socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
74     if(socket_connect($socket, 'localhost', '8080') === true){
75         $action_update_params = ["REQUEST_ACTION" => 1, "VERSION" => 2];
76         $json_update_params = json_encode($action_update_params).json_encode($values);
77         socket_write($socket, $json_update_params, strlen($json_update_params));
78     }
79
80     public function researcherPage(){
81         return view('researcherpage');
82     }
83
84     public function researcherAmplificationPage(){
85         return view('researcher.amplification');
86     }
87
88     public function researcherFeedbackPage(){
89         return view('researcher.feedback');
90     }
91
92     public function researcherNoisePage(){
93         return view('researcher.noise');
94     }
95
96     public function welcomePage(){
97         return view('welcome');
98     }
99
100    public function ansi(){
101        return view('ansi');
102    }
103
104    public function load4afc(){
105        return view('4afc');
106    }
107
108    public function loadEWA(){
109        return view('ewa');
110    }
111 }
112 
```

The screenshot shows a code editor displaying the `welcome.blade.php` file. The code is a single-line HTML template using Bootstrap's card component. It includes a header, a subtitle, and a paragraph, along with a button that links to the researcher amplification page.

```

54     <div class="card" style="width: 23rem; height: 23rem; padding:10px;">
55         <h5 class="card-header">Researcher Page</h5>
56         <div class="card-body">
57             <h6 class="card-subtitle mb-2 text-muted">Includes amplification, noise and feedback parameters</h6>
58             <p class="card-text">You can use this page to send parameters directly to the real-time master</p>
59             <a href="{{ url('/researcherAmplification') }}" class="btn btn-primary">Go to the Researcher Amplification Page</a>
60         </div>
61     </div>
62 
```

You can learn more about cards and other bootstrap elements here:
<https://getbootstrap.com/docs/4.0/components/card/>

To change the name of this card to say something else, you would need to replace the text on line 56 that says Researcher Page. Once you make a change, save it and then re-start the server to make sure that change gets recognized. Line 60 is the part of this card, which contains the link to the researcher page. The syntax of line 60 corresponds to pointing to a URL in the Laravel framework. Laravel is like Bootstrap, but for the PHP side of the web-apps. Laravel is a framework, which comes with ready to use elements that every website needs. For example, Laravel enables us to designate a URL in this simple manner. On the second half of line 80, you notice the classes "btn" and "btn-primary". These are also Bootstrap components, buttons. There is a whole section in the Bootstrap documentation

Figure 1.5: Screenshot of the controller for this web-app. This bottom portion that you see contains the functions which render many of the web-apps. The top portion of this screenshot features the code for pushing the last row of the Amplification page table. Notice that there are several web-apps which receive commands from this controller.

Figure 1.6: This is code from the `welcome.blade.php` file, located in the `Resources>Views` folder. It contains the HTML skeleton of the web-app, as well as styling elements. (Note: You can also follow along in your own development environment, browser or Finder. (note: these screenshots of code are taken in a development environment called Atom. If you download Atom for your computer, you can go to File>Open> and select the directory that you have this code saved in. In other environments, there may be no colors)

website about buttons, and a number of different types to choose from. To change the button color, for example, you could change `btn-primary` to `btn-secondary`, or one of the other options described in the documentation.

The Researcher Page web-app

The researcher page is a tool researchers can use to control the hearing aid. All of the parameters of the real-time master hearing aid (RT-MHA) are exposed, and can be changed in real time. These parameters include controls for amplification, noise management and feedback management. The parameter states can be transmitted in real time to a connected device, creating real time sound processing.

Amplification							Noise Management				Feedback Management			
Control via: G50/G80 : CR/G65 Which ear?: Both / Left / Right AFC: On / Off							Read *							
L250	L500	L1000	L2000	L4000	L8000	L-All	CR	R250	R500	R1000	R2000	R4000	R8000	R-All
1	1	1	1	1	1		G50	0	0	0	0	0	0	
0	0	0	0	0	0		G65	0	0	0	0	0	0	
0	0	0	0	0	0		Knee	45	45	45	45	45	45	
45	45	45	45	45	45		MPO	120	120	120	120	120	120	
120	120	120	120	120	120		Attack	5	5	5	5	5	5	
5	5	5	5	5	5		Release	20	20	20	20	20	20	
20	20	20	20	20	20									

[Undo](#) [Transmit](#)

Figure 2.7: Researcher page web-app, in the amplification parameters section. The url of this page is </researcher/amplification>. Note that the URLs of the noise and feedback management pages are different.

2.5 Testing the Researcher Page

2.5.1 Testing Amplification Parameters

1. Make sure OSP is running
2. Open the Researcher Page web-app in your browser If you are running the web-app on the same device as you are running the OSP, type this URL into your browser:

localhost:8000/researcher/amplification

If you are running the web-app on another device, enter the following URL into the browser of your device:

IP_ADDRESS/researcher/amplification

3. Add at least 5db to each cell in the G65 row. You can use the All column to populate each cell in the row with the same value.
4. Press transmit, and listen to the sound that plays.
5. Now, remove those 5 db from each cell in G65, and listen again. If you hear a difference between the two parameter states, then you know the software is working as intended.

2.5.2 Testing Noise Management Parameters

1. Open the Researcher Page and click on the Noise Management Tab in your browser If you are running the web-app on the same device as you are running the OSP, type this URL into your browser:

localhost:8000/researcher/noise

If you are running the web-app on another device, enter the following URL into the browser of your device:

IP_ADDRESS/researcher/noise

Figure 2.8: Researcher page web-app, in the Noise Management section. The url of this page is /researcher/noise.

2. Click on the circle next to one of the available procedures to select it.
3. Notice whether or not you heard a change in the resulting sound. If you have, this page is connected

2.5.3 Testing Feedback Management Parameters

This page is used to control the RT-MHA's adaptive feedback cancellation module. This page showcases the different options one has with regards to using feedback management. It also features a diagram of where this module fits into the bigger picture of the RT-MHA.

Amplification Noise Management **Feedback Management**

FxLMS
 PMLMS
 SLMS

mu - The step size parameter, which has to be set to a positive value. It controls the adaptation rate of the AFC filter, in the sense that a larger mu is more suitable for a highly changing environment. Note, however, a larger mu also leads to a higher possibility of instability and more artifacts.

rho - A forgetting factor for controlling the update of the signal power estimate. It ranges from 0-1. For speech signals, typical values of rho are around 0.9. If it is too large, the update of the power estimate would fail to catch up with speech variations. If it is set too small, the variance of the estimate will be high, leading to more artifacts.

Troubleshooting 4AFC
When instability is detected by the user, he/she should press the "Undo" button below. This will (i) initialize AFC filter to its initial coefficients, obtained from average of multiple feedback path measurements; and (ii) reduce amplification gain in all bands by x dB. (Arthur to experiment and suggest value of x. Initially, x=10).

Undo Transmit

```

graph LR
    mic((mic)) -- "mic input signal" --> sum(( ))
    sum -- "-" --> compensated[feedback-com pensated signal]
    compensated --> HAMP[Hearing Aid Main Processing]
    HAMP -- "hearing aid output signal" --> receiver((receiver))
    HAMP -- "feedback signal estimate" --> Adap[Adaptation parameters: (\mu, \rho)]
    Adap -- "updated filter taps" --> AFC[AFC Filter]
    AFC -- " " --> Adap
  
```

Figure 2.9: Researcher page web-app, in the feedback parameters section. The url of this page is /researcher/feedback

1. Open the Researcher Page and click on the Noise Management Tab in your browser

If you are running the web-app on the same device as you are running the OSP, type this URL into your browser:

localhost:8000/researcher/feedback

If you are running the web-app on another device, enter the following URL into the browser of your device:

IP_ADDRESS/researcher/feedback

2. Click on the circle next to one of the top three lines to select it.

3. Notice whether or not you heard a change in the resulting sound. If you have, this page is connected

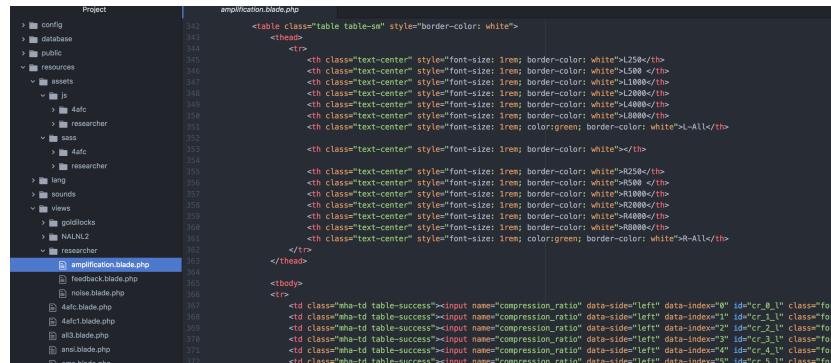
2.6 Researcher Page Documentation

2.6.1 Amplification Parameters

This section includes screenshots of the code that makes the amplification parameters page work.

Recall the components of a web page we discussed in the under the hood section.

1. **The view**, which contains the HTML skeleton and some styling elements is located in the Resources>Views>researcher/amplification.blade.php file. The top of the amplification parameters page includes the navigation bar, which displays button links to the other researcher pages, as well as the control via, save, save as and read buttons. The bottom of the amplification parameters page contains two tables: one for the left ear and one for the right ear, as well as the reset and transmit buttons. In the rest of this section, I will be referring to the views by either saying top or bottom.



```

Project
> config
> database
> public
> resources
  > assets
    > is
      > 4fc
      > researcher
    > sass
      > 4fc
      > researcher
  > lang
  > sounds
  > views
    > goldilocks
    > NALNI.2
    > researcher
  > researcher
amplification.blade.php
  <table class="table table-sm" style="border-color: white">
    <thead>
      <tr>
        <th class="text-center" style="font-size: 1rem; border-color: white">L-1</th>
        <th class="text-center" style="font-size: 1rem; border-color: white">L-2</th>
        <th class="text-center" style="font-size: 1rem; border-color: white">L-3</th>
        <th class="text-center" style="font-size: 1rem; border-color: white">L-4</th>
        <th class="text-center" style="font-size: 1rem; border-color: white">R-1</th>
        <th class="text-center" style="font-size: 1rem; border-color: white">R-2</th>
        <th class="text-center" style="font-size: 1rem; border-color: white">R-3</th>
        <th class="text-center" style="font-size: 1rem; border-color: white">R-4</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td class="w-100" data-side="left" data-index="0" id="cr_0_1" data-table="success"><input name="compression_ratio" type="text" value="0.5" /></td>
        <td class="w-100" data-side="left" data-index="1" id="cr_1_1" data-table="success"><input name="compression_ratio" type="text" value="0.5" /></td>
        <td class="w-100" data-side="left" data-index="2" id="cr_2_1" data-table="success"><input name="compression_ratio" type="text" value="0.5" /></td>
        <td class="w-100" data-side="left" data-index="3" id="cr_3_1" data-table="success"><input name="compression_ratio" type="text" value="0.5" /></td>
        <td class="w-100" data-side="right" data-index="0" id="cr_0_2" data-table="success"><input name="compression_ratio" type="text" value="0.5" /></td>
        <td class="w-100" data-side="right" data-index="1" id="cr_1_2" data-table="success"><input name="compression_ratio" type="text" value="0.5" /></td>
        <td class="w-100" data-side="right" data-index="2" id="cr_2_2" data-table="success"><input name="compression_ratio" type="text" value="0.5" /></td>
        <td class="w-100" data-side="right" data-index="3" id="cr_3_2" data-table="success"><input name="compression_ratio" type="text" value="0.5" /></td>
      </tr>
    </tbody>
  </table>

```

Figure 2.10: This is a screenshot of the start of the table in the bottom of the amplification parameters page. You may notice that the code lines in the screenshots do not match the code lines in your download. Do not worry, this is simply the result of comments being added to the code, or things being moved around.

In the image above, you can see the HTML code to make the table in the bottom of the amplification parameters page. Look at line 242 in this screenshot to see how a table is initialized using Bootstrap. We create a div, and then call the class table, and table-sm. These both refer to the table class, but the sm makes the table adjust when presented on a mobile screen.

The `<thead>` denotes the start of the head of the table. Note that it is essential to close the head with a `</thead>` as you can see in the bottom line of the screenshot. Then the `<tr>` element denotes the start of a table row. Notice how there many lines labeled `<th>` until the closing of this row with the `</tr>`. Each `<th>` specifies what the head of that column will be called.

You can always visit: <https://getbootstrap.com/docs/4.0/content/tables/> to learn more about specifics about the table class.

Notice that once <tbody> starts, the columns are referred to using the <td> element instead. Here is an exact copy of first column in the first row, the Compression Ratio:

```
<td class="mba-td table-success"><input name="compression_ratio" data-side="left" data-index="0" id="cr_o_1" class="form-control form-control-sm table-field font-weight-bold mba-input-field" type="number" value="1"></td>
```

Notice all of the many classes in use to make this table possible. Pay particular attention to the id, which is cr_o_1. This stands for

2. **The route** information for this web-app can be found in the web.php file, located in the Routes folder, and contains information about the link to this page. Reference Figure 2.10, and notice lines 19, which specifies the route for the amplification parameters page. It makes a call to the web controller, to access the researcher amplification page. Notice that the other two components of the researcher web-app: the noise and feedback management pages, have their own routes.



The screenshot shows a code editor with a sidebar labeled "Project" containing project files like embeddedwebserver, .git, app, bootstrap, config, database, public, resources, routes, api.php, channels.php, and console.php. The main pane shows the content of the file "web.php". The code is as follows:

```

Project
[embeddedwebserver
[.git
[app
[bootstrap
[config
[database
[public
[resources
[routes
[api.php
[channels.php
[console.php
[web.php
[storage
[tests
[vendor

web.php

3  /*
4  |
5  | Web Routes
6  |
7  |
8  | Here is where you can register web routes for your application. These
9  | routes are loaded by the RouteServiceProvider within a group which
10 | contains the "web" middleware group. Now create something great!
11 |
12 */
13
14 Route::get('/', 'WebController@welcomePage');
15 Route::get('ansi', 'WebController@ansiPage');
16
17
18 /** Researcher routes */
19 Route::get('researcher/amplification', 'WebController@researcherAmplificationPage');
20 Route::get('researcher/feedback', 'WebController@researcherFeedbackPage');
21 Route::get('researcher/noise', 'WebController@researcherNoisePage');
22

```

Figure 2.11: This is a screenshot of the code for the researcher page routes found in web.php. Notice that the route for the amplification parameters page is the first line under the comment "Researcher Routes"

3. **The controller** on this page is responsible for the live updating functionality. The controller for the whole researcher page is called WebController.php, and is located in the app>Http>Controllers folder.

You can see that the first main function of this file starts on line 17 in Figure 2.11, and is called "updateParams". This is a request, meaning it must be initiated. In the amplification parameters page, the request is initiated when a user presses "Transmit".

Notice on line 23 in the Figure 2.11 that it asks for the "g50" values. In the lines of code before the "g65" label, you see requests to return integer values of the input in the g50 row. The numbers in orange, 0-5, correspond to the columns. As such, lines 23-29 in this screenshot can get the input from the g50 row on the amplification parameters page.

The screenshot shows a file browser interface with a sidebar on the left listing project files and a main pane on the right displaying the code of `WebController.php`. The sidebar includes files like `embeddedwebserver`, `app`, `Http`, `Providers`, `bootstrap`, `config`, and `database`. The main pane shows the following PHP code:

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class WebController extends Controller
8 {
9
10     /**
11      * Updates the parameters of the MIA as modified in the web client via web socket.
12      *
13      * @param Request $request The POST request data
14      *
15      * @return string
16      */
17     public function updateParams(Request $request){
18
19         $values = [
20             "nopl" => 0,
21             "afc" => 1,
22             "feedback" => 1,
23             "rear" => 1,
24             "g50" => [
25                 (int)$request->input('g50')[0],
26                 (int)$request->input('g50')[1],
27                 (int)$request->input('g50')[2],
28                 (int)$request->input('g50')[3],
29                 (int)$request->input('g50')[4],
30                 (int)$request->input('g50')[5]
31             ],
32             "g80" => [
33                 (int)$request->input('g80')[0],
34                 (int)$request->input('g80')[1],
35                 (int)$request->input('g80')[2],
36                 (int)$request->input('g80')[3],
37                 (int)$request->input('g80')[4],
38                 (int)$request->input('g80')[5]
39             ],
40             "kneelow" => [
41                 (int)$request->input('kneelow')[0],
42                 (int)$request->input('kneelow')[1],
43                 (int)$request->input('kneelow')[2],
44                 (int)$request->input('kneelow')[3],
45                 (int)$request->input('kneelow')[4],
46                 (int)$request->input('kneelow')[5]
47             ],
48             "mpoLimit" => [
49                 (int)$request->input('mpoLimit')[0]
50             ]
51         ];
52
53         return json_encode($values);
54     }
55 }

```

Figure 2.12: This is a screenshot of the controller code for this web-app. You can see

2.6.2 Noise Management Parameters

This section includes screenshots of the code that makes the noise management parameters page work. Recall the components of a web-page we discussed in the under the hood section.

The screenshot shows a file browser interface with a sidebar on the left listing project files and a main pane on the right displaying the content of `feedback.blade.php`. The sidebar includes files like `NALNL2`, `researcher`, `amplification.blade.php`, `noise.blade.php`, `4afc.blade.php`, `4afc1.blade.php`, `all3.blade.php`, `ansi.blade.php`, `ema.blade.php`, `goldlocks-javascript.blade.php`, `researcherpage.blade.php`, `start.blade.php`, and `welcome.blade.php`. The main pane shows the following blade template code:

```

27 <div class="container" style="margin-top: 10px">
28
29     <nav class="nav nav-pills nav-justified" style="margin-bottom: 10px">
30         <a class="nav-item nav-link text-info bg-white" href="#">util</a>
31         <a class="nav-item nav-link text-info bg-white" href="#">Amplification</a>
32         <a class="nav-item nav-link text-white bg-info" href="#">Noise Management</a>
33     </nav>
34
35     <div class="form-check">
36         <input class="form-check-input" type="radio" name="afc" id="afc_1" value="1">
37         <label class="form-check-label" for="afc_1">
38             FxMS
39         </label>
40     </div>
41     <div class="form-check">
42         <input class="form-check-input" type="radio" name="afc" id="afc_2" value="2">
43         <label class="form-check-label" for="afc_2">
44             FreqMS
45         </label>
46     </div>
47     <div class="form-check">
48         <input class="form-check-input" type="radio" name="afc" id="afc_3" value="3">
49         <label class="form-check-label" for="afc_3">
50             SNS
51         </label>
52     </div>
53
54     </div>
55
56     <form>
57         <div class="form-group">
58             <label for="stepSize"><strong>n</strong> - The step size parameter, which has to be set to a positive value. It controls
59             <input type="number" class="form-control" id="n" value="0">
60         </div>
61         <div class="form-group">
62             <label for="rho"><strong>rho</strong>- A forgetting factor for controlling the update of the signal power estimate
63             <input type="number" class="form-control" id="rho" value="0">
64         </div>
65     </form>

```

Figure 2.13: Screenshot of the code of the `feedback.blade.php` file, which contains the content that is used to visualize the feedback management web-app, also called the view.

1. **The View**, which contains the HTML skeleton and some styling elements is located in the `Resources>Views>researcher/noise.blade.php` file. This is where you can enter the text that you want to appear on a page, and describe any buttons and what they do. You can see in Figure 2.12, lines 41-45 create the navigation bar you use

in the interface to go between the three pages of the researcher web-app. Notice that line 42 and 44 include links to the respective pages, using the same syntax we saw in the welcome page links.

You can think of each component of the page being wrapped in its own div. For example, lines 48-53 wrap up the input for the Arslan power averaging procedure. Line 48 starts the div, and establishes that this will be a check form. Then, in line 49, the expected input is specified, it will be a radio type of input, indicated by a black dot in a circle, when selected. Lines 50-52 specify the "label" for this field in the form, and that is where you see the text "Arslan power averaging procedure", which appears on the page. You can see lines 54-59, 60-65 and 66-71 in this photo follow this pattern for the other options in the noise management parameters page.

2. **The route** information for this web-app can be found in the web.php file, is located in the Routes folder. It contains information about the link to this page. Reference Figure 2.10 and notice line 21, which specifies the route for the noise management parameters page. It makes a call to the web controller, to access the noise management page.
3. **The Controller** is located in the app>Http>WebController.php file. As described in the route section above, the route asks the controller to render this page in the routes. The actual function which renders the page lives in the controller, along with the other functions that encode the page's functionality.

2.6.3 *Feedback Management Parameters*

This section includes documentation and code snippets from the feedback management page. It will also be broken down into the components: view, route, controller.

1. **The View** file for this page, feedback.blade.php is located in resources>views>researcher, along with noise.blade.php and amplification.blade.php. Notice in figure 2.13, you see the same code to define the nav bar, only this time, there are links for the amplification and noise pages.

Notice that the structure of this page is very similar to that of the noise management page. In particular, look at lines 47-52 in figure 2.13. You see the opening div with the form-check, followed by the line which specifies the input type as radio, followed by the label for the field, and ended by the closing field. This pattern repeats for the other options of AFC.

The screenshot shows a file browser interface with a sidebar on the left listing files and folders. In the main area, the content of the file 'feedback.blade.php' is displayed. The code is a PHP blade template with some CSS and HTML. It includes navigation links for 'Amplification', 'Noise Management', and 'Feedback Management'. Below this, there are several form sections. One section contains three radio buttons labeled 'FMS', 'SUMS', and 'LMS'. Another section contains two input fields labeled 'mu' and 'rho'. The code uses Bootstrap classes like 'form-group' and 'form-control'.

```

Project
  > NALNL2
  > researcher
    > amplification.blade.php
    > feedback.blade.php
      <-- feedback.blade.php
    > noise.blade.php
    > 4afc1.blade.php
    > all3.blade.php
    > ansi.blade.php
    > ema.blade.php
    > goldilocks-javascript.blade.php
    > researcherpage.blade.php
    > start.blade.php
    > welcome.blade.php
  > routes
  > storage
  > tests
  > vendor
    > .env
    > .env.example
  > .gitattributes

feedback.blade.php

1 <div class="container" style="margin-top: 10px">
2   <nav class="nav" style="margin-bottom: 10px">
3     <a class="nav-item nav-link text-info bg-white href="#/researcher/amplification">Amplification</a>
4     <a class="nav-item nav-link text-white bg-info href="#/researcher/noise">Noise Management</a>
5     <a class="nav-item nav-link text-white bg-info href="#/feedbackManagement">Feedback Management</a>
6   </nav>
7
8   <div class="form-check">
9     <input class="form-check-input" type="radio" name="afc" id="afc_1" value="1">
10    <label class="form-check-label" for="afc_1">
11      FMS
12    </label>
13  </div>
14  <div class="form-check">
15    <input class="form-check-input" type="radio" name="afc" id="afc_2" value="2">
16    <label class="form-check-label" for="afc_2">
17      SUMS
18    </label>
19  </div>
20  <div class="form-check">
21    <input class="form-check-input" type="radio" name="afc" id="afc_3" value="3">
22    <label class="form-check-label" for="afc_3">
23      LMS
24    </label>
25  </div>
26  <form>
27    <div class="form-group">
28      <label for="mu"><strong>mu</strong> – The step size parameter, which has to be set to a positive value. It controls
29      <input type="number" class="form-control" id="mu" value="0">
30    </div>
31    <div class="form-group">
32      <label for="rho"><strong>rho</strong> – A forgetting factor for controlling the update of the signal power estimate
33      <input type="number" class="form-control" id="rho" value="0">
34    </div>
35  </form>

```

Figure 2.14: Researcher page web-app, in the amplification parameters section. The url of this page is /researcherAmplification Note that the URLs of the noise and feedback management pages are different.

Find the tag `<form>`, on line 66 in figure 2.13, and notice the similarities and differences between a form which has a check input, and a form which has a character input. This section encodes the mu and rho parameters, which users can enter by typing in numbers. As such, this is a basic and useful type of input which is good to be familiar with. To learn more about making forms with the bootstrap framework we are using, visit this website :

<https://getbootstrap.com/docs/4.0/components/forms/>

2. **The Route** information for this page can be found in the `web.php` file, located in the `Routes` folder. Recall that this is the same file in which the other routes are, and reference figure 2.10. Line 20 in this image defines the route for the feedback management page, by calling the controller which contains the function which renders the page.
3. **The Controller** is located in the `app>HTTP>WebController.php` file. It is the same controller as the other pages of the researcher web-app. As described in the route section above, the route asks the controller to render this page in the routes. The actual function which renders the page lives in the controller, along with the other functions that encode the page's functionality.

4 Alternate Forced Choice (4AFC) web-app

The 4AFC web-app is a tool to measure outcomes, which allows researchers to run four choice question tasks using live sound processing, or canned stimuli.

This web-app is still in progress, and will soon be connected to the real time audio functionality. Please feel free to look at the user interface, and let us know if you have suggestions on improving the behavior flow.

3.1 Testing the 4AFC web-app

1. Make sure OSP is running.
2. **Open the 4afc web-app in your browser** If you are running the web-app on the same device as you are running the OSP, type this URL into your browser:

localhost:8000/4afc

If you are running the web-app on another device, enter the following URL into the browser of your device:

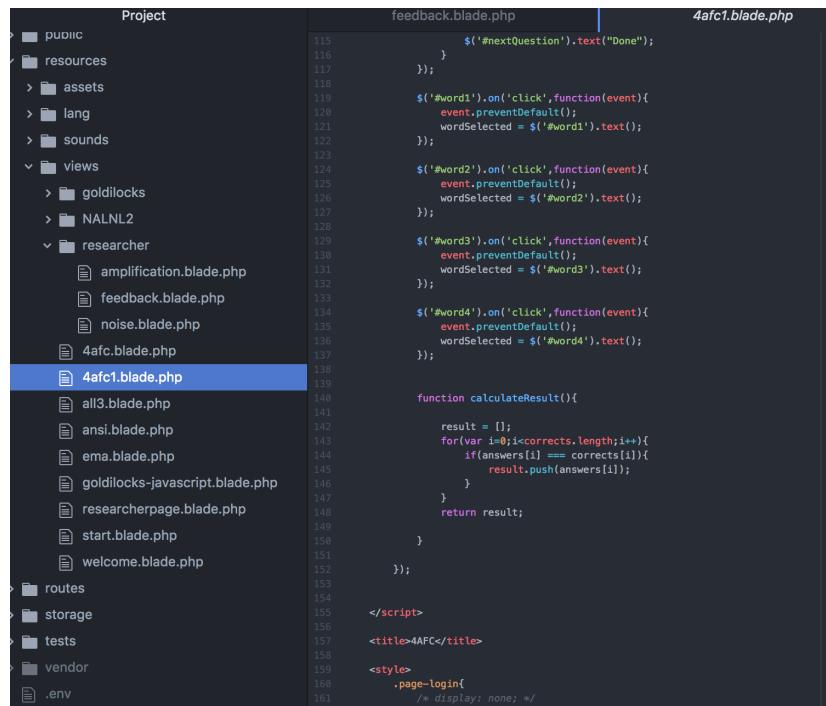
IP_ADDRESS/4afc

3. **Click on each of the four options in the first question** and confirm that the sound that plays is in fact the word that is written. If the words are playing correctly, the app is working properly. To learn how to edit the contents, see the documentation section below.

3.2 4AFC Documentation

In this section we will go over the three primary components we have been discussing: the view, route and controller, and introduce a special format for storing data to be read into a web page.

1. The View for this page, 4afc1.blade.php, can be found in the resources>views folder. What is important to understand about this web-app, is that each question is not an entirely new web page, but rather, the contents of the questions are dynamically read in, on click, from the word_sets.json file (Figure 3.15) in the public folder.



The screenshot shows a file explorer interface with a sidebar on the left listing project files and folders. The main area displays the content of the 4afc1.blade.php file. The code is a combination of HTML and JavaScript. It includes event listeners for four buttons labeled '#word1' through '#word4'. Each button's click event triggers a function that prevents the default action and updates a variable 'wordSelected' to the button's text value. Below these, there is a 'calculateResult' function that iterates through an array of answers and a corresponding array of correct answers, pushing matching answers into a result array. Finally, the code ends with a closing script tag, a title tag containing '4AFC', and a style tag with a class 'page-login' and a note about its display being set to none.

```

Project
  - public
  - resources
    - assets
    - lang
    - sounds
  - views
    - goldilocks
    - NALNL2
    - researcher
      - amplification.blade.php
      - feedback.blade.php
      - noise.blade.php
      - 4afc1.blade.php
      - 4afc1.blade.php (selected)
        - all3.blade.php
        - ansi.blade.php
        - ema.blade.php
        - goldilocks-javascript.blade.php
        - researcherpage.blade.php
        - start.blade.php
        - welcome.blade.php
    - routes
    - storage
    - tests
    - vendor
    - .env

feedback.blade.php | 4afc1.blade.php
115     $('#nextQuestion').text("Done");
116   });
117 });
118 $('#word1').on('click',function(event){
119   event.preventDefault();
120   wordSelected = $('#word1').text();
121 });
122 $('#word2').on('click',function(event){
123   event.preventDefault();
124   wordSelected = $('#word2').text();
125 });
126 $('#word3').on('click',function(event){
127   event.preventDefault();
128   wordSelected = $('#word3').text();
129 });
130 $('#word4').on('click',function(event){
131   event.preventDefault();
132   wordSelected = $('#word4').text();
133 });
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
function calculateResult(){
  result = [];
  for(var i=0;i<corrects.length;i++){
    if(answers[i] === corrects[i]){
      result.push(answers[i]);
    }
  }
  return result;
}
</script>
<title>4AFC</title>
<style>
  .page-login{
    /* display: none; */
  }

```

Figure 3.15: This is code from the 4afc1.blade.php file, which shows some of the JavaScript functionality, which contributes to the buttons being able to update with each question. .

Reference figure 3.14, and notice that all of this code is happening within the <head> html tag. You can see a <script> tag, which means that the code until the </script> tag will be in JavaScript, or another non-HTML language. The <style> tag, means that the code until the </style> tag is CSS styling code, which can be used to define the way content looks. It is also possible to add style directly into the HTML, which you can see in figure 2.9. Each row heading is styled to a certain font size.

Now, reference figure 3.15 to see the code to create the elements you see on the page. Notice the first section is a way to include a

login screen directly into the web-app. Following down the page, notice the code for the progress bar. This is a bootstrap class we use to create the look and feel of a dynamic web-app.

Towards the bottom of figure 3.15, you can see # word1 and # word2 being loaded into btn elements. This is how the words choices appear in the buttons on the 4afc web-app.

You can easily change the number of questions the app displays, or the contents of the questions using the steps provided next, without changing the styling aspects of the page. This will make it very simple for you to customize this web-app for your needs.

```

Project
└── embeddedwebserver
    ├── .git
    ├── app
    ├── bootstrap
    ├── config
    ├── database
    └── public
        ├── apigen
        ├── css
        ├── fonts
        ├── js
        │   └── .htaccess
        ├── 4afcBlock.png
        ├── ansi2.png
        ├── favicon.ico
        ├── index.php
        ├── mix-manifest.json
        ├── robots.txt
        ├── web.config
        └── word_sets.json
    └── resources
        ├── assets
        ├── lang
        ├── sounds
        └── views
            └── feedback.blade.php
                219 <body>
                220     <section class="App">
                221         <div class="container-login">
                222             <h3 id="AppTitle">4afc Web App</h3>
                223             <div class="form-group">
                224                 <input class="form-control" type="text" placeholder="Tester ID" id="testerID">
                225                 <input class="form-control" type="text" placeholder="Participant ID" id="participantID">
                226             </div>
                227             <button type="button" class="btn btn-primary" id="Login">Log In</button>
                228         </div>
                229     </section>
                230
                231     <section class="page-question">
                232
                233         <div class="container-questions">
                234             <h4 class="AppTitle">4afc Web App</h4>
                235             <h5>Press play and select the word you hear</h5>
                236             <div class="progress">
                237                 <div class="progress-bar progress-bar-striped progress-bar-animated" role="progressbar" aria-valuemin="0" aria-valuemax="100" id="progressBar">
                238                     <div style="width: 0%;></div>
                239                 </div>
                240             </div>
                241             <div class="description">
                242                 Press play and select the word you hear
                243             </div>
                244             <div class="container-questions">
                245                 <div class="row">
                246                     <div class="col-sm-6">
                247                         <button type="button" class="btn btn-outline-primary btn-lg btn-block btn-margin" id="word1">
                248                             word1
                249                         </button>
                250                     </div>
                251                     <div class="col-sm-6">
                252                         <button type="button" class="btn btn-outline-primary btn-lg btn-block btn-margin" id="word2">
                253                             word2
                254                         </button>
                255                     </div>
                256                 </div>
                257             </div>
                258         </div>
                259     </div>
                260
                261
                262
                263
                264
            </div>
        </div>
    </div>
</body>

```

You can find more information about Bootstrap classes including progress on: <https://getbootstrap.com/docs/4.0/components/progress/>

To learn more about the JavaScript used to actually make the words appear, check out: https://www.w3schools.com/js/js_json_html.asp

Figure 3.16: This is code from the 4afc1.blade.php file, which shows some the skeleton of what the user sees on the page, including the progress bar, and the button containing the first word.

```

Project
└── public
    └── word_sets.json
        [1] {
        [2]     "words": ["states", "states", "state", "sate"],
        [3]     "correct_answer": "states",
        [4]     "audio": { "Word_Set": "./../../resources/sounds/01/", "Actual_answer": "./../../resources/sounds/01/01.wav" }
        [5] },
        [6] {
        [7]     "words": ["peak", "teak", "peat", "teat"],
        [8]     "correct_answer": "teak",
        [9]     "audio": { "Word_Set": "./../../resources/sounds/02/", "Actual_answer": "./../../resources/sounds/02/02.wav" }
        [10],
        [11] },
        [12] {
        [13]     "words": ["base", "pace", "bait", "pate"],
        [14]     "correct_answer": "bait",
        [15]     "audio": { "Word_Set": "./../../resources/sounds/03/", "Actual_answer": "./../../resources/sounds/03/03.wav" }
        [16],
        [17] },
        [18] {
        [19]     "words": ["mat", "bat", "mad", "bad"],
        [20]     "correct_answer": "bad",
        [21]     "audio": { "Word_Set": "./../../resources/sounds/04/", "Actual_answer": "./../../resources/sounds/04/04.wav" }
        [22]
    }

```

Figure 3.17: This is code from the word_set.JSON file, located in the Public folder.

- (c) To edit the existing questions, replace the text in the word_sets.json file, that appears in green on the image below For example, you could change the first question to have the options: add bad cad dad; pick bad as the correct answer and your resulting JSON segment would look as follows:

```

Project
└── public
    └── word_sets.json
        [1] [
        [2] {
        [3]     "words": ["add", "bad", "cad", "dad"],
        [4]     "correct_answer": "bad",
        [5]     "audio": { "Word_Set": "./../../resources/sounds/01/", "Actual_answer": "./../../resources/sounds/01/01.wav" }
        [6] },

```

Figure 3.18: This would be the first segment of word_sets.json would look like if you performed step 3.

- (d) In order to change the audio files first, organize your audio files such that there is a folder for each question, and the four options are inside of the folder. Second, copy and paste these folders of sound files into the Resources>Sounds folder. Second, change the third line of each segment in the JSON file, named "audio", to reflect your sound files. For example, say you added a folder of sounds called "example", and inside it contained four sound files: 01, 02, 03, and 04. Let's say the correct answer was "03". This is what the "audio" line of code would look like:

```
"audio": { "Word_Set": "./../../resources/sounds/example/", "Actual_answer": "./../../resources/sounds/example/03.wav" }
```

- (e) To add a new question, copy and paste the code for an existing segment, and replace its contents with desirable ones. For example, we could add a fifth question, by copying and pasting the code for question one, after question four. Then, we need to change the "words" and "correct_answer" fields for the new question. Let's say the audio files you added to Resources>Sounds in step 4, are the correct sound files for this fifth question. Then, the whole JSON for all 5 questions would look like this:

Note: Notice that line 21 now has a comma after it, and there is no comma after line 26. This is because line 26 is the end of



```

Project
  - embeddedwebserver
    - .git
    - app
    - bootstrap
    - config
    - database
    - public
      - assets
        - asigen
        - css
        - fonts
        - js
          - .htaccess
          - 404Block.png
          - asnd2.png
          - favicon.ico
          - index.php
          - mix-manifest.json
          - robots.txt
          - web.config
    - resources
      - assets
        - word_sets.json
  - word_sets.json

```

```

1  [
2   {
3     "words": ["add", "bad", "cad", "dad"],
4     "correct_answer": "bad",
5     "audio": { "Word_set": "../../../../resources/sounds/01/", "Actual_answer": "../../../../resources/sounds/01/01.wav" }
6   },
7   {
8     "words": ["peak", "teak", "peat", "teat"],
9     "correct_answer": "teak",
10    "audio": { "Word_set": "../../../../resources/sounds/02/", "Actual_answer": "../../../../resources/sounds/02/02.wav" }
11  },
12  {
13    "words": ["base", "pace", "bait", "pate"],
14    "correct_answer": "bait",
15    "audio": { "Word_set": "../../../../resources/sounds/03/", "Actual_answer": "../../../../resources/sounds/03/03.wav" }
16  },
17  {
18    "words": ["matt", "bat", "mad", "bad"],
19    "correct_answer": "bad",
20    "audio": { "Word_set": "../../../../resources/sounds/04/", "Actual_answer": "../../../../resources/sounds/04/04.wav" }
21  },
22  {
23    "words": ["states", "states", "state", "sate"],
24    "correct_answer": "states",
25    "audio": { "Word_set": "../../../../resources/sounds/example/", "Actual_answer": "../../../../resources/sounds/example/03.wav" }
26  }
27 ]

```

Figure 3.19: This would be the first segment of word_sets.json would look like if you performed step 5.

the last segment in this file, so no comma is needed. However, between segments, it is essential to have a comma, as demonstrated in lines 6, 11, 16 and 21.

2. **The Route** information for this page is in the web.php file, in the Routes folder.
3. **The Controller** for this page is WebController.php, and is located in the app>Http>Controllers folder. The only function the controller has for this web-app, is the one that renders the view of the page.

AB Test web-app

The AB test web-app is a tool to measure outcomes, which researchers can use to run AB comparison tests between two particular MHA setting combinations. Users respond using a 7 point likert scale. Using this app, you can AB test different parameter "prescriptions", and reach the best prescription for the end listener. If you are using the portable version of our system, you can also deploy an AB test in real time, and the users can complete them in the wild.

This web-app is still in progress, and will soon be connected to the real time audio functionality. Please feel free to look at the user interface, and let us know if you have suggestions on improving the behavior flow

4.1 *Testing the AB Test web-app*

1. Make sure OSP is running.
2. Open the AB test web-app in your browser If you are running the web-app on the same device as you are running the OSP, type this URL into your browser:

localhost:8000/ab

If you are running the web-app on another device, enter the following URL into the browser of your device:

IP_ADDRESS/ab

3. Click on the play button indicated by a triangle in a box in the upper right hand corner.
4. Click on A, and listen to the output
5. Then, click on B and listen to the output You may have to repeat steps 3-5 a couple of times to get the hang of it.

6. Make sure that you can hear a difference in the sound you hear after you press A or B If you hear a difference, the AB web-app is working properly. To learn how to customize the sounds and parameters of the AB web-app, read the documentation section below .

4.2 AB Test Documentation

1. The View file for this web-app is called ab.blade.php, and is located in the resources>views folder. You can see many similar elements from the other outcomes apps, such as the progress bar, and results table at the end.

```

Project
├── ema.json
├── favicon.ico
├── index.php
├── mix-manifest.json
├── robots.txt
├── web.config
└── word_sets.json
  └── resources
    ├── assets
    ├── lang
    ├── sounds
    └── views
      ├── goldilocks
      ├── NALNL2
      ├── researcher
      ├── 4afc.blade.php
      ├── 4afc1.blade.php
      └── ab.blade.php
ab.blade.php
  159 <div> PULL AN AB TEST FROM YOUR ANALYTICS
  160 </div>
  161
  162 <div class="row">
  163   <div class="col" style="text-align: center;">
  164     <button type="button" id = "A" class="btn btn-default btn-play">
  165       <span class="oi oi-play-circle oi-play"></span>
  166     </button>
  167     <span>A</span>
  168   </div>
  169
  170   <div class="col" style="text-align: center;">
  171     <button type="button" id = "B" class="btn btn-outline-primary btn-lg btn-block">
  172       <span class="oi oi-play-circle oi-play"></span>
  173     </button>
  174     <h3>B</h3>
  175   </div>
  176
  177 </div>
  178
  179 <div class="container-choices">
  180   <div class="grid-choices">
  181     <div class="row">
  182       <div class="col-sm-6">
  183         <button type="button" class="btn btn-outline-primary btn-lg btn-block">A >> B</button>
  184         <button type="button" class="btn btn-outline-primary btn-lg btn-block">A >> B</button>
  185         <button type="button" class="btn btn-outline-primary btn-lg btn-block">A > B</button>
  186       </div>
  187       <div class="col-sm-6">
  188         <button type="button" class="btn btn-outline-primary btn-lg btn-block">A = B</button>
  189       </div>
  190     </div>
  191   </div>
  192
  193 <div class="col-sm-6">
  194   <button type="button" class="btn btn-outline-primary btn-lg btn-block">B >> A</button>
  195   <button type="button" class="btn btn-outline-primary btn-lg btn-block">B >> A</button>
  196   <button type="button" class="btn btn-outline-primary btn-lg btn-block">B > A</button>
  197 </div>
  198
  199 </div>

```

Figure 4.20: This is a screenshot of the part of the view code, which codes for the A and B buttons, as well as the option buttons you see in this web-app.

2. The Route information for this web-app is located in the web.php file. You can see that the controller, webController is being called to run the function which renders the webpage.
3. The Controller for this web-app is called has not been implemented yet, as the routes page contains the function to render the page.

The screenshot shows a file browser interface with a sidebar on the left and the content of a file on the right.

Project Structure:

- embeddedwebserver
 - .git
 - app
 - bootstrap
 - config
 - database
 - public
 - resources
 - routes
 - api.php
 - channels.php
 - console.php
 - web.php** (highlighted)
 - storage

Content of web.php:

```

20 Route::get('ab', function () {
21     return view('ab');
22 });
23 });
24
25 Route::get('goldilocks-nav', function(){
26     return view('goldilocks-nav');
27 });
28
29 /**
30  * Researcher routes */
31 Route::get('researcher/amplification', 'WebController@researcherAmplificationPage');
32 Route::get('researcher/feedback', 'WebController@researcherFeedbackPage');
33 Route::get('researcher/noise', 'WebController@researcherNoisePage');
34
35 Route::get('researcherpage', 'WebController@researcherPage');
36 Route::post('researcherpage', 'WebController@updateParams');
37
38 /**
39  * na1-nl2 routes */
40 Route::get('na1-nl2', 'NALNL2\NALNL2Controller@loadPage');
41 Route::post('na1-nl2', 'NALNL2\NALNL2Controller@getParameters');
42 Route::post('na1-nl2update', 'NALNL2\NALNL2Controller@updateParameters');
43
44 /**
45  * 4afc web app route */
46 Route::get('4afc', 'WebController@load4AFC');

```

Figure 4.21: This is a screenshot of the part of the Route file, web.php. Notice at the top, you can see the function which returns the view of the AB web-app. Note that this is not done by calling a controller.

Environmental Momentary Assessment (EMA) web-app

This web-app is a tool for sampling the listener's mental state when they had a problem with the system so researchers have additional information to improve the hearing aid process. The questions can be easily changed to meet the needs of any study. One could even apply machine learning to this tool to auto-adjust the hearing aid. If you are using the portable version of our system, you can also deploy these assessments in real time, and the users can complete them in the wild.

5.1 Testing the EMA web-app

1. Make sure OSP is running.
2. **Open the EMA web-app in your browser** If you are running the web-app on the same device as you are running the OSP, type this URL into your browser:

localhost:8000\ema

If you are running the web-app on another device, enter the following URL into the browser of your device:

IP_ADDRESS\ema

3. **Go through the questions, select an answer, and then go onto the next question** At the end, you will come to a summary table of your answers. If you get to this page, your web-app is working! To learn how to edit the questions, see section 5.2 EMA Documentation

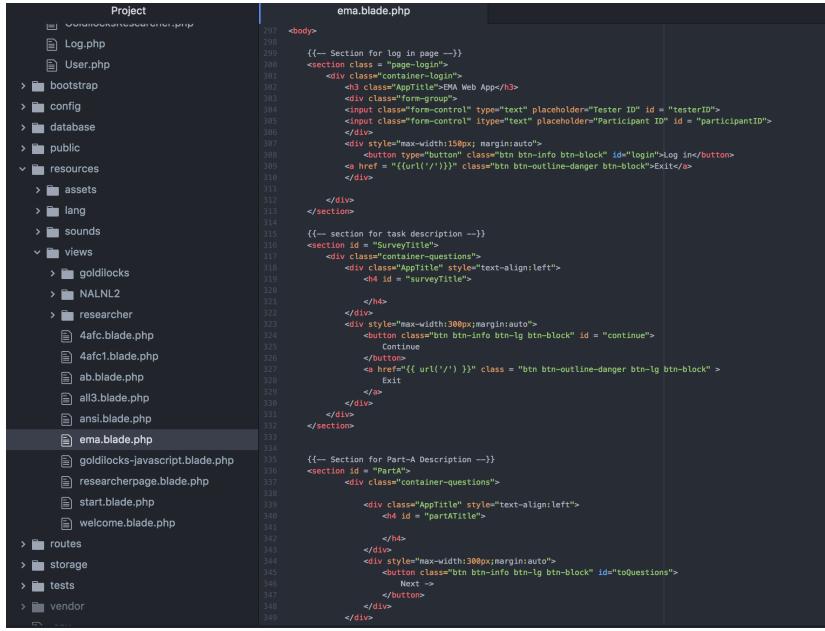
5.2 EMA Documentation

The EMA web-app is very similar to the 4AFC web-app in structure, and also has a JSON data structure, which you can edit to change the questions presented. We will first go over the three components, and then how to change questions in the EMA app.

1. **The View** file for this web-app, ema.blade.php can be found in the resources>views folder. Notice how Figure 5.22 is mostly written in JavaScript. You can see the part of the code (at the top), which actually saves the tester and listener ids. Next, notice on line 43, there is a call to fetch ema.json. This is the part of the code that loads in the JSON file.

```

Project
└── ema.blade.php
  29
  30  //log in handler
  31  $("#login").click(function(event){
  32    event.preventDefault();
  33
  34    //get the tester and participant ID
  35    testerID = $("#testerID").val();
  36    participantID = $("#participantID").val();
  37
  38    $("#page-login").hide();
  39
  40    //load the questions from json file
  41    async function load_survey(){
  42      try{
  43        const response = await fetch("ema.json");
  44
  45        const survey = await response.json();
  46
  47        return survey;
  48      }
  49      catch(err){
  50        alert(err.message);
  51      }
  52    }
  53
  54    //resolve the promise from async load function
  55    load_survey().then(data => {
  56      title = data["title"];
  57      partATitle = data["partA"];
  58      questions = data["questions"];
  59      numQuestions = questions.length;
  60      for(let i = 0; i < numQuestions; i++){
  61        if(questions[i]["isMultiple"]){
  62          $multiple.push(questions[i]);
  63        }
  64      });
  65
  66    });
  67
  68    $("#continue").click(function(event){
  69      event.preventDefault();
  70      $("#surveyTitle").hide();
  71      $("#partATitle").text(partATitle);
  72      $("#PartA").show();
  73      $("#nextQuestion").attr('disabled','disabled');
  74    });
  75
  76
  77
  78
  79
  80
  81
  82
  83
  84
  85
  86
  87
  88
  89
  90
  91
  92
  93
  94
  95
  96
  97
  98
  99
  100
  101
  102
  103
  104
  105
  106
  107
  108
  109
  110
  111
  112
  113
  114
  115
  116
  117
  118
  119
  120
  121
  122
  123
  124
  125
  126
  127
  128
  129
  130
  131
  132
  133
  134
  135
  136
  137
  138
  139
  140
  141
  142
  143
  144
  145
  146
  147
  148
  149
  150
  151
  152
  153
  154
  155
  156
  157
  158
  159
  160
  161
  162
  163
  164
  165
  166
  167
  168
  169
  170
  171
  172
  173
  174
  175
  176
  177
  178
  179
  180
  181
  182
  183
  184
  185
  186
  187
  188
  189
  190
  191
  192
  193
  194
  195
  196
  197
  198
  199
  200
  201
  202
  203
  204
  205
  206
  207
  208
  209
  210
  211
  212
  213
  214
  215
  216
  217
  218
  219
  220
  221
  222
  223
  224
  225
  226
  227
  228
  229
  230
  231
  232
  233
  234
  235
  236
  237
  238
  239
  240
  241
  242
  243
  244
  245
  246
  247
  248
  249
  250
  251
  252
  253
  254
  255
  256
  257
  258
  259
  260
  261
  262
  263
  264
  265
  266
  267
  268
  269
  270
  271
  272
  273
  274
  275
  276
  277
  278
  279
  280
  281
  282
  283
  284
  285
  286
  287
  288
  289
  290
  291
  292
  293
  294
  295
  296
  297
  298
  299
  300
  301
  302
  303
  304
  305
  306
  307
  308
  309
  310
  311
  312
  313
  314
  315
  316
  317
  318
  319
  320
  321
  322
  323
  324
  325
  326
  327
  328
  329
  330
  331
  332
  333
  334
  335
  336
  337
  338
  339
  340
  341
  342
  343
  344
  345
  346
  347
  348
  349
  350
  351
  352
  353
  354
  355
  356
  357
  358
  359
  360
  361
  362
  363
  364
  365
  366
  367
  368
  369
  370
  371
  372
  373
  374
  375
  376
  377
  378
  379
  380
  381
  382
  383
  384
  385
  386
  387
  388
  389
  390
  391
  392
  393
  394
  395
  396
  397
  398
  399
  400
  401
  402
  403
  404
  405
  406
  407
  408
  409
  410
  411
  412
  413
  414
  415
  416
  417
  418
  419
  420
  421
  422
  423
  424
  425
  426
  427
  428
  429
  430
  431
  432
  433
  434
  435
  436
  437
  438
  439
  440
  441
  442
  443
  444
  445
  446
  447
  448
  449
  450
  451
  452
  453
  454
  455
  456
  457
  458
  459
  460
  461
  462
  463
  464
  465
  466
  467
  468
  469
  470
  471
  472
  473
  474
  475
  476
  477
  478
  479
  480
  481
  482
  483
  484
  485
  486
  487
  488
  489
  490
  491
  492
  493
  494
  495
  496
  497
  498
  499
  500
  501
  502
  503
  504
  505
  506
  507
  508
  509
  510
  511
  512
  513
  514
  515
  516
  517
  518
  519
  520
  521
  522
  523
  524
  525
  526
  527
  528
  529
  530
  531
  532
  533
  534
  535
  536
  537
  538
  539
  540
  541
  542
  543
  544
  545
  546
  547
  548
  549
  550
  551
  552
  553
  554
  555
  556
  557
  558
  559
  560
  561
  562
  563
  564
  565
  566
  567
  568
  569
  570
  571
  572
  573
  574
  575
  576
  577
  578
  579
  580
  581
  582
  583
  584
  585
  586
  587
  588
  589
  590
  591
  592
  593
  594
  595
  596
  597
  598
  599
  600
  601
  602
  603
  604
  605
  606
  607
  608
  609
  610
  611
  612
  613
  614
  615
  616
  617
  618
  619
  620
  621
  622
  623
  624
  625
  626
  627
  628
  629
  630
  631
  632
  633
  634
  635
  636
  637
  638
  639
  640
  641
  642
  643
  644
  645
  646
  647
  648
  649
  650
  651
  652
  653
  654
  655
  656
  657
  658
  659
  660
  661
  662
  663
  664
  665
  666
  667
  668
  669
  670
  671
  672
  673
  674
  675
  676
  677
  678
  679
  680
  681
  682
  683
  684
  685
  686
  687
  688
  689
  690
  691
  692
  693
  694
  695
  696
  697
  698
  699
  700
  701
  702
  703
  704
  705
  706
  707
  708
  709
  710
  711
  712
  713
  714
  715
  716
  717
  718
  719
  720
  721
  722
  723
  724
  725
  726
  727
  728
  729
  730
  731
  732
  733
  734
  735
  736
  737
  738
  739
  740
  741
  742
  743
  744
  745
  746
  747
  748
  749
  750
  751
  752
  753
  754
  755
  756
  757
  758
  759
  760
  761
  762
  763
  764
  765
  766
  767
  768
  769
  770
  771
  772
  773
  774
  775
  776
  777
  778
  779
  780
  781
  782
  783
  784
  785
  786
  787
  788
  789
  790
  791
  792
  793
  794
  795
  796
  797
  798
  799
  800
  801
  802
  803
  804
  805
  806
  807
  808
  809
  810
  811
  812
  813
  814
  815
  816
  817
  818
  819
  820
  821
  822
  823
  824
  825
  826
  827
  828
  829
  830
  831
  832
  833
  834
  835
  836
  837
  838
  839
  840
  841
  842
  843
  844
  845
  846
  847
  848
  849
  850
  851
  852
  853
  854
  855
  856
  857
  858
  859
  860
  861
  862
  863
  864
  865
  866
  867
  868
  869
  870
  871
  872
  873
  874
  875
  876
  877
  878
  879
  880
  881
  882
  883
  884
  885
  886
  887
  888
  889
  890
  891
  892
  893
  894
  895
  896
  897
  898
  899
  900
  901
  902
  903
  904
  905
  906
  907
  908
  909
  910
  911
  912
  913
  914
  915
  916
  917
  918
  919
  920
  921
  922
  923
  924
  925
  926
  927
  928
  929
  930
  931
  932
  933
  934
  935
  936
  937
  938
  939
  940
  941
  942
  943
  944
  945
  946
  947
  948
  949
  950
  951
  952
  953
  954
  955
  956
  957
  958
  959
  960
  961
  962
  963
  964
  965
  966
  967
  968
  969
  970
  971
  972
  973
  974
  975
  976
  977
  978
  979
  980
  981
  982
  983
  984
  985
  986
  987
  988
  989
  990
  991
  992
  993
  994
  995
  996
  997
  998
  999
  1000
  1001
  1002
  1003
  1004
  1005
  1006
  1007
  1008
  1009
  1010
  1011
  1012
  1013
  1014
  1015
  1016
  1017
  1018
  1019
  1020
  1021
  1022
  1023
  1024
  1025
  1026
  1027
  1028
  1029
  1030
  1031
  1032
  1033
  1034
  1035
  1036
  1037
  1038
  1039
  1040
  1041
  1042
  1043
  1044
  1045
  1046
  1047
  1048
  1049
  1050
  1051
  1052
  1053
  1054
  1055
  1056
  1057
  1058
  1059
  1060
  1061
  1062
  1063
  1064
  1065
  1066
  1067
  1068
  1069
  1070
  1071
  1072
  1073
  1074
  1075
  1076
  1077
  1078
  1079
  1080
  1081
  1082
  1083
  1084
  1085
  1086
  1087
  1088
  1089
  1090
  1091
  1092
  1093
  1094
  1095
  1096
  1097
  1098
  1099
  1100
  1101
  1102
  1103
  1104
  1105
  1106
  1107
  1108
  1109
  1110
  1111
  1112
  1113
  1114
  1115
  1116
  1117
  1118
  1119
  1120
  1121
  1122
  1123
  1124
  1125
  1126
  1127
  1128
  1129
  1130
  1131
  1132
  1133
  1134
  1135
  1136
  1137
  1138
  1139
  1140
  1141
  1142
  1143
  1144
  1145
  1146
  1147
  1148
  1149
  1150
  1151
  1152
  1153
  1154
  1155
  1156
  1157
  1158
  1159
  1160
  1161
  1162
  1163
  1164
  1165
  1166
  1167
  1168
  1169
  1170
  1171
  1172
  1173
  1174
  1175
  1176
  1177
  1178
  1179
  1180
  1181
  1182
  1183
  1184
  1185
  1186
  1187
  1188
  1189
  1190
  1191
  1192
  1193
  1194
  1195
  1196
  1197
  1198
  1199
  1200
  1201
  1202
  1203
  1204
  1205
  1206
  1207
  1208
  1209
  1210
  1211
  1212
  1213
  1214
  1215
  1216
  1217
  1218
  1219
  1220
  1221
  1222
  1223
  1224
  1225
  1226
  1227
  1228
  1229
  1230
  1231
  1232
  1233
  1234
  1235
  1236
  1237
  1238
  1239
  1240
  1241
  1242
  1243
  1244
  1245
  1246
  1247
  1248
  1249
  1250
  1251
  1252
  1253
  1254
  1255
  1256
  1257
  1258
  1259
  1260
  1261
  1262
  1263
  1264
  1265
  1266
  1267
  1268
  1269
  1270
  1271
  1272
  1273
  1274
  1275
  1276
  1277
  1278
  1279
  1280
  1281
  1282
  1283
  1284
  1285
  1286
  1287
  1288
  1289
  1290
  1291
  1292
  1293
  1294
  1295
  1296
  1297
  1298
  1299
  1300
  1301
  1302
  1303
  1304
  1305
  1306
  1307
  1308
  1309
  1310
  1311
  1312
  1313
  1314
  1315
  1316
  1317
  1318
  1319
  1320
  1321
  1322
  1323
  1324
  1325
  1326
  1327
  1328
  1329
  1330
  1331
  1332
  1333
  1334
  1335
  1336
  1337
  1338
  1339
  1340
  1341
  1342
  1343
  1344
  1345
  1346
  1347
  1348
  1349
  1350
  1351
  1352
  1353
  1354
  1355
  1356
  1357
  1358
  1359
  1360
  1361
  1362
  1363
  1364
  1365
  1366
  1367
  1368
  1369
  1370
  1371
  1372
  1373
  1374
  1375
  1376
  1377
  1378
  1379
  1380
  1381
  1382
  1383
  1384
  1385
  1386
  1387
  1388
  1389
  1390
  1391
  1392
  1393
  1394
  1395
  1396
  1397
  1398
  1399
  1400
  1401
  1402
  1403
  1404
  1405
  1406
  1407
  1408
  1409
  1410
  1411
  1412
  1413
  1414
  1415
  1416
  1417
  1418
  1419
  1420
  1421
  1422
  1423
  1424
  1425
  1426
  1427
  1428
  1429
  1430
  1431
  1432
  1433
  1434
  1435
  1436
  1437
  1438
  1439
  1440
  1441
  1442
  1443
  1444
  1445
  1446
  1447
  1448
  1449
  1450
  1451
  1452
  1453
  1454
  1455
  1456
  1457
  1458
  1459
  1460
  1461
  1462
  1463
  1464
  1465
  1466
  1467
  1468
  1469
  1470
  1471
  1472
  1473
  1474
  1475
  1476
  1477
  1478
  1479
  1480
  1481
  1482
  1483
  1484
  1485
  1486
  1487
  1488
  1489
  1490
  1491
  1492
  1493
  1494
  1495
  1496
  1497
  1498
  1499
  1500
  1501
  1502
  1503
  1504
  1505
  1506
  1507
  1508
  1509
  1510
  1511
  1512
  1513
  1514
  1515
  1516
  1517
  1518
  1519
  1520
  1521
  1522
  1523
  1524
  1525
  1526
  1527
  1528
  1529
  1530
  1531
  1532
  1533
  1534
  1535
  1536
  1537
  1538
  1539
  1540
  1541
  1542
  1543
  1544
  1545
  1546
  1547
  1548
  1549
  1550
  1551
  1552
  1553
  1554
  1555
  1556
  1557
  1558
  1559
  1560
  1561
  1562
  1563
  1564
  1565
  1566
  1567
  1568
  1569
  1570
  1571
  1572
  1573
  1574
  1575
  1576
  1577
  1578
  1579
  1580
  1581
  1582
  1583
  1584
  1585
  1586
  1587
  1588
  1589
  1590
  1591
  1592
  1593
  1594
  1595
  1596
  1597
  1598
  1599
  1600
  1601
  1602
  1603
  1604
  1605
  1606
  1607
  1608
  1609
  1610
  1611
  1612
  1613
  1614
  1615
  1616
  1617
  1618
  1619
  1620
  1621
  1622
  1623
  1624
  1625
  1626
  1627
  1628
  1629
  1630
  1631
  1632
  1633
  1634
  1635
  1636
  1637
  1638
  1639
  1640
  1641
  1642
  1643
  1644
  1645
  1646
  1647
  1648
  1649
  1650
  1651
  1652
  1653
  1654
  1655
  1656
  1657
  1658
  1659
  1660
  1661
  1662
  1663
  1664
  1665
  1666
  1667
  1668
  1669
  1670
  1671
  1672
  1673
  1674
  1675
  1676
  1677
  1678
  1679
  1680
  1681
  1682
  1683
  1684
  1685
  1686
  1687
  1688
  1689
  1690
  1691
  1692
  1693
  1694
  1695
  1696
  1697
  1698
  1699
  1700
  1701
  1702
  1703
  1704
  
```



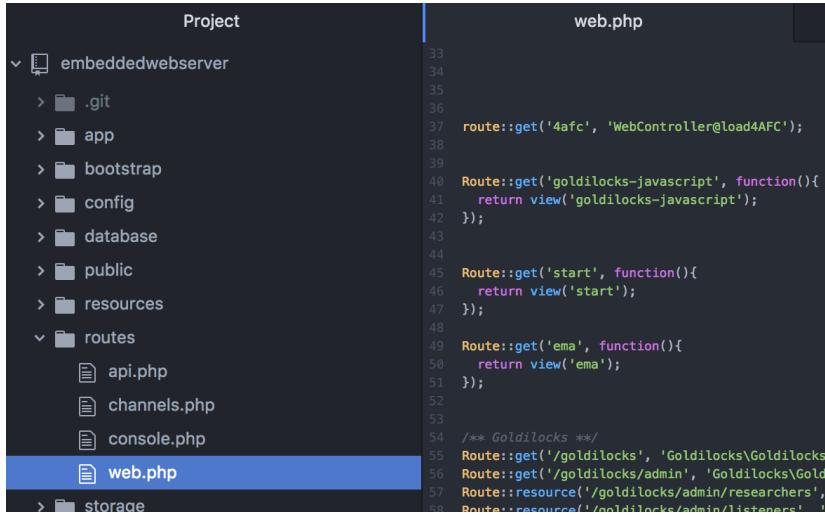
```

Project
└── ema.blade.php
    ...
    <body>
        ...
        <!-- Section for log in page -->
        <section class="page-login">
            ...
        </section>
        ...
        <!-- section for task description -->
        <section id="SurveyTitle">
            ...
        </section>
        ...
        <!-- Section for Part-A Description -->
        <section id="PartA">
            ...
        </section>
        ...
    </body>

```

Figure 5.23: This is a screenshot of the further down the ema.blade.php, which shows the HTML elements you see on the screen.

- The Route for this web-app is in the file web.php, located in the Routes folder. Reference figure 5.19. The EMA route starts on line 49 in the screenshot. Notice that this route is calling a function that returns the view of the EMA page. This is how the browser knows to display a specific page, given a particular URL. In this case, the URL is /ema, as you can see in the URL bar of your browser when you are on the page.



```

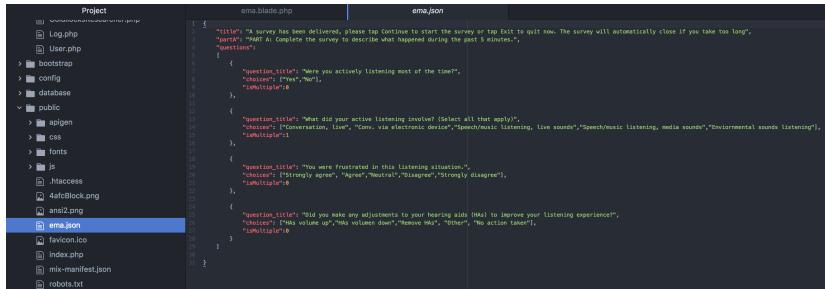
Project
└── routes
    └── web.php
        ...
        33
        34
        35
        36
        37     Route::get('4afc', 'WebController@load4AFC');
        38
        39
        40     Route::get('goldilocks-javascript', function(){
        41         return view('goldilocks-javascript');
        42     });
        43
        44
        45     Route::get('start', function(){
        46         return view('start');
        47     });
        48
        49     Route::get('ema', function(){
        50         return view('ema');
        51     });
        52
        53
        54     /** Goldilocks ***/
        55     Route::get('/goldilocks', 'Goldilocks\GoldilocksController@loadGoldilocks');
        56     Route::get('/goldilocks/admin', 'Goldilocks\GoldilocksController@loadGoldilocksAdmin');
        57     Route::resource('/goldilocks/admin/researchers', 'Goldilocks\GoldilocksController@loadResearcher');
        58     Route::resource('/goldilocks/admin/listeners', 'Goldilocks\GoldilocksController@loadListener');

```

Figure 5.24: This is a screenshot of the web.php file, which shows the route for the EMA web-app

3. There is no controller for this web-app

4. The JSON file associated with this webapp, ema.json, is in the public folder. As you look through the file, notice that it has all of the text content of the EMA web-app. This includes the task descriptions, as well as the contents of the questions. This is, in fact, the file that is being fetched in Figure 5.22.



```

Project
  -> bootstrap
    -> Log.php
    -> User.php
  -> config
  -> database
  -> public
    -> js
      -> spigen
        -> 4afcBlock.png
        -> ema.json
        -> favIcon.ico
        -> Index.php
        -> mix-manifest.json
        -> robots.txt
        -> web.config
  -> ema.blade.php
  -> ema.json

```

```

{
  "title": "A survey has been delivered, please tap Continue to start the survey or tap Exit to quit now. The survey will automatically close if you take too long",
  "partA": "PART A: Complete the survey to describe what happened during the past 5 minutes",
  "questions": [
    {
      "question_title": "Were you actively listening most of the time?",
      "choices": ["Yes", "No"],
      "isMultiple": 0
    },
    {
      "question_title": "What did your active listening involve? (Select all that apply)",
      "choices": ["Conversation, live", "Convo. via electronic device", "Speech/music listening, live sounds", "Speech/music listening, media sounds", "Environmental sounds listening"],
      "isMultiple": 1
    },
    {
      "question_title": "You were frustrated in this listening situation.",
      "choices": ["Strongly agree", "Agree", "Neutral", "Disagree", "Strongly disagree"],
      "isMultiple": 0
    },
    {
      "question_title": "Did you make any adjustments to your hearing aids (HAs) to improve your listening experience?",
      "choices": ["HAs volume up", "HAs volume down", "Remove HAs", "Other", "No action taken"],
      "isMultiple": 0
    }
  ]
}

```

Figure 5.25: This is a screenshot of the ema.json file, which holds all of the content that will be read into the web-app. This is the file you should edit to change the contents of the app.

In order to change the contents of the first page, which has the "title" text, simply change the text in this file to what you would like it to say. The next line "Part A" holds the instructions for the first part of the task. Then, all of the questions and question choices follow. Notice the line in the each block of question choices, "is-Multiple". This is a binary variable, either 1 if the question could have multiple answers, or 0 if there is only one answer choice.

There are many example questions provided that you could change the text of in order to make your own custom questions. Simply change the text that appears green from an existing question, restart and reload. Say we changed the last question to ask about how old the participant is. Because the participant should only have one answer for this question, I will set isMultiple=0.



```

Project
  -> bootstrap
    -> Log.php
    -> User.php
  -> config
  -> database
  -> public
    -> js
      -> spigen
        -> 4afcBlock.png
        -> ema.json
        -> favIcon.ico
        -> Index.php
        -> mix-manifest.json
        -> robots.txt
        -> web.config
  -> ema.blade.php
  -> ema.json

```

```

{
  "title": "A survey has been delivered, please tap Continue to start the survey or tap Exit to quit now. The survey will automatically close if you take too long",
  "partA": "PART A: Complete the survey to describe what happened during the past 5 minutes",
  "questions": [
    {
      "question_title": "Were you actively listening most of the time?",
      "choices": ["Yes", "No"],
      "isMultiple": 0
    },
    {
      "question_title": "What did your active listening involve? (Select all that apply)",
      "choices": ["Conversation, live", "Convo. via electronic device", "Speech/music listening, live sounds", "Speech/music listening, media sounds", "Environmental sounds listening"],
      "isMultiple": 1
    },
    {
      "question_title": "You were frustrated in this listening situation.",
      "choices": ["Strongly agree", "Agree", "Neutral", "Disagree", "Strongly disagree"],
      "isMultiple": 0
    },
    {
      "question_title": "How old are you?",
      "choices": ["18-24", "25-35", "35-45", "40", "No action taken"],
      "isMultiple": 0
    }
  ]
}

```

Figure 5.26: This is a screenshot of the ema.json file, with the last question changed to ask about the participant's age

See figure 5.26 to look at the JSON file including this question. See figure 5.27 to see how this actually looks in the web-app! Editing

the content of this web-app is as simple as changing the text in this file.

A screenshot of a web-based application interface. At the top, there is a teal header bar. Below it, the main content area has a white background. A large, bold, black font asks "How old are you?". Below this question are five rectangular buttons, each containing a different age range: "18-24", "25-35", "35-45", "45-55", and "55+". At the bottom of the screen are two large buttons: a red one on the left labeled "<- Back" and a teal one on the right labeled "Done".

Figure 5.27: This is a screenshot of the EMA web-app, which shows the updated question