

- **Changes**

Design of my website has not changed a lot. I decided to add some media queries in the mobile CSS file, because I was not sure how viewport property works and that they are eventually needed to control size of fonts and table for different mobile screens. I added media queries for minimum screen size of 400px and 700px.

I also added round corners because my website has a maximum size and now borders look better. I wanted to show some of my work but eventually I decided that none of the pieces of code is really interesting for target audience, so instead I put in the degree page images that illustrate every module in the simple way.

- **Organisation**

I created template for HTML file with the same header, nav and footer elements. Then I copied it six times and created different main sections in every page. I created additional folders for CSS files, JavaScript files and images. In debugging I mainly used console.log() to print values in Google Chrome Developer Mode and see if the code works correctly. I also used JSLint and JSHint to find any problems with my code. Most of the problems in canvas.js file occurred due to missing semicolons and too many characters in a line. There were no problems with the quiz.js file.

```
129 /
130 // CHARTS
131
132 // draws pie chart with percent indicators
133 function drawPie() {
134   clearCanvas();
135   start();
136
137   let sum = sum_array(arr_whatsapp) + sum_array(arr_messenger) + sum_array(arr_instagram);
138   let x = Math.PI * 2 * (sum_array(arr_whatsapp) / sum);
139   let y = Math.PI * 2 * (sum_array(arr_messenger) / sum);
140   let z = Math.PI * 2 * (sum_array(arr_instagram) / sum);
141
142   context.lineWidth = "1";
143   context.font = "60px Comic Sans MS";
144   context.fillStyle = "black";
145   context.textAlign = "center";
146
147   // red
148   context.fillText(Math.round(sum_array(arr_instagram) * 100 / sum) + "%", 300, 170);
149   //green
150   context.fillText(Math.round(sum_array(arr_whatsapp) * 100 / sum) + "%", 700, 800);
151   //blue
152   let percentage = 100 - Math.round(sum_array(arr_instagram) * 100 / sum) - Math.round(sum
153   context.fillText(percentage + "%", 100, 700);
154
155   context.beginPath();
156   context.strokeStyle = colour_first;
157   context.fillStyle = colour_first;
158   context.arc(500, 500, 300, 0, x, false);
159   context.lineTo(500, 500);
160   context.fill();
161   context.stroke()
162
163   context.beginPath();
164   context.strokeStyle = colour_second;
165   context.fillStyle = colour_second;
166   context.arc(500, 500, 300, x, x + y, false);
167   context.lineTo(500, 500);
168   context.fill();
169   context.stroke()
170
171   context.beginPath();
172   context.strokeStyle = colour_third;
173   context.fillStyle = colour_third;
174   context.arc(500, 500, 300, x + y, 0, false);
175   context.lineTo(500, 500);
176   context.fill();
177   context.stroke()
178
179   diagram_check = 0;
180 }
181
182
183
184 // draws bar chart
185 function drawBar() {
186   clearCanvas();
187   start();
188   lines();
189
190   let x = 120;
191   let y = 999;
```

143 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

144 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

137 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

138 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

139 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

140 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

152 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

161 Missing semicolon.

169 Missing semicolon.

177 Missing semicolon.

190 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

191 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

192 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

193 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

194 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

201 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

241 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

242 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

243 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

244 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

251 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

Figure 1 - JSHint check

Warnings		
JSLint was unable to finish.		
Expected ';' and instead saw 'var'.		5.0
var canvas = document.getElementById("canvas");		
Unexpected trailing space.		86.1
Line is longer than 80 characters.		137.80
let sum = sum_array(arr_whatsapp) + sum_array(arr_messenger) + sum_array(arr_instagram);		
Line is longer than 80 characters.		148.80
context.fillText(Math.round(sum_array(arr_instagram) * 100 / sum) + "%", 300, 170);		
Line is longer than 80 characters.		150.80
context.fillText(Math.round(sum_array(arr_whatsapp) * 100 / sum) + "%", 700, 800);		
Line is longer than 80 characters.		152.80
let percentage = 100 - Math.round(sum_array(arr_instagram) * 100 / sum) - Math.round(sum_array(arr_whatsapp) * 100 / sum);		
Unexpected trailing space.		219.3
Unexpected trailing space.		252.3
Unexpected trailing space.		276.3
Function Report		

Figure 2- JSLint check

- **Optimisation**

I used Google Lighthouse to determine whether my page has problems with loading time. Several times I needed to compress images, especially ones on pages with multiple images, to improve the website performance.

- **Security**

My page will not be put onto the public server, but if I wanted to, I would protect it with HTTPS. Every unprotected website can be used to reveal information about the user. That is because intruder can see all communication between a browser and a server. Having in mind that my website contains a form it would be a good approach to create function which encodes special character in the place control characters. It would prevent execution of intruder's code.

References:

[https://developer.mozilla.org/en-US/docs/Learn/Server-side/First\\_steps/Website\\_security](https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Website_security)

<https://web.dev/why-https-matters/>

- **Debugging**

I used HTML and CSS validators

HTML:

**Nu Html Checker**

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

**Showing results for degree.html**

Checker Input

Show ☐ source ☐ outline ☐ image report [Options...](#)

Check by [file upload](#) [Choose File](#) No file chosen

Uploaded files with .xhtml or .xht extensions are parsed using the XML parser.

[Check](#)

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

[Message Filtering](#)

- Error** Duplicate ID `module_image`.  
From line 59, column 11; to line 59, column 89  
``
- Warning** The first occurrence of ID `module_image` was here.  
From line 45, column 11; to line 45, column 91  
``
- Error** Duplicate ID `module_image`.  
From line 73, column 11; to line 73, column 94  
``
- Warning** The first occurrence of ID `module_image` was here.  
From line 45, column 11; to line 45, column 91  
``
- Error** Duplicate ID `module_image`.  
From line 88, column 11; to line 88, column 99  
``
- Warning** The first occurrence of ID `module_image` was here.

Figure 3 - HTML degree check

I accidentally copied the same ID for all degree images.

CSS:

**W3C** The W3C CSS Validation Service  
W3C CSS Validator results for mobileStyles.css (CSS level 3 + SVG)

[Jump to: Errors](#)

**W3C CSS Validator results for mobileStyles.css (CSS level 3 + SVG)**

**Sorry! We found the following errors (3)**

URI : mobileStyles.css

27	nav	Value Error : text-align <code>just</code> is not a <code>text-align</code> value : <code>just</code>
63	.name_mobile	Value Error : font-weight <code>1000</code> is not a <code>font-weight</code> value : <code>1000</code>
75	#banner	Value Error : font-weight <code>1000</code> is not a <code>font-weight</code> value : <code>1000</code>

Figure 4 - CSS mobile check

I used here wrong values for text-align and font-weight properties. The first one should be justify and the second one should be 900.

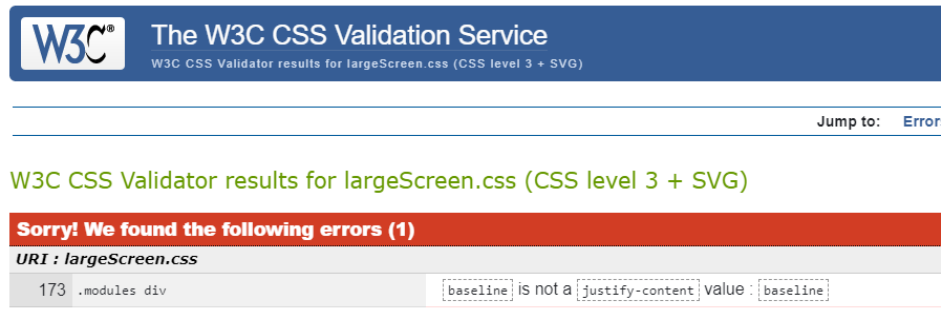


Figure 5 - CSS desktop check

I used here justify-content instead of align-items.

All other files did not have any errors.

## • Testing

I tested how my website looks and behave on different browsers and I have not noticed any problem with pages, especially I looked at quiz and canvas pages.

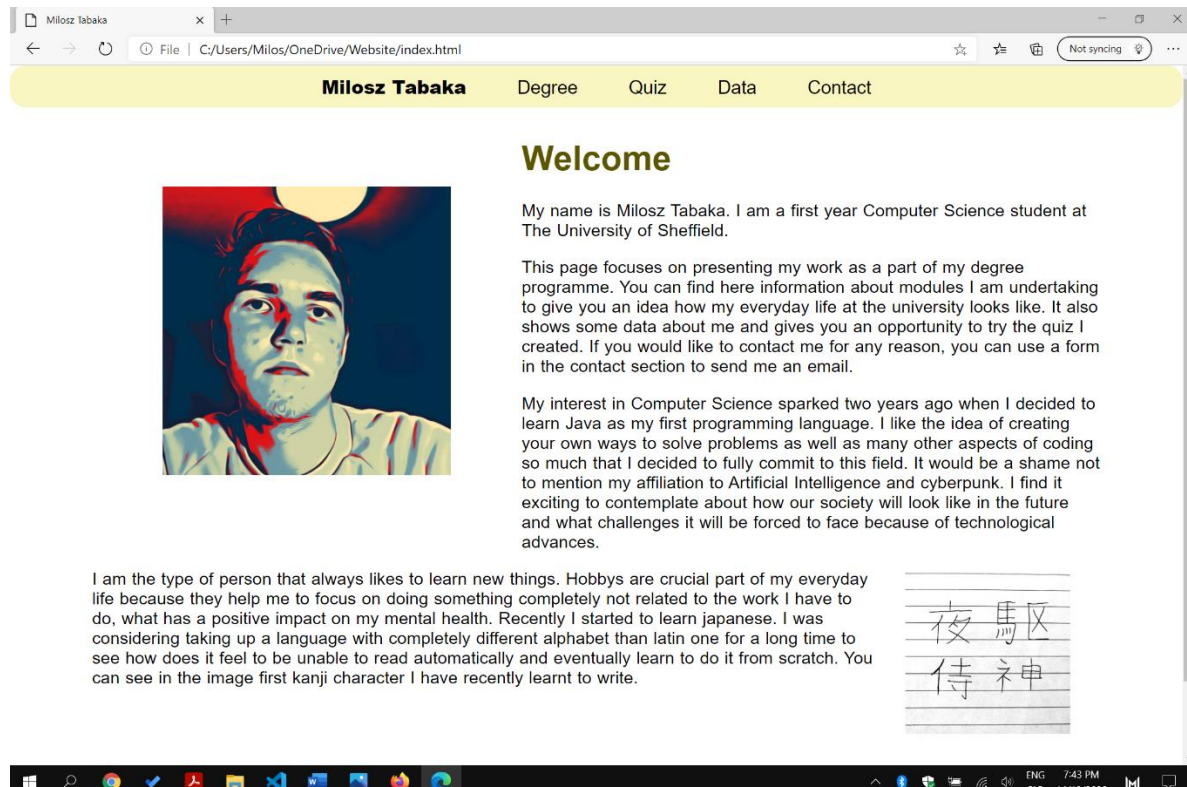


Figure 6 - Microsoft Edge test

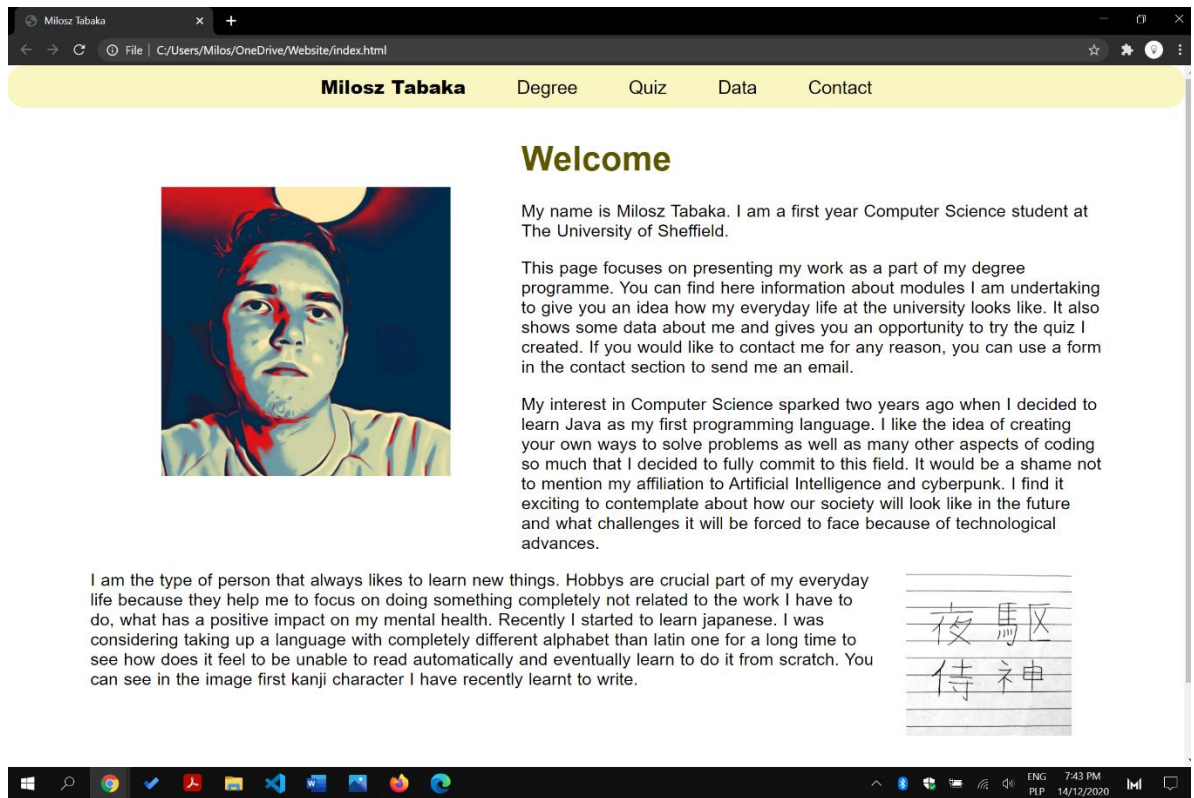


Figure 7 - Google Chrome test

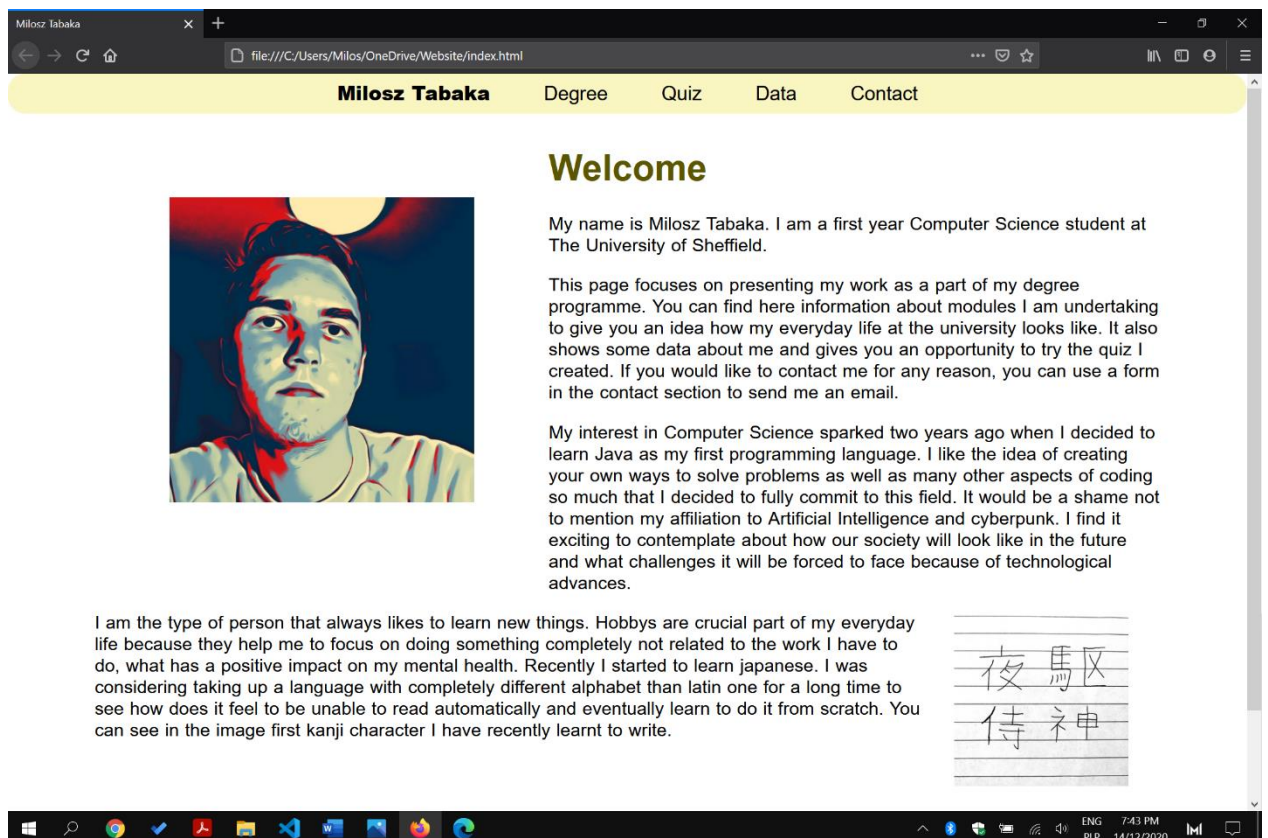


Figure 8 - Mozilla Firefox test



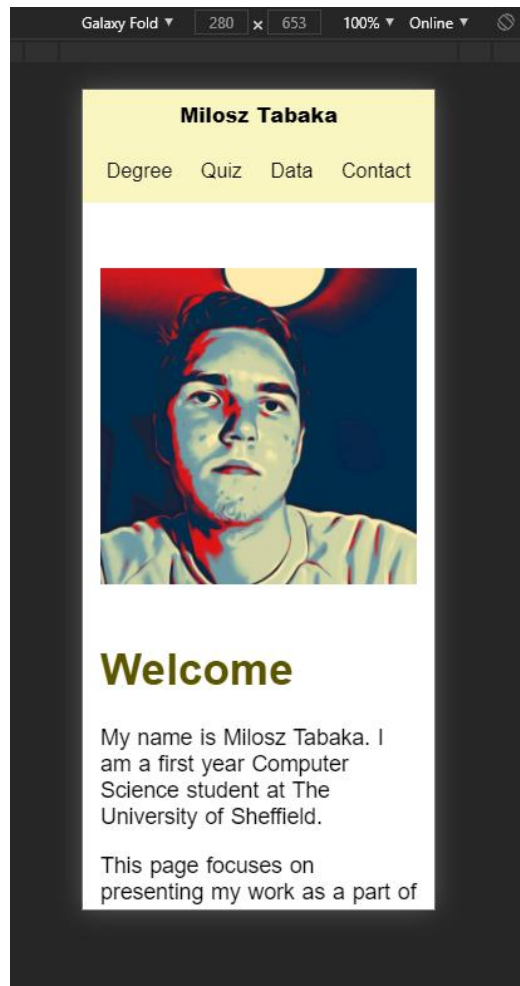


Figure 9 - Minimum screen size test

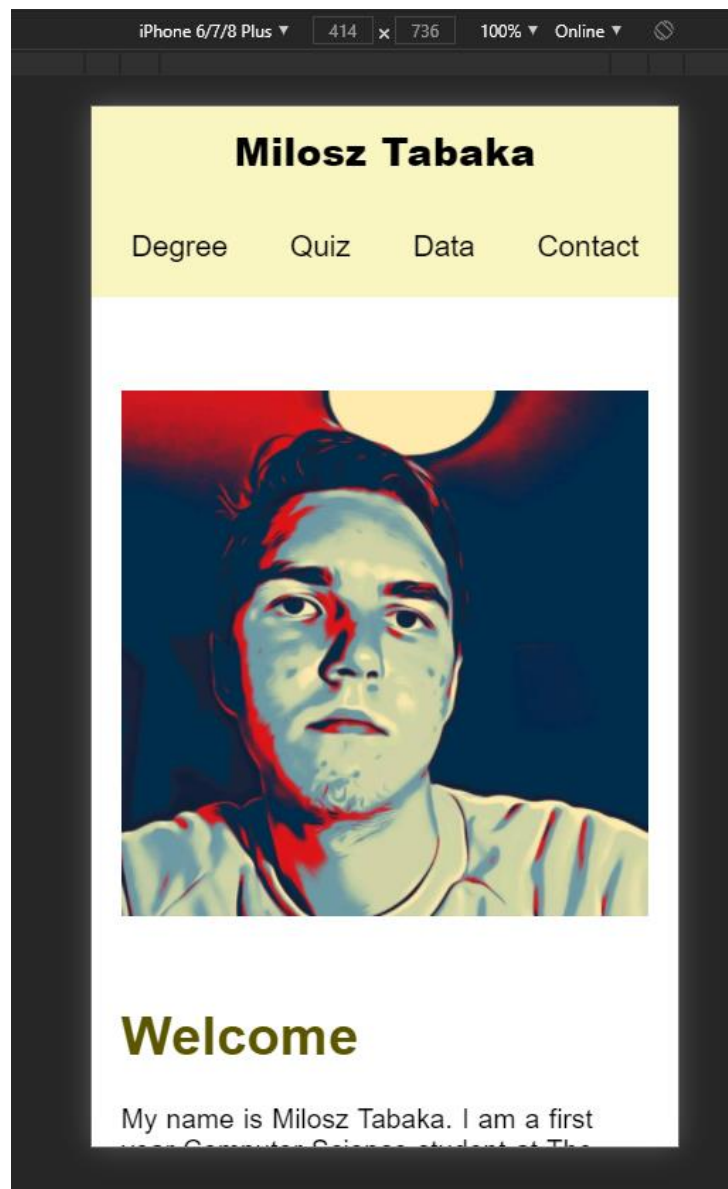


Figure 10 - Second breakpoint test

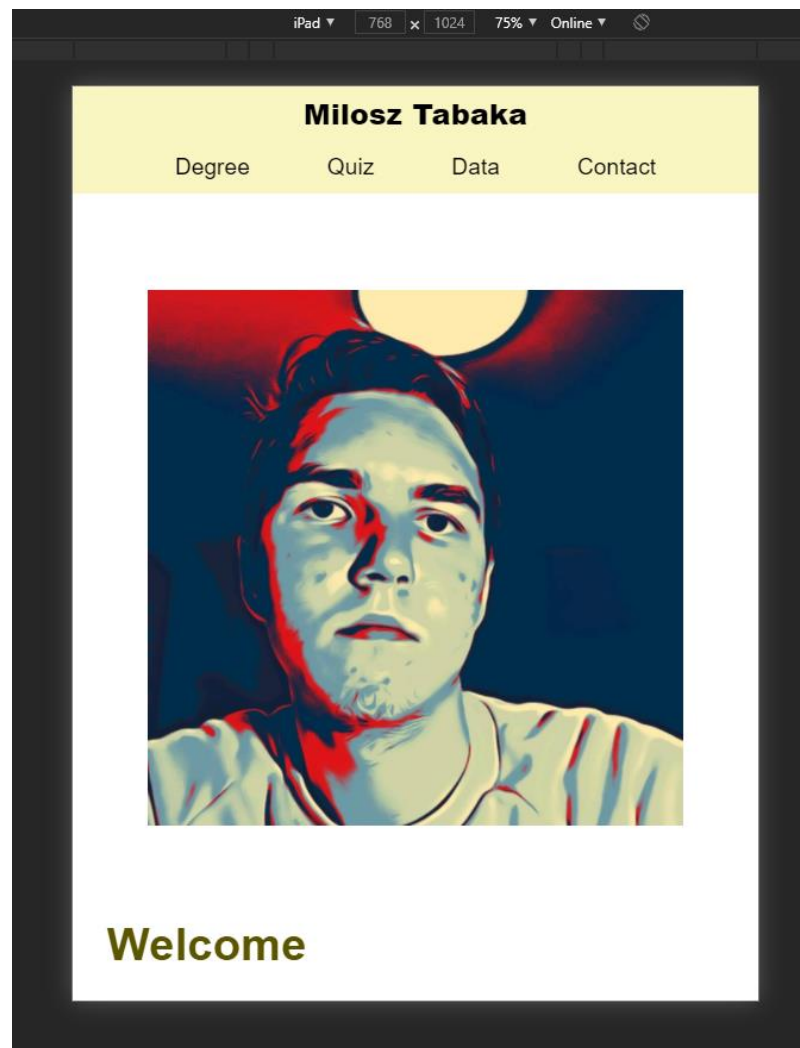


Figure 11 - Third breakpoint test



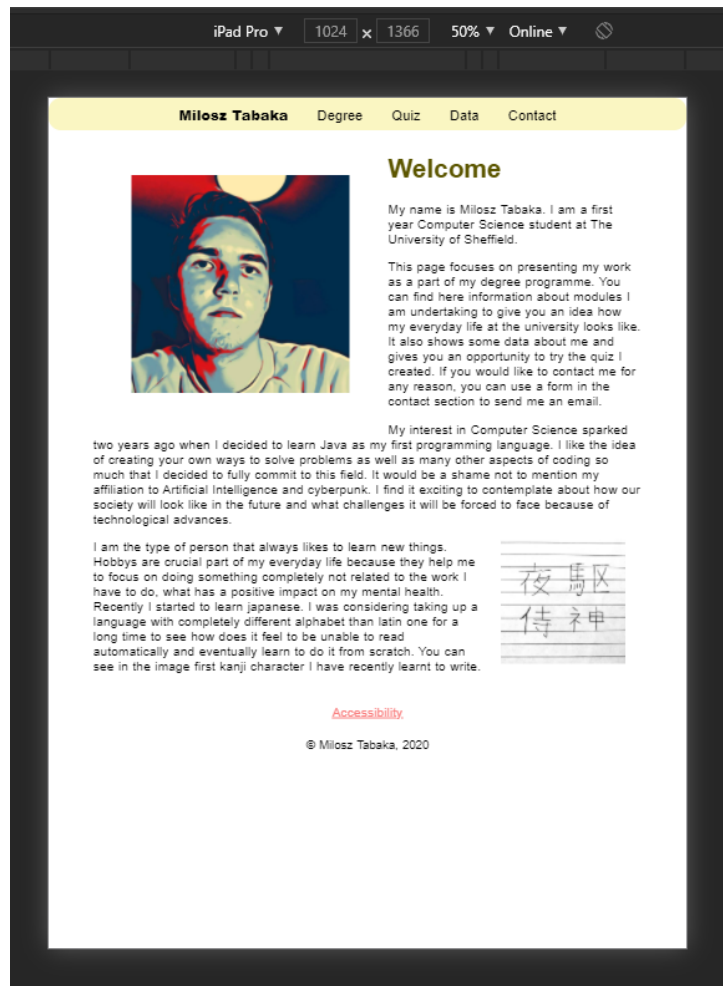


Figure 12 - Minimum desktop screen size test

I also tested every page using Google Lighthouse Tool

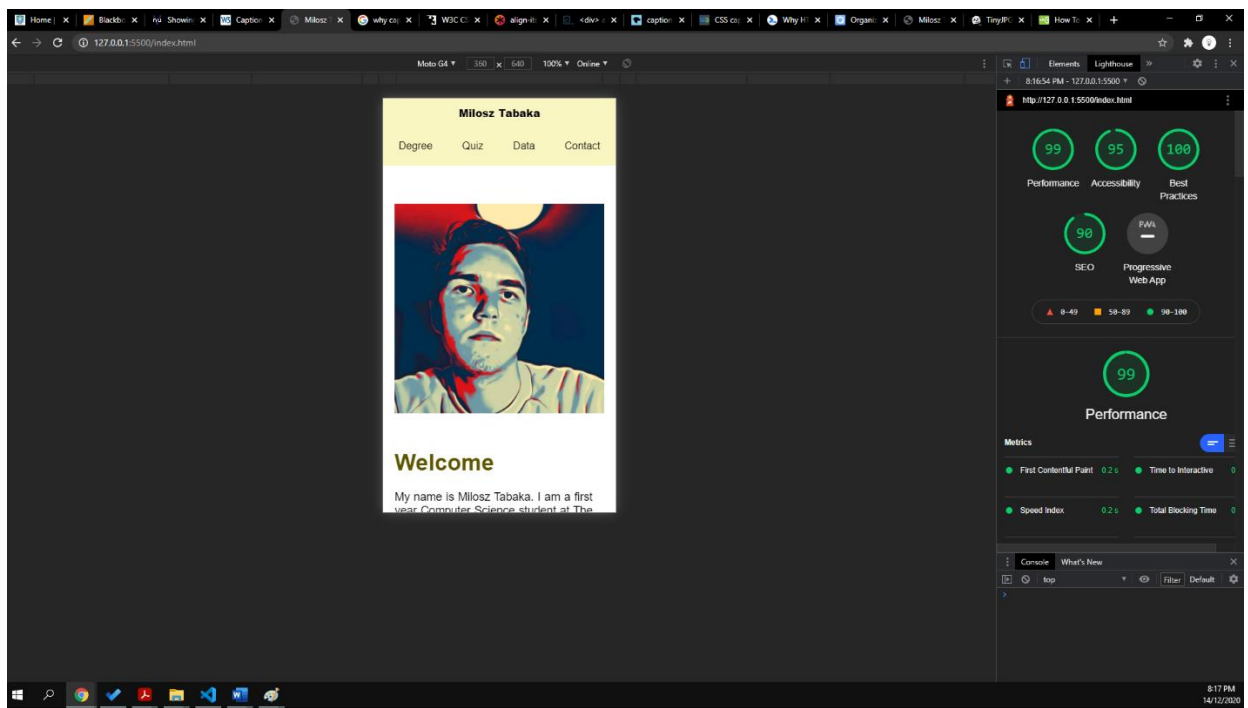


Figure 13 - Mobile Index test

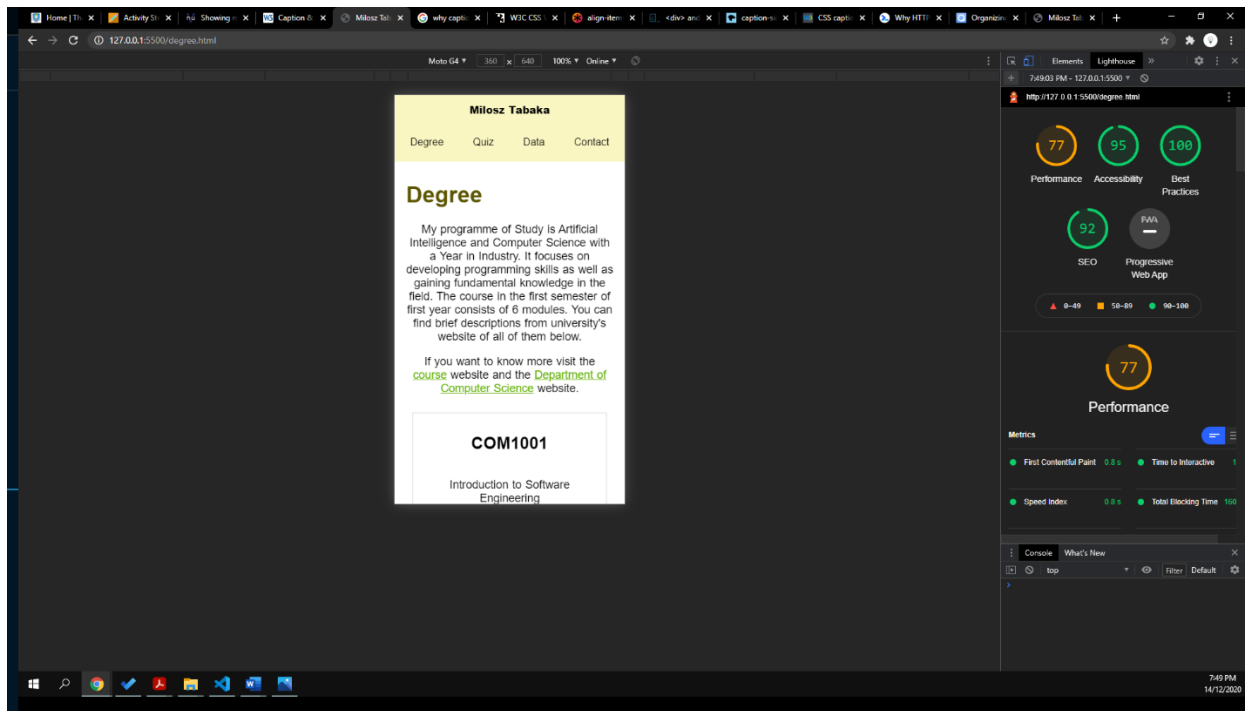


Figure 14 - Mobile Degree test (before compression)

Here I compressed the images

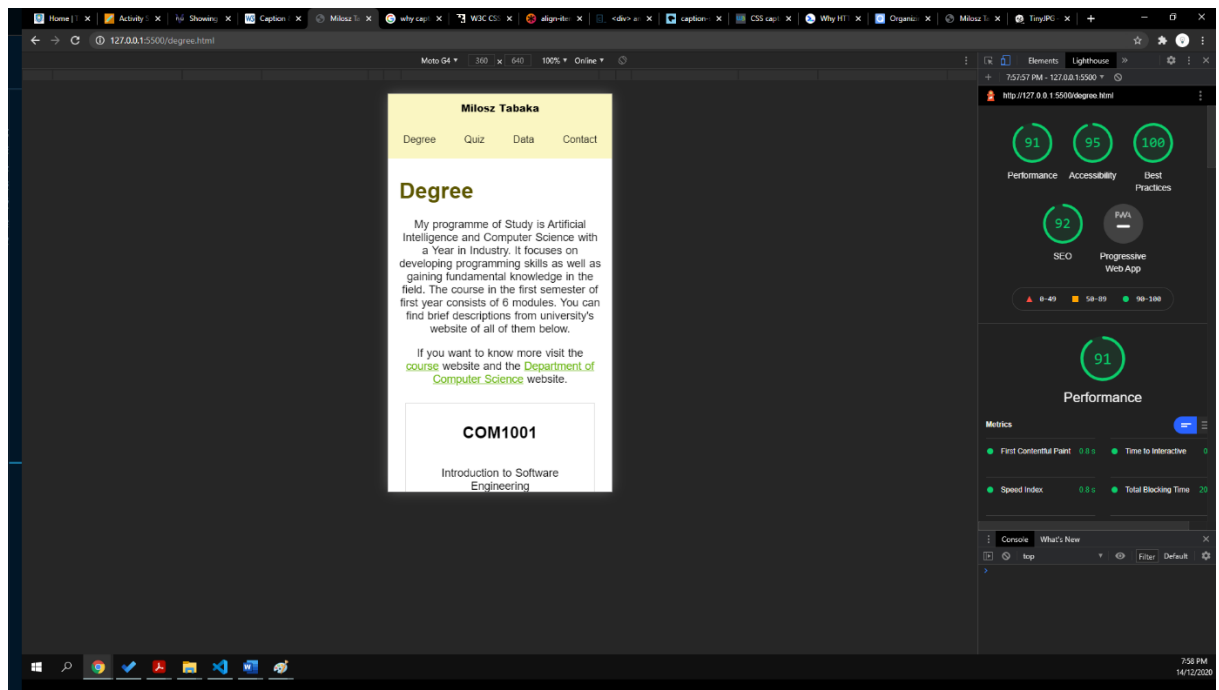


Figure 15 - Mobile Degree test (after compression)

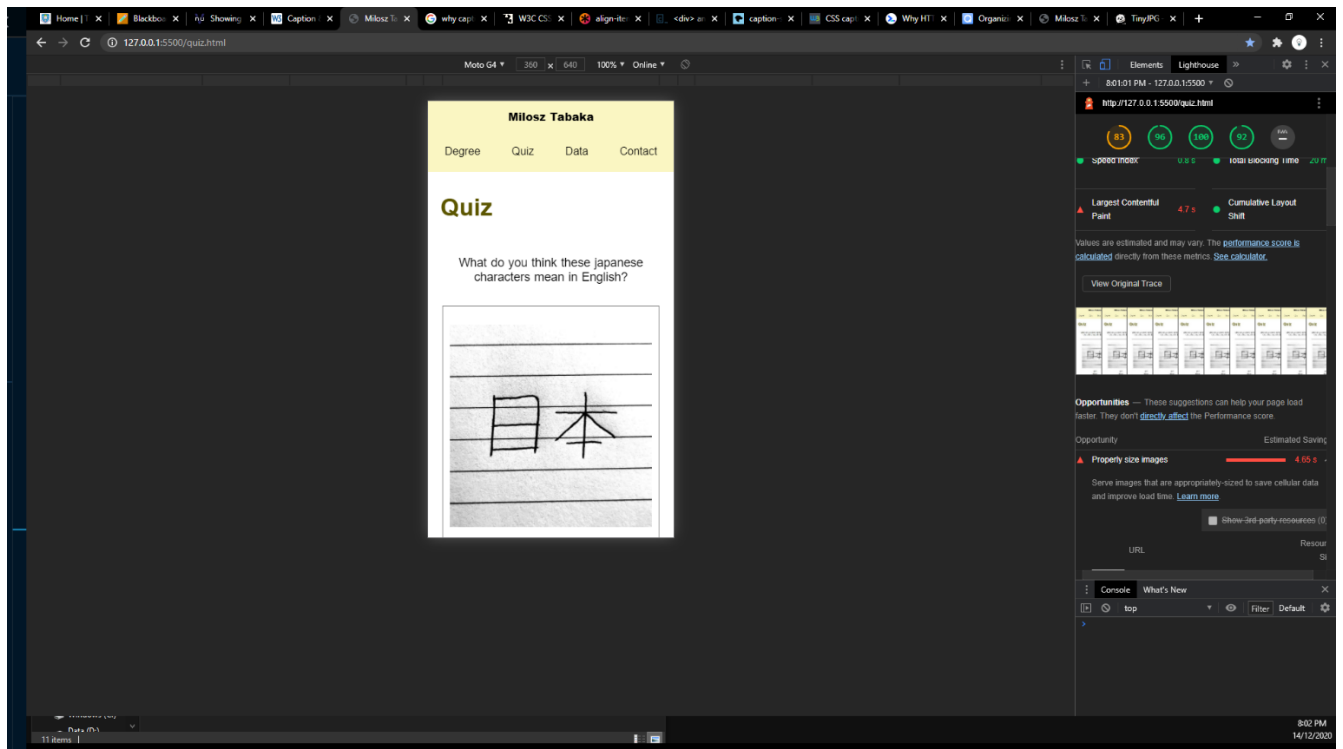


Figure 16 - Mobile Quiz test

Here I also compressed the images a few times, but still that is the best result I achieved

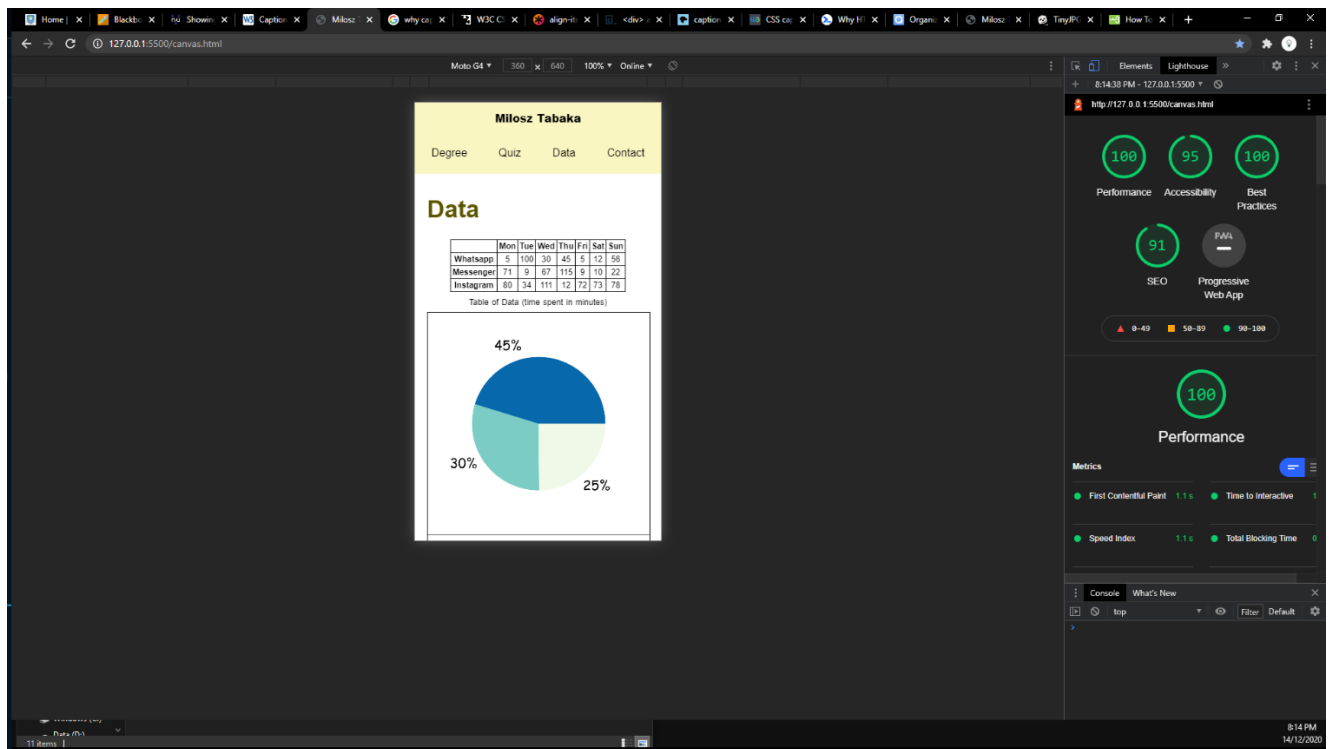


Figure 17 - Mobile Data test

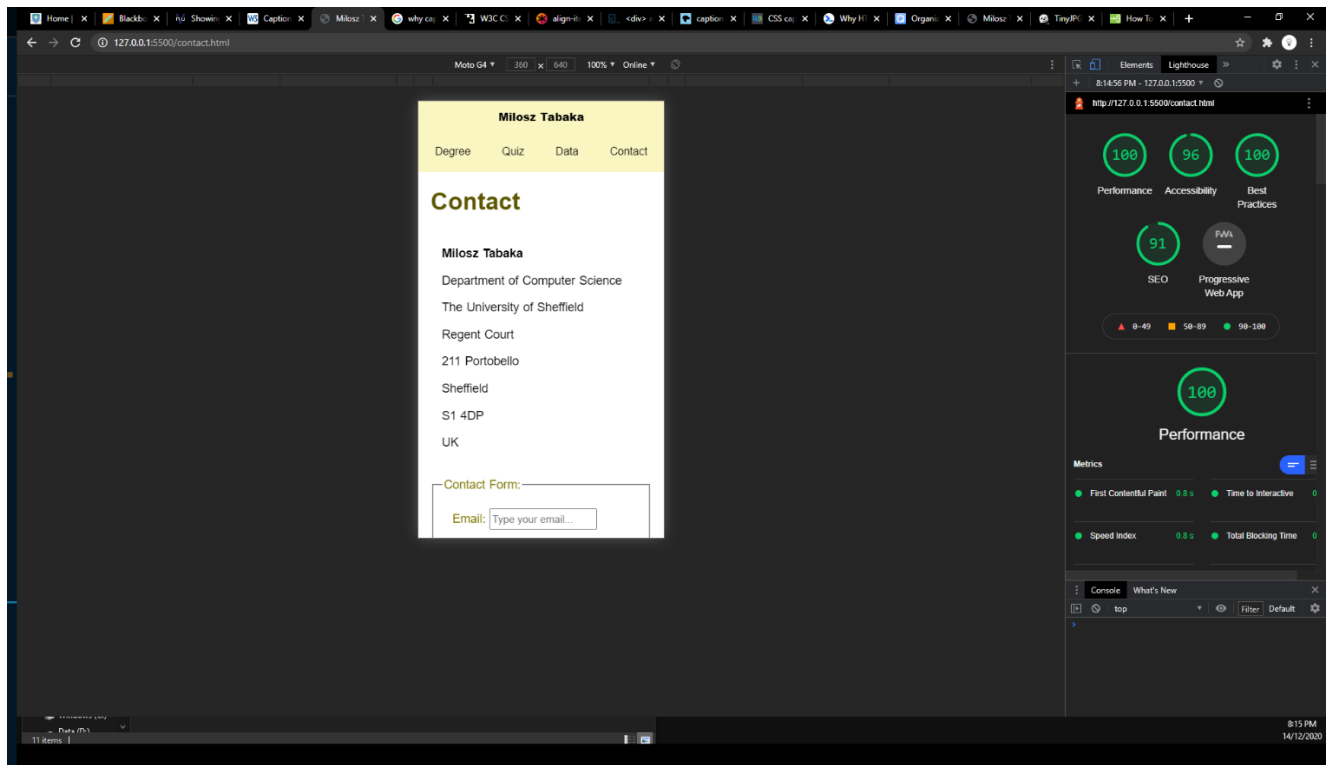


Figure 18 - Mobile Contact test

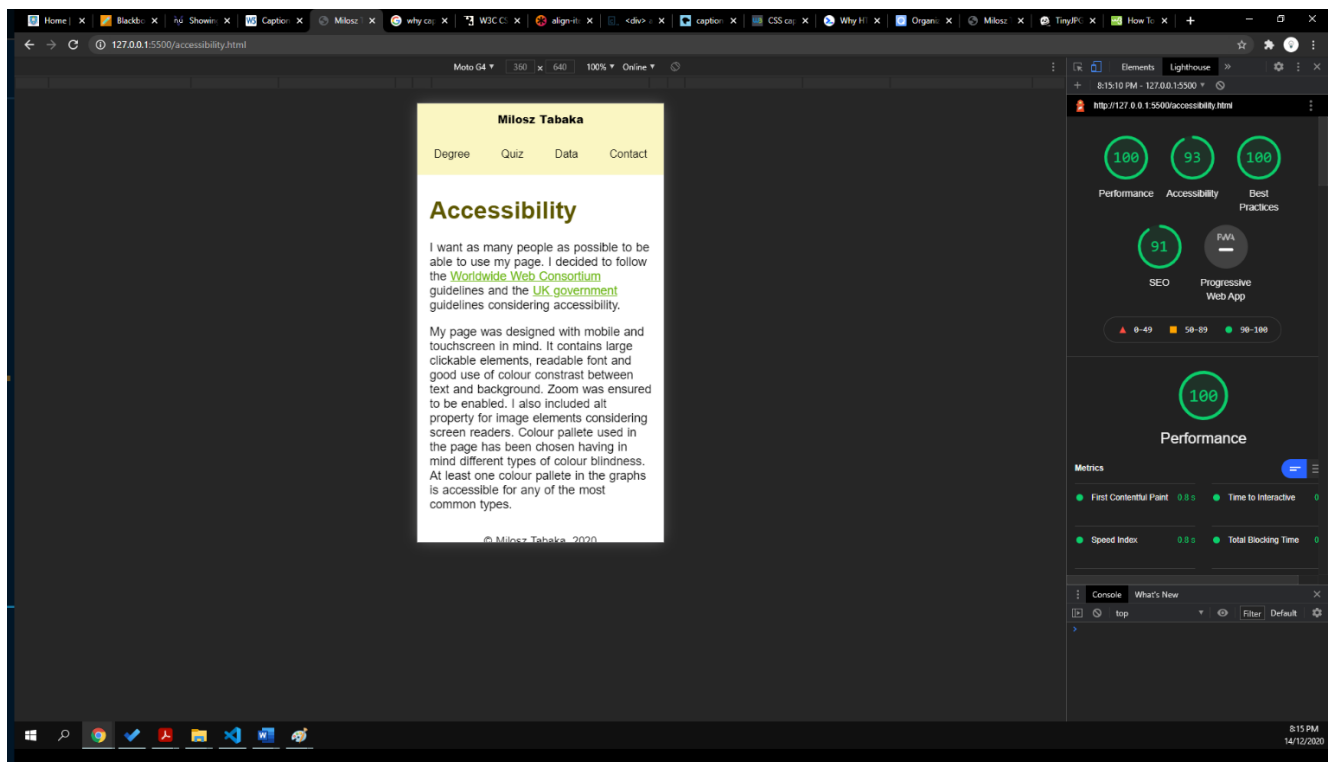


Figure 19 - Mobile Accessibility test

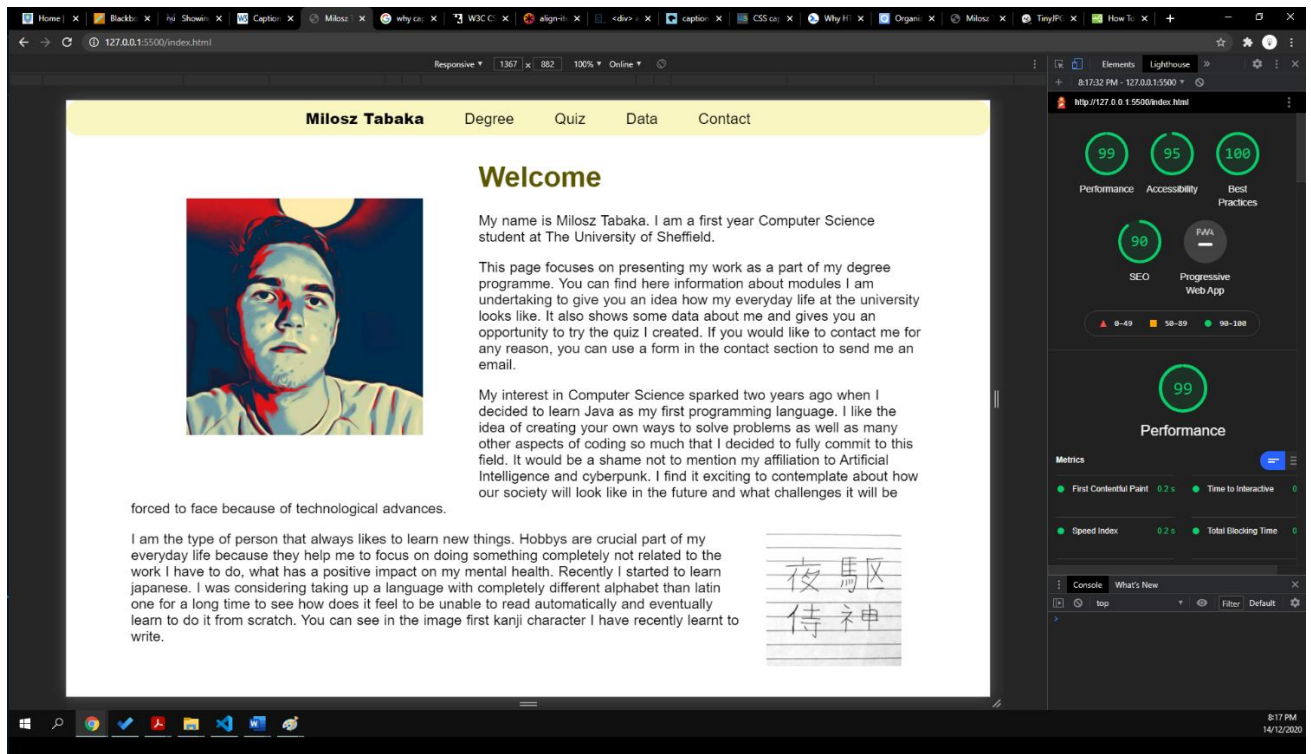


Figure 20 - Desktop Index test

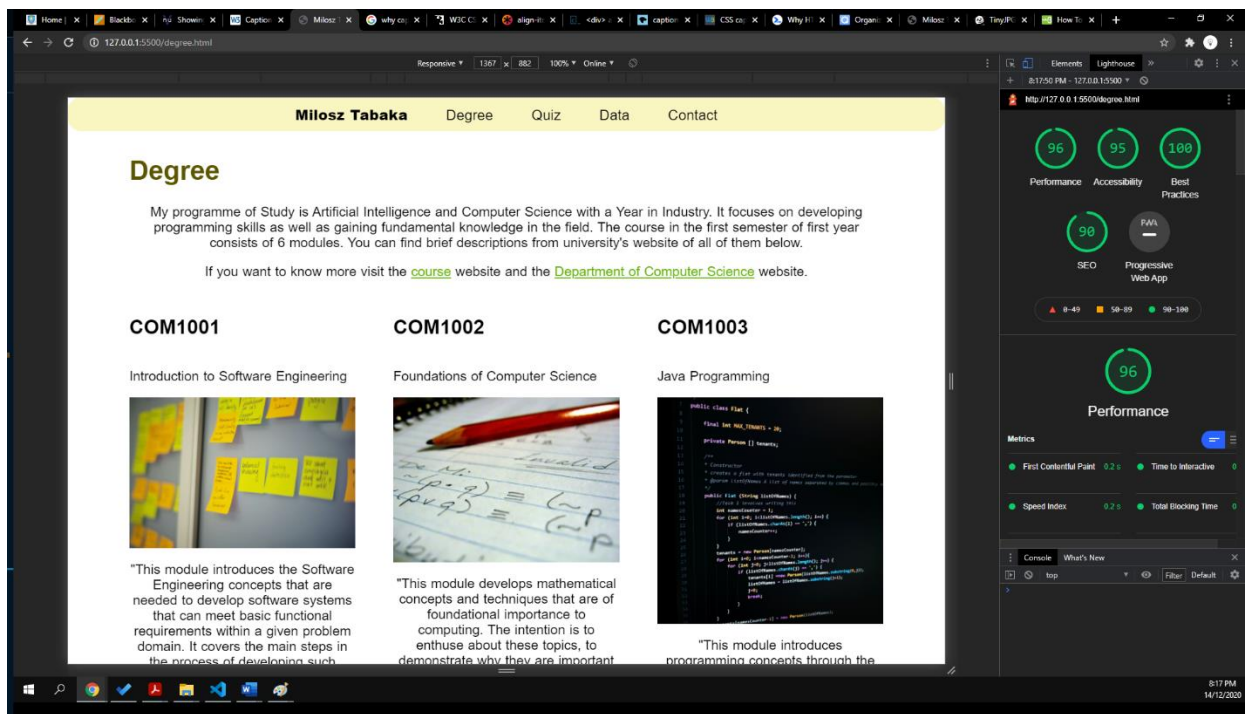


Figure 21 - Desktop Degree test

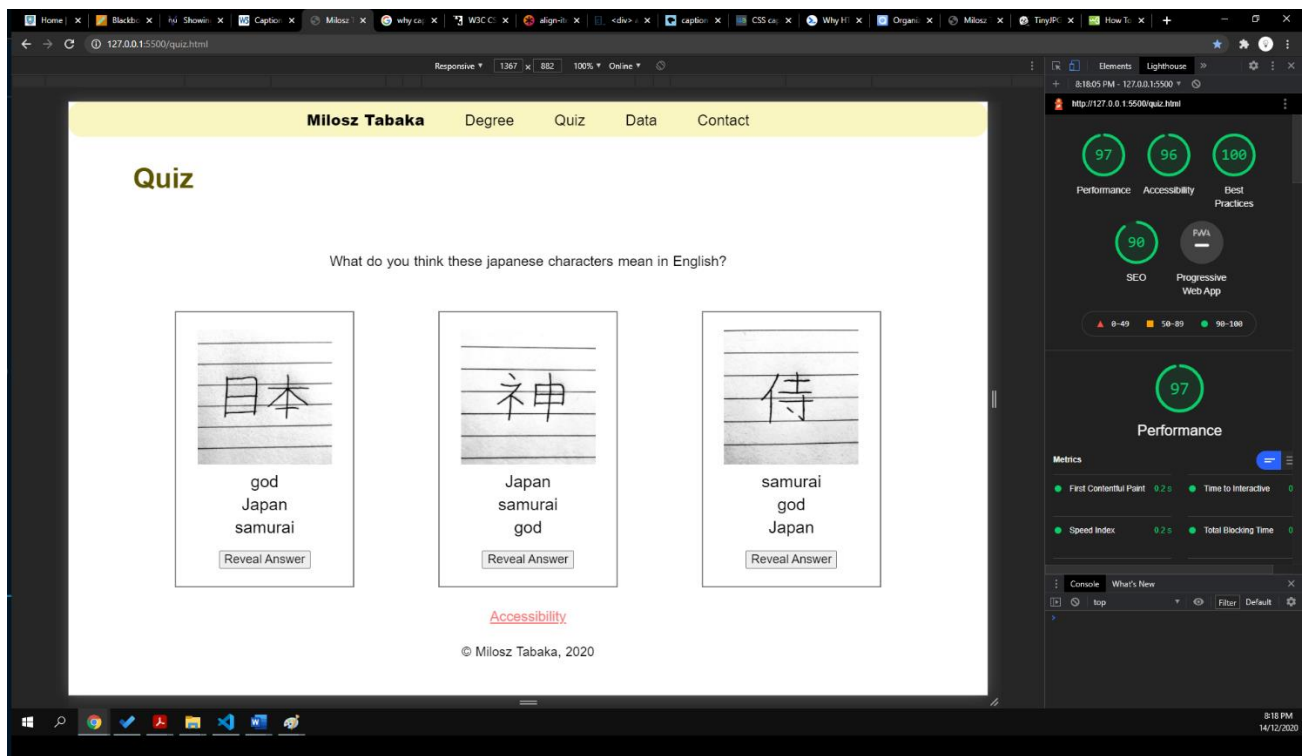


Figure 22 - Desktop Quiz test

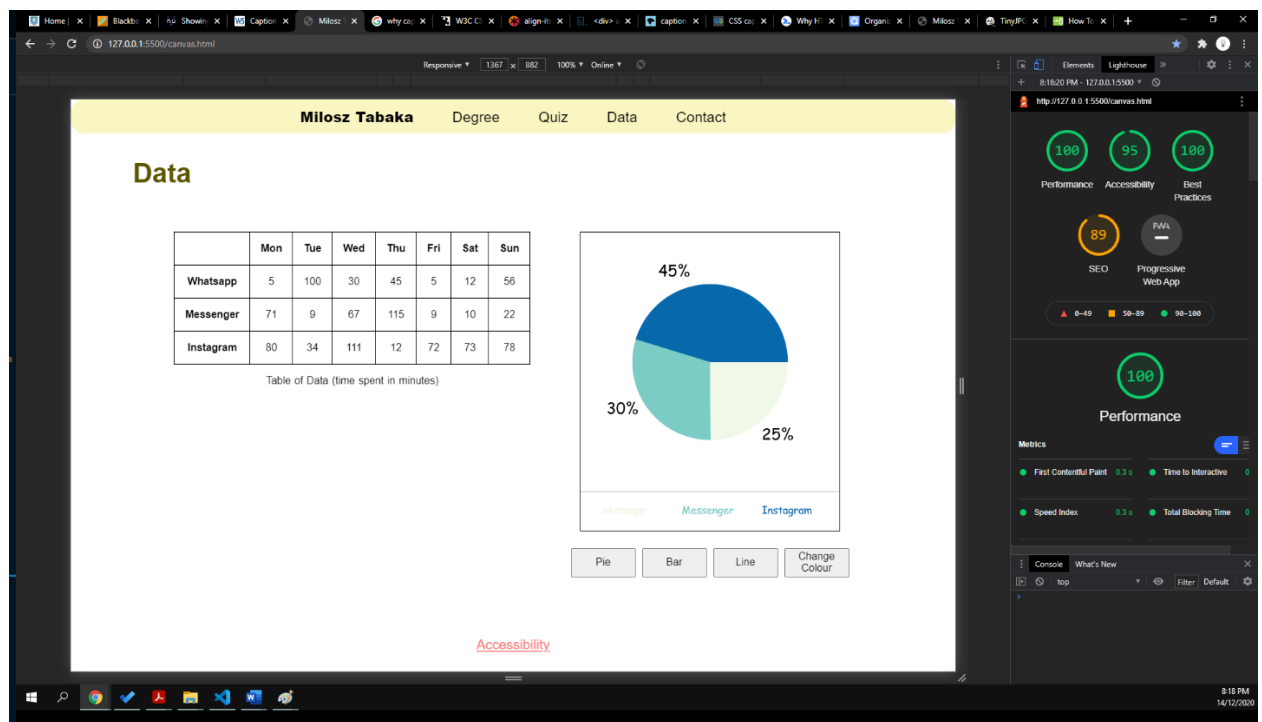


Figure 23 - Desktop Data test



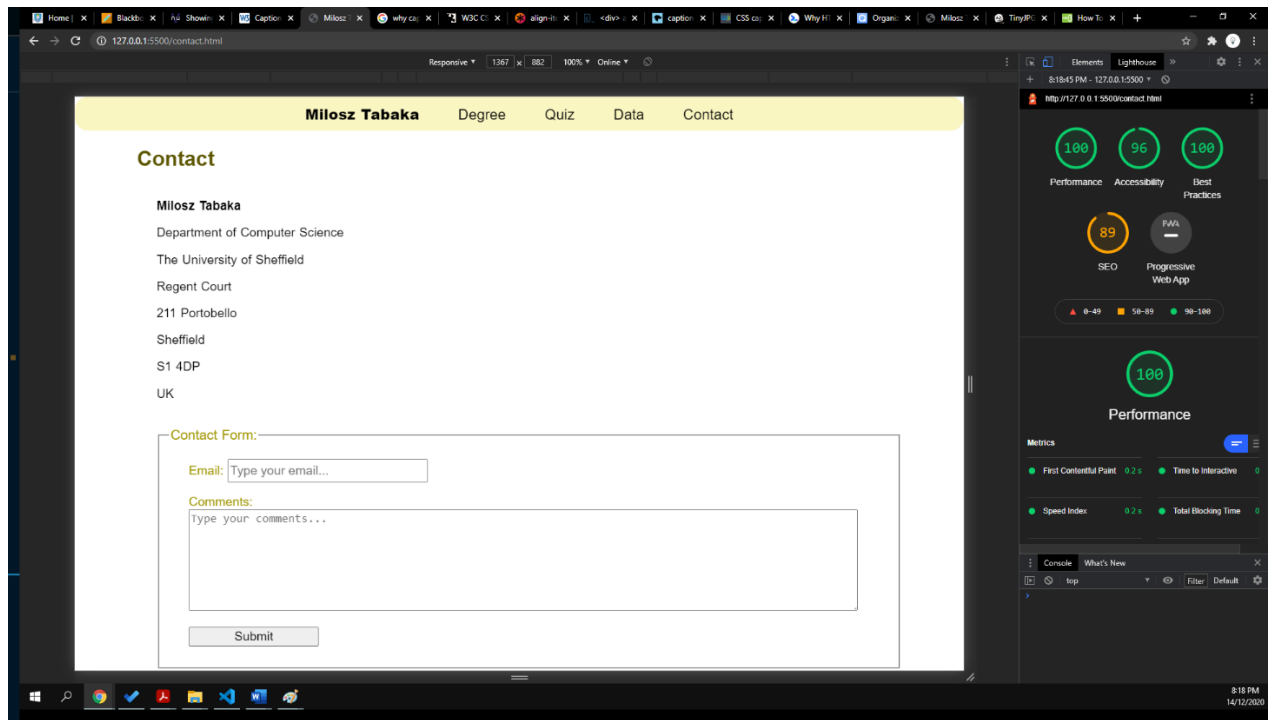


Figure 24 - Desktop Contact test

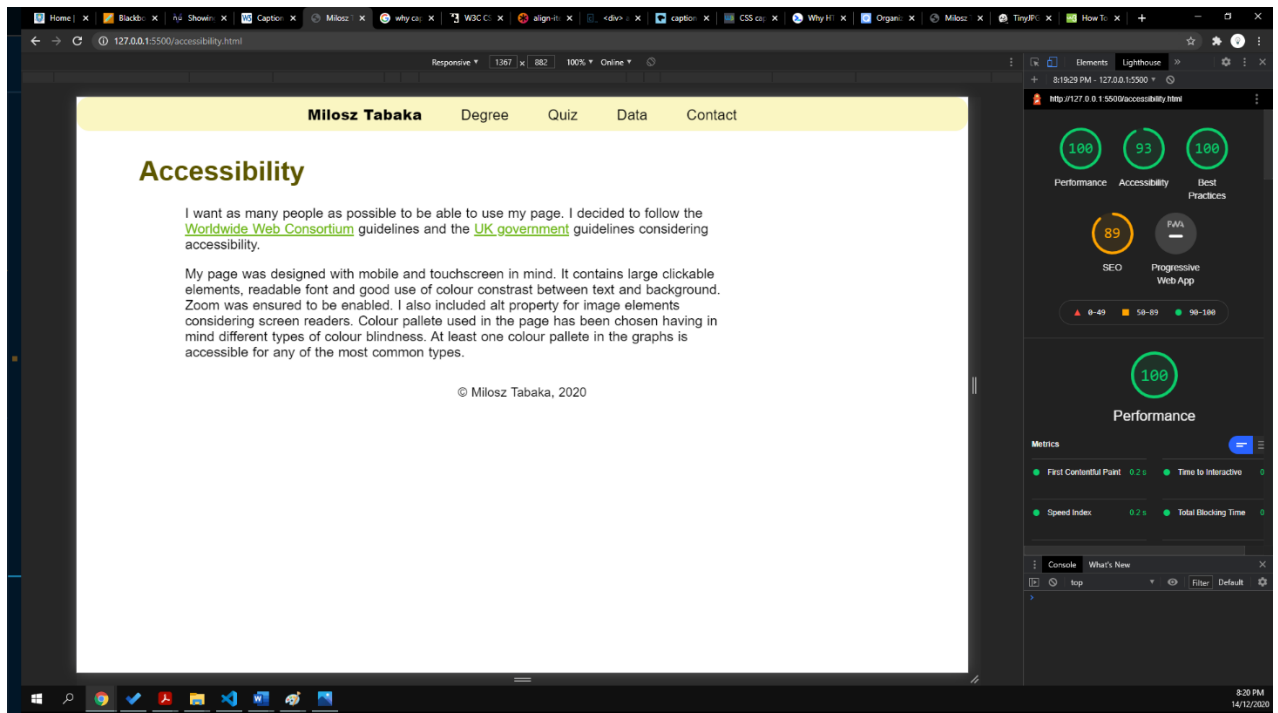


Figure 25 - Desktop Accessibility test

I also tested my website for colour blindness accessibility using Colorblinding Tool for Google Chrome. The results are visible especially good on canvas.

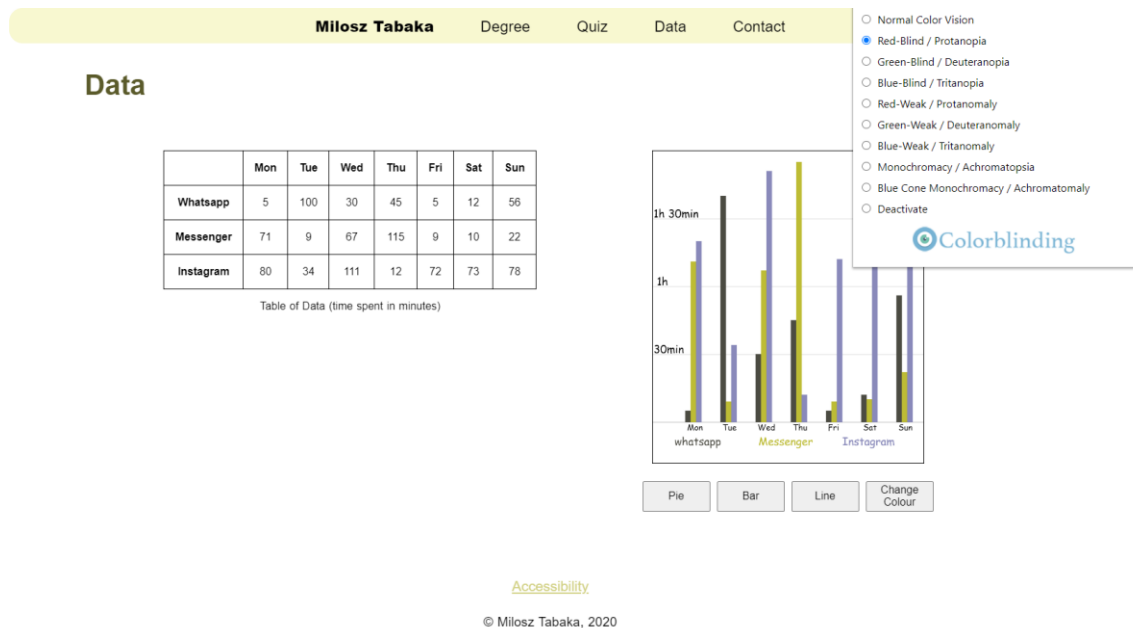


Figure 26 - Protanopia test

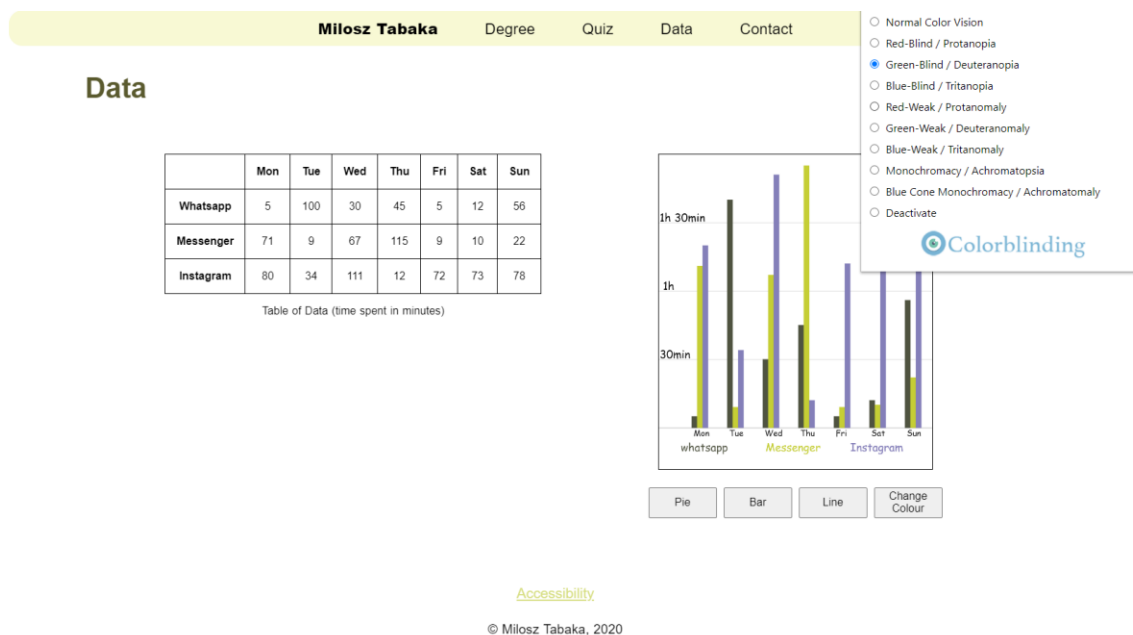
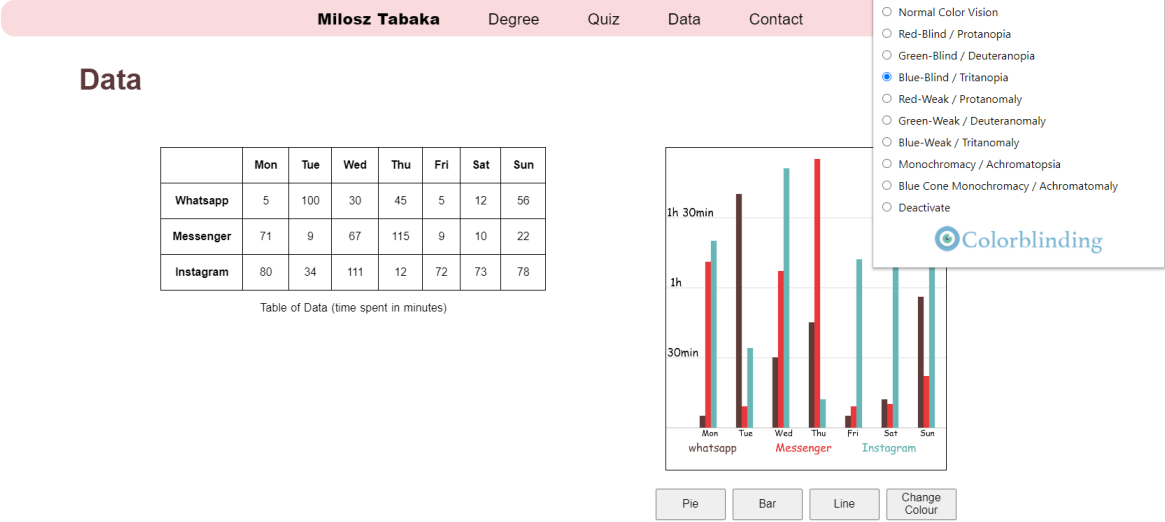


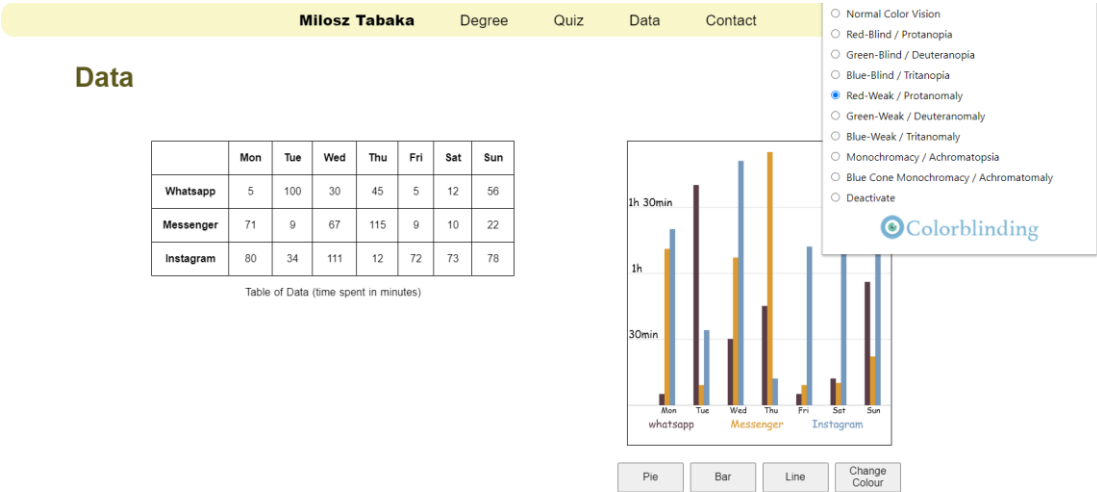
Figure 27 - Deuteranopia test



Accessibility

© Milosz Tabaka, 2020

Figure 28 – Tritanopia test



Accessibility

© Milosz Tabaka, 2020

Figure 29 - Protanomaly test

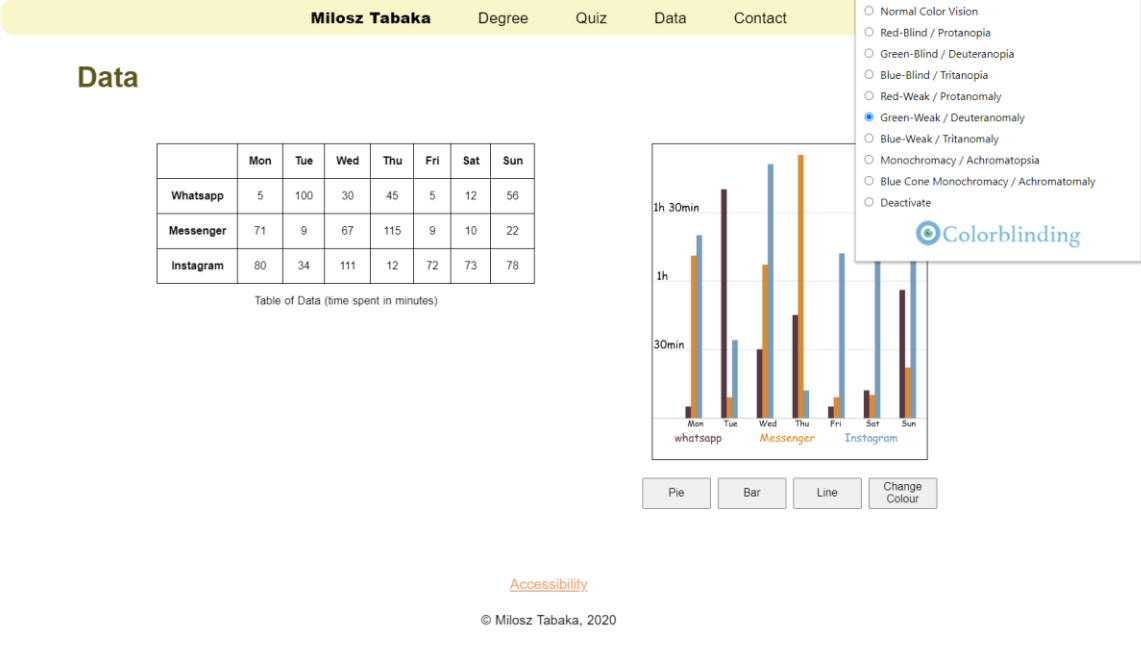


Figure 30 - Deuteranomaly test



Figure 31 - Tritanomaly test

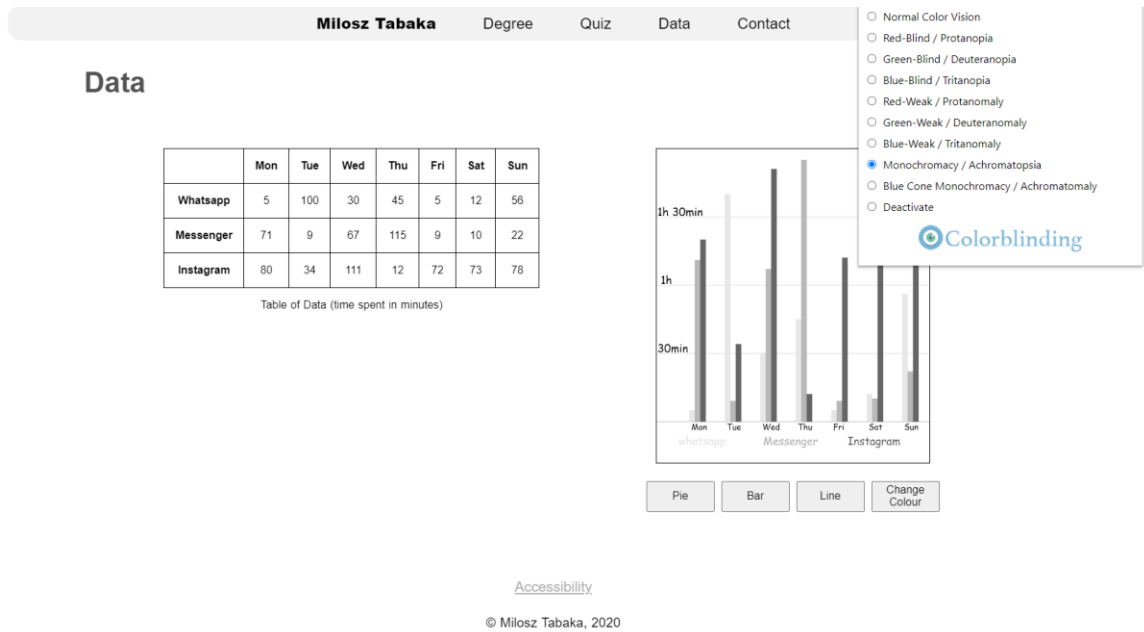


Figure 32 – Achromatopsia test

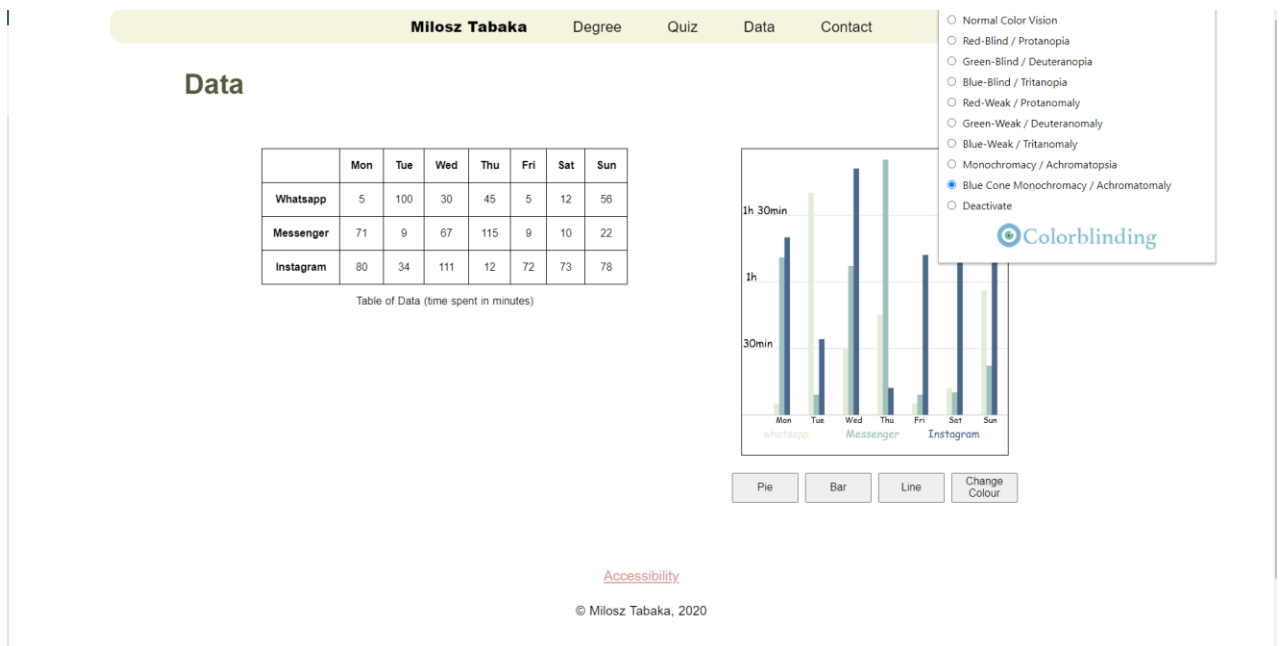


Figure 33 - Achromatomaly test