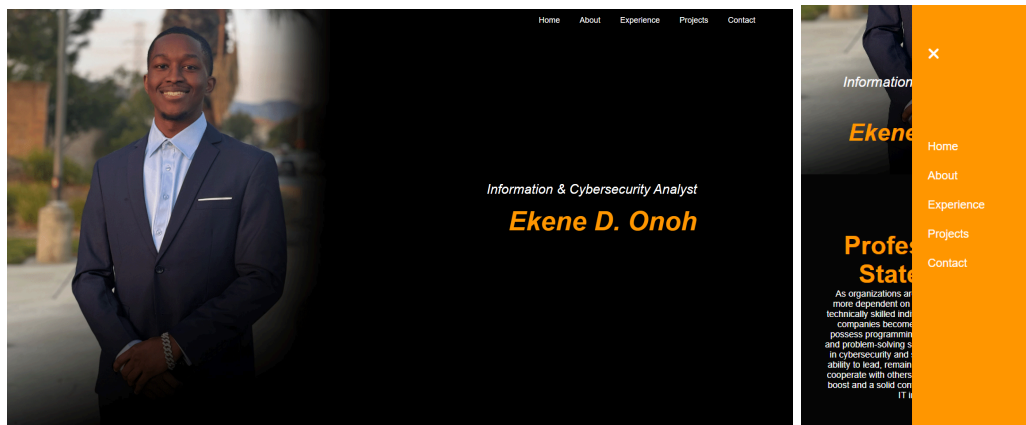


## **Building Portfolio Website**

This project consisted of creating a portfolio website from scratch using HTML for structure, CSS for styling, and JavaScript for interactivity. Additionally, Google Apps Script was used, for backend functionality to integrate the site with Google Spreadsheets. The site's purpose is to showcase my skills in web development and ability to integrate external services and APIs along with a hub to show my IT and cybersecurity projects and experience.



### **HTML Breakdown**

#### 1. Document Type Declaration and Head Section

The code starts with a standard HTML document structure, including meta tags for character encoding, viewport settings, and compatibility. It links to an external stylesheet for styling and includes a favicon. Additionally, it imports Font Awesome icons for use on the website. “<!DOCTYPE html>” is placed at the start of the file to declare the document type and version of HTML being used which is HTML 5. The file's root element starts with “<html>” and contains the whole HTML content. “<head>” contains meta information about the document, such as its title, character set, viewport settings, stylesheets, and scripts.

#### 2. Body Section

The body section contains the visible content of the page. The body of the website is divided into sections using <div> elements. The "header" section contains a navigation menu and a header text with Ekene's name and professional title. The "about" section includes a professional statement and tabs for experience, skills, and education. Each tab contains relevant information listed in an organized manner. “<div id="header">” contains the navigation menu

“<nav>” and header text. The navigation menu consists of a list of links “<ul>” and menu toggles using Font Awesome icons “<i class=“fa-solid fa-bars” onclick=“openmenu()”></i>” and “<i class=“fa-solid fa-xmark” onclick=“closemenu()”></i>”. The section containing the main content is divided into sections like About, Experience, Projects, and Contact, each with its own content. Tabs were used to switch between different content sections (“<div class=“tab-titles”>” and “<div class=“tab-contents”>”). The experience section holds information about professional experience, hackathons, and teaching/leadership roles. The skills section lists various technical and soft skills. The education section lists educational qualifications and certifications. The projects section displays projects with images, descriptions, and links to obtain more details. The contact section provides contact info and a form to submit messages.

### 3. Scripts

The area of the code that is relevant to the scripts contains JavaScript code for tab switching, side menu toggling, and form submission handling. The tab-switching script enables tab-switching functionality (“opentab”). The side menu script toggles the side menu (“openmenu” and “closemenu”). The form submission script handles form submission using Fetch API to send data to a Google Sheet (“submit-to-google-sheet”).

```
index.html x 1 READMe.md 2 style.css
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 <title>Ekene Onoh</title>
8 <link rel="stylesheet" href="style.css">
9 <link rel="icon" type="image/x-icon" href="/images/favicon.ico">
10 <script src="https://kit.fontawesome.com/3cccf3c3.js" crossorigin="anonymous"></script>
11 </head>
12 <body>
13 <div id="header">
14 <div class="container">
15 <nav>
16 
17 <ul id="sidemenu">
18 <li><a href="#header">Home</a></li>
19 <li><a href="#about">About</a></li>
20 <li><a href="#experience">Experience</a></li>
21 <li><a href="#projects">Projects</a></li>
22 <li><a href="#contact">Contact</a></li>
23 <li><i class="fa-solid fa-bars" onclick="openmenu()"></i>
24 </li>
25 </ul>
26 <i class="fa-solid fa-xmark" onclick="closemenu()"></i>
27 </nav>
28 <div class="header-text">
29 <p>Information & Cybersecurity Analyst</p>
30 <p>Ekene D. Onoh</p>
31 </div>
32 </div>
33 </div>
34 <!-- about -->
35 <div id="about">
36 <div class="container">
37 <div class="about">
38 <div class="about-col">
39 <h3 class="sub-title">Professional Statement</h3>
40 <p>As organizations around the globe become
41 more dependent on technology, the need for
42 technically skilled individuals who can aid these
43 companies becomes more blatant. While I
44 possess programming, networking, analytical,
45 and problem-solving skills along with experience
46 in cybersecurity and software engineering, my
47 ability to lead, remain optimistic, learn fast, and
48 cooperate with others allows me to be a morale
49 boost and a solid contributor to any team in the
50 IT industry.</p>
51 <p><small>© Ekene Onoh</small></p>
52 <div class="tab-titles" id="experience-section">
53 <p class="tab-links active-link" onclick="opentab('experience')">Experience</p>
54 <p class="tab-links" onclick="opentab('skills')">Skills</p>
55 <p class="tab-links" onclick="opentab('education')">Education & Certifications</p>
56 </div>
57 <div class="tab-contents active-tab" id="experience">
58 <ul>
59 <li><span>Cybersecurity Researcher - </li>
60 Apollo Information Systems</span></li>
61 <li>Worked with a team at AIS where I was tasked
62 with researching modern hacking tools,
63 collecting data about companies that have been
64 attacked, and assisting companies with
65 improving web security.</li>
66 </ul>
67 <li><span>Software Engineer - </li>
```

## CSS

The cascading stylesheet file included various controls that were used to assist in the visual design of the webpage.

### Margin

- Margin is the space outside the border of an element.
- It controls the gap between elements and is specified using various units like pixels, percentages, or em.

### Padding

- Padding is the space inside the border of an element.
- It controls the distance between the element's content and its border.

### Font-family

- Font-family specifies the typeface or font family of text in an element.
- It allows you to define a prioritized list of font family names to use.

### Box-sizing

- Box-sizing determines how the total width and height of an element are calculated.
- The default value is "content-box", which includes only the content, while "border-box" includes padding and border in the calculation.

### Scroll-behavior

- Scroll-behavior determines the scrolling behavior of the document when a user navigates to a different part of the page.
- It can be set to "smooth" to enable smooth scrolling.

### Background

- Background sets the background color or image of an element.
- It can be used to create visually appealing backgrounds for elements.

### Color

- Color sets the text color of an element's content.
- It can be specified using color names, hexadecimal values, RGB, or HSL values.

### Width

- Width sets the width of an element.

- It can be specified using various units like pixels, percentages, or em.

### Height

- Height sets the height of an element.
- Like width, it can be specified using different units.

### Background-image

- Background-image sets an image as the background of an element.
- It can be combined with other background properties to control the appearance of the background image.

### Background-size

- Background-size determines the size of the background image.
- It can be set to "cover" to scale the image as large as possible.

### Background-position

- Background-position sets the starting position of the background image.
- It can be used to position the image within the element.

### Display

- Display specifies the type of box used for an element.
- It can be set to "block", "inline", "inline-block", "flex", "grid", etc., to control the layout of the element.

### List-style

- List-style sets the style of the list marker (such as bullets or numbers) of a list item.
- It can be used to customize the appearance of lists.

### Align-items

- Align-items aligns flex items along the cross-axis of the flex container.
- It is used in flexbox layouts to control the vertical alignment of items.

### Justify-content

- Justify-content aligns flex items along the main axis of the flex container.
- It is used in flexbox layouts to control the horizontal alignment of items.

### Flex-wrap

- Flex-wrap specifies whether flex items should wrap or not if they exceed the container's width.
- It is used in flexbox layouts to control how items flow within the container.

### Text-decoration

- Text-decoration sets the decoration of text, such as underline, overline, or line-through.
- It can be used to add visual emphasis to text.

### Font-size

- Font-size sets the size of the font used for text.
- It can be specified using various units like pixels, percentages, or em.

### Position

- Position specifies the positioning method of an element.
- It can be set to "static", "relative", "absolute", "fixed", or "sticky" to control how the element is positioned within its parent.

### Content

- Content is used with the "::before" and "::after" pseudo-elements to insert content before or after an element's content.

### Left

- Left sets the distance between the left edge of a positioned element and the left edge of its containing element.
- It is used with "position: absolute" or "position: relative".

### Bottom

- Bottom sets the distance between the bottom edge of a positioned element and the bottom edge of its containing element.
- It is used with "position: absolute" or "position: relative".

### Transition

- Transition sets the transition effect to elements when their properties change.
- It allows you to control the speed and timing of the transition effect.

### Margin-top

- Margin-top sets the top margin of an element.
- It controls the space above the element and can be specified using various units.

### Margin-right

- Margin-right sets the right margin of an element.
- It controls the space to the right of the element and can be specified using various units.

### Text-align

- Text-align sets the horizontal alignment of text within an element.
- It can be set to "left", "center", "right", or "justify".

### Font-style

- Font-style sets the style of the font, such as normal, italic, or oblique.
- It can be used to add emphasis to text.

### Flex-basis

- Flex-basis specifies the initial size of a flex item along the main axis before any remaining space is distributed.
- It is used in flexbox layouts.

### Font-weight

- Font-weight sets the thickness of the font.
- It can be set to "normal", "bold", "lighter", or "bolder".

### Cursor

- Cursor specifies the type of cursor to display when the mouse pointer is over an element.
- It can be set to "pointer", "default", "text", etc.

### Grid-template-columns

- Grid-template-columns sets the number and size of columns in a grid layout.
- It is used to create grid layouts with specified column sizes.

### Grid-gap

- Grid-gap sets the gap between grid rows and columns in a grid layout.
- It is used to create space between grid items.

### Border-radius

- Border-radius sets the radius of the border corners of an element.
- It can be used to create rounded corners.

## Overflow

- Overflow specifies what should happen if the content of an element overflows its box.
- It can be set to "visible", "hidden", "scroll", or "auto".

## Flex-direction

- Flex-direction specifies the direction in which flex items are placed in the flex container. It can be set to "row", "row-reverse", "column", or "column-reverse".

## Line-height

- Line-height sets the height of a line of text.
- It can be specified using various units like pixels, percentages, or em.

## Z-index

- Z-index specifies the stack order of an element.
- It determines which elements are displayed in front of or behind other elements.
- Elements with a higher z-index value are displayed in front of elements with a lower z-index value.

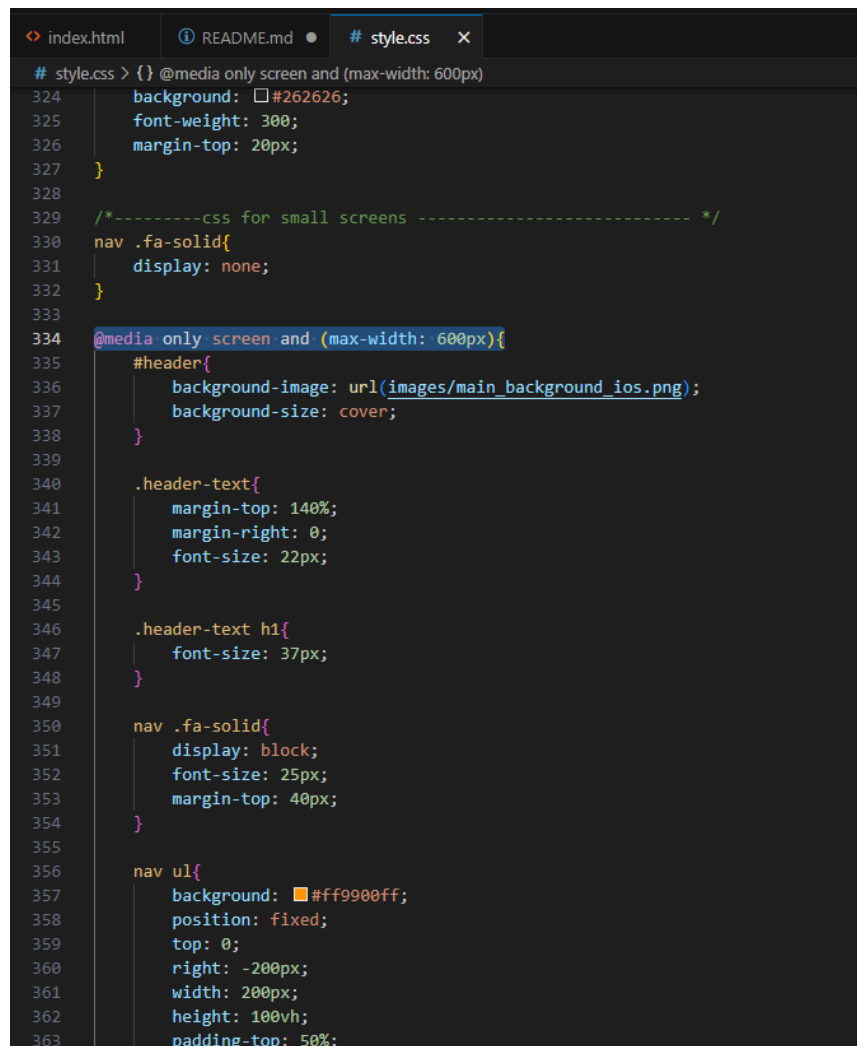
```
1  index.html  10 README.md  11 styles.css  X
2  # styles.css > {} @media only screen and (max-width: 600px)
3  {
4    margin: 0;
5    padding: 0;
6    font-family: 'Poppins', sans-serif;
7    box-sizing: border-box;
8  }
9
10 html{
11   scroll-behavior: smooth;
12 }
13
14 body{
15   background: #f0f0f0;
16   color: #fff;
17 }
18
19 #header{
20   width: 100%;
21   height: 100px;
22   background-image: url(images/main_background.png);
23   background-size: cover;
24   background-position: left;
25 }
26
27 .container{
28   padding: 10px 10%; /*10 pixels from the top and 90 from the left (10% right)*/
29 }
30
31 nav{
32   display: flex;
33   align-items: right;
34   justify-content: space-between;
35   flex-wrap: wrap;
36 }
37
38 .logo{
39   width: 140px;
40 }
41
42 nav ul li{
43   display: inline-block;
44   list-style: none;
45   margin: 10px 20px;
46 }
47
48 nav ul li a{
49   color: #fff;
50   text-decoration: none;
51   font-size: 18px;
52   position: relative;
53 }
54
55 nav ul li a::after{
56   content: '';
57   width: 0;
58   height: 3px;
59   background: #ff0000;
60   position: absolute;
61   left: 0;
62   bottom: -6px;
63   transition: 0.5s;
64 }
65
66 nav ul li a:hover::after{
67   width: 100%;
68 }
```

## Mobile Compatibility

The `@media` rule in CSS is used to apply different styles for different media types or device characteristics. In this case, `@media only screen and (max-width: 600px){}` is a media query that targets screens with a maximum width of 600px, which typically includes smaller devices like mobile phones.

By using this media query, CSS styles within the curly braces `{}` will only apply when the screen width is 600px or less. This allows developers to create responsive designs that adapt to different screen sizes, ensuring a better user experience on various devices.

For example, within this media query, you might adjust the layout, font sizes, or padding to make the content more readable and accessible on smaller screens. This separation of styles for different screen sizes helps ensure that your website looks and functions well across a range of devices, from desktop computers to smartphones.

A screenshot of a code editor with three tabs: 'index.html', 'README.md', and '# style.css'. The '# style.css' tab is active, showing CSS code. The code includes a media query `@media only screen and (max-width: 600px){}` that applies styles for screens 600px wide or less. The styles include background color, font weight, margin-top, display: none, background image, background size, margin-top, margin-right, font-size, display: block, font-size, margin-top, background color, position: fixed, top, right, width, height, and padding-top.

```
# style.css > {} @media only screen and (max-width: 600px)
324     background: #262626;
325     font-weight: 300;
326     margin-top: 20px;
327 }
328
329 /*-----css for small screens ----- */
330 nav .fa-solid{
331     display: none;
332 }
333
334 @media only screen and (max-width: 600px){
335     #header{
336         background-image: url(images/main_background_ios.png);
337         background-size: cover;
338     }
339
340     .header-text{
341         margin-top: 140%;
342         margin-right: 0;
343         font-size: 22px;
344     }
345
346     .header-text h1{
347         font-size: 37px;
348     }
349
350     nav .fa-solid{
351         display: block;
352         font-size: 25px;
353         margin-top: 40px;
354     }
355
356     nav ul{
357         background: #ff9900ff;
358         position: fixed;
359         top: 0;
360         right: -200px;
361         width: 200px;
362         height: 100vh;
363         padding-top: 50%;
```



## The Contact Section

```
1  var sheetName = 'Sheet1'
2  var scriptProp = PropertiesService.getScriptProperties()
3
4  function initialSetup () {
5      var activeSpreadsheet = SpreadsheetApp.getActiveSpreadsheet()
6      scriptProp.setProperty('key', activeSpreadsheet.getId())
7  }
8
9  function doPost (e) {
10     var lock = LockService.getScriptLock()
11     lock.tryLock(10000)
12
13     try {
14         var doc = SpreadsheetApp.openById(scriptProp.getProperty('key'))
15         var sheet = doc.getSheetByName(sheetName)
16
17         var headers = sheet.getRange(1, 1, 1, sheet.getLastColumn()).getValues()[0]
18         var nextRow = sheet.getLastRow() + 1
19
20         var newRow = headers.map(function(header) {
21             return header === 'timestamp' ? new Date() : e.parameter[header]
22         })
23
24         sheet.getRange(nextRow, 1, 1, newRow.length).setValues([newRow])
25
26         return ContentService
27             .createTextOutput(JSON.stringify({ 'result': 'success', 'row': nextRow }))
28             .setMimeType(ContentService.MimeType.JSON)
29     }
30
31     catch (e) {
32         return ContentService
33             .createTextOutput(JSON.stringify({ 'result': 'error', 'error': e }))
34             .setMimeType(ContentService.MimeType.JSON)
35     }
36
37     finally {
38         lock.releaseLock()
39     }
40 }
```

Using JavaScript and the open-source Google Apps Script code from Jamie Wilson allows website users to submit their name, email, and message through a form on the website. The form submission is then processed by Google Apps Script and stored in a Google Spreadsheet.

The HTML code defines a form with the id "submit-to-google-sheet" and includes fields for the user's name, email, and message. The form also includes a submit button.

The JavaScript code uses the Fetch API to send a POST request to the specified Google Apps Script URL (scriptURL) when the form is submitted. The request includes the form data as a FormData object. If the request is successful, a success message is displayed to the user, and the form is reset. If an error occurs, the error message is logged to the console.

The Google Apps Script code defines a function called doPost() that is called when the POST request is received. The function first locks the script to prevent concurrent access. It then opens the specified Google Spreadsheet and retrieves the sheet named "Sheet1". It gets the headers of the sheet (the first row) to determine the column names. It then constructs a new row of data using the form data submitted by the user, with the timestamp set to the current date and time. The new row is appended to the end of the sheet, and a successful response is returned to the client.

If an error occurs during the process, an error response is returned instead. Finally, the script releases the lock to allow other processes to access the script.

## **Conclusion & Lessons Learned**

I was able to gain more experience and comfort with HTML and CSS along with the many intricacies of the features each language consists of. The project also introduced me to Google Apps Script. I was so intrigued by its resourcefulness that I hope to utilize it again sometime in the future.

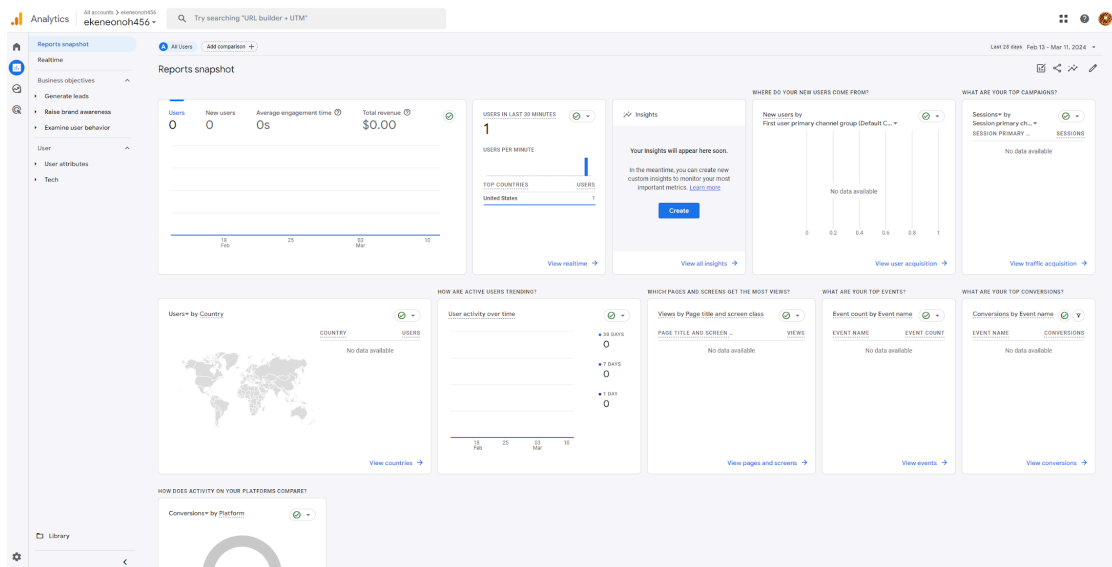
There were multiple moments where I attempted to be creative and experimental with the page design which led to many instances of either glitches, bugs, or unwanted results. However, I did my research to find solutions and overcome these obstacles. In addition to finding solutions, there were also times when removing an idea or component of the code was the best option. It is important to sometimes work smarter and not harder. The best solutions are not always the overcomplicated ones. Since I will be constantly updating the page, there will be more opportunities to grow and develop not only this site but also my knowledge of web development.

## UPDATE MARCH 13, 2024

### Google Analytics Implementation

Google Analytics is a web analytics service offered by Google that helps website owners and marketers track and analyze their website traffic. It gives valuable insights into how users interact with a website, allowing businesses and organizations to make informed decisions.

One of the key features of Google Analytics is its ability to track various metrics related to website traffic. This includes the number of visitors, page views, bounce rate, average session duration, etc. These metrics help website owners understand how users are engaging with their site and identify areas for improvement. It also provides demographic and interest data about website visitors, such as their age, gender, interests, and geographic location. This information can be used to tailor content and advertising to better target specific audience segments. Another important feature of Google Analytics is its goal-tracking capabilities. Website owners can set up goals, such as making a purchase or completing a contact form, and track the percentage of users who complete these goals. This helps businesses measure the effectiveness of their website in achieving desired outcomes.



To enable Google Analytics to work with a website, a script provided by Google Analytics needs to be inserted into the website's HTML code. This script, typically provided in the form of a JavaScript snippet, is responsible for collecting data about user interactions on the website and sending it to Google Analytics for analysis. The script includes a unique tracking ID that associates the data with the specific Google Analytics account. Once the script is inserted into the website's code, Google Analytics

starts tracking visitor activity, providing valuable insights into how users engage with the site.

Overall, Google Analytics is a powerful tool for understanding and optimizing website performance. Its comprehensive set of features provides valuable insights that can help businesses attract more visitors and improve user experience.