Themes and Purpose

Website Theme: Fitness and Health Resource

Purpose: The website is designed to be a comprehensive resource for fitness enthusiasts, offering a wide array of features aimed at supporting users in their journey towards better health and fitness. The primary purpose is to provide tailored workout routines, dietary advice, and motivational content to keep users inspired and committed to their fitness goals.

The website will include dynamic content such as progress trackers and workout generators, which use JavaScript to store user data locally in the browser. This approach ensures user privacy and simplifies the implementation by avoiding server-side data processing. Additionally, the site will offer a blog section with regularly updated articles on fitness trends, nutrition tips, and motivational stories to keep users engaged.

By sharing personal fitness experiences and achievements, the website aims to motivate and inspire its visitors. It will also feature tools like BMI and one-rep max calculators to help users better understand their fitness levels and progress. Overall, the website aims to provide a holistic and user-friendly fitness resource accessible across multiple devices and browsers.

Personas

Persona 1: John Doe

- **Age:** 28
- **Gender:** Male
- **Occupation:** Software Developer
- **Interests:** Technology, fitness, running
- **Goals and Motives:** John is looking to balance his sedentary job with a healthy lifestyle. He wants quick, effective workout routines that fit into his busy schedule.
- **Approach to Using the Site:** John prefers structured, easy-to-follow workout plans and enjoys tracking his progress digitally.

Persona 2: Mary Smith

- **Age:** 35
- **Gender:** Female
- **Occupation:** Stay-at-home Mom
- **Interests:** Family activities, cooking, fitness
- **Goals and Motives:** Mary aims to stay fit while managing her household responsibilities. She needs flexible workout routines that she can do at home.

- **Approach to Using the Site:** Mary likes a mix of structured and spontaneous workout routines. She often looks for dietary advice and motivational tips.

Persona 3: Ahmed Khan

- **Age:** 42
- **Gender:** Male
- **Occupation:** Marketing Manager
- **Interests:** Business, sports, healthy living
- **Goals and Motives:** Ahmed is looking to maintain his health and fitness despite his demanding job. He seeks workout routines that can relieve stress and keep him energized.
- **Approach to Using the Site:** Ahmed prefers exploring different workout routines and is interested in the latest fitness trends and nutritional advice.

Persona 4: Lisa Wong

- **Age:** 24
- **Gender:** Female
- **Occupation:** College Student
- **Interests:** Yoga, vegetarianism, mindfulness
- **Goals and Motives:** Lisa wants to incorporate fitness into her daily routine to complement her yoga practice. She seeks routines that promote flexibility and mindfulness.
- **Approach to Using the Site:** Lisa looks for holistic fitness approaches and enjoys reading articles about wellness and nutrition.

Persona 5: Robert Johnson

- **Age:** 55
- **Gender:** Male
- **Occupation:** Retired
- **Interests:** Gardening, walking, health
- **Goals and Motives:** Robert aims to maintain his health and mobility as he ages. He looks for low-impact workout routines and health advice tailored for older adults.
- **Approach to Using the Site:** Robert prefers simple, easy-to-navigate sites with clear instructions and accessible content.

Scenarios

Scenario 1:

- **Persona:** John Doe

- **Scenario:** John visits the site to find a quick 20-minute workout he can do before heading to work. He uses the workout generator to find a routine that fits his time constraints.

Scenario 2:

- **Persona:** John Doe
- **Scenario:** John uses the progress tracker to monitor his fitness improvements over the past month, viewing charts of his workout completion and progress.

Scenario 3:

- **Persona:** Mary Smith
- **Scenario:** Mary accesses the website to find home-based workout routines she can perform while her children are at school. She uses the blog section for dietary tips.

Scenario 4:

- **Persona:** Mary Smith
- **Scenario:** Mary reads motivational stories on the website to stay inspired and committed to her fitness journey, finding relatable content that keeps her motivated.

Scenario 5:

- **Persona:** Ahmed Khan
- **Scenario:** Ahmed visits the site during his lunch break to find stress-relieving workout routines. He browses through articles on the latest fitness trends.

Scenario 6:

- **Persona:** Ahmed Khan
- **Scenario:** Ahmed uses the BMI calculator to assess his body mass index and reads articles on how to maintain a healthy weight through balanced nutrition.

Scenario 7:

- **Persona:** Lisa Wong
- **Scenario:** Lisa looks for yoga routines that she can incorporate into her daily practice. She finds a series of videos and articles that help her improve her flexibility and mindfulness.

Scenario 8:

- **Persona:** Lisa Wong
- **Scenario:** Lisa reads the blog for vegetarian recipes that can complement her fitness routine and provide the necessary nutrients for her lifestyle.

Scenario 9:

- **Persona:** Robert Johnson
- **Scenario:** Robert searches for low-impact workout routines to maintain his mobility. He finds a series of videos and written guides that are easy to follow and suitable for his fitness level.

Scenario 10:

- **Persona:** Robert Johnson
- **Scenario:** Robert reads articles on the website about healthy aging and finds advice on how to adjust his diet and exercise routine to stay fit and active.

Further Requirments:

Adjustments Based on Requirements

1. **Technical Constraints:**
    - **Dynamic Content:** Use JavaScript for local storage of user progress and preferences. This avoids the need for server-side processing and ensures user privacy.
    - **Device Compatibility:** Ensure the site is responsive, working seamlessly on desktops, tablets, and smartphones. Use CSS media queries to adjust the layout for different screen sizes.
2. **Legal and Social Constraints:**
    - **Privacy Compliance:** Since no personal data will be stored on servers, the site will use local storage to keep user data private and secure.
    - **Inclusivity:** Ensure the website design and content are accessible to people with disabilities by following web accessibility standards (e.g., using ARIA roles, providing alt text for images).
    - **Content Sensitivity:** Avoid controversial topics and ensure that all content is respectful and non-offensive.
3. **Organizational Constraints:**
    - **Corporate Style:** Adhere to a clean and professional design that aligns with common corporate styles. Use a consistent color scheme and typography.
    - **Budget Considerations:** Utilize free or low-cost tools for design and development. Focus on creating a scalable website that can be easily updated without significant costs.
    - **Sustainability:** Design the site to be easily maintainable, with clear documentation and a simple structure that allows for future updates and expansions.

New Personas and Scenarios

Persona 6: Sarah Green

- **Age:** 30
- **Gender:** Female
- **Occupation:** Graphic Designer
- **Interests:** Creative arts, healthy living, mobile fitness apps
- **Goals and Motives:** Sarah wants to integrate fitness into her daily routine using her iPad. She seeks visually appealing workout routines that she can follow during breaks.
- **Approach to Using the Site:** Sarah prefers content that is visually engaging and easy to navigate on her iPad. She likes to follow along with videos and interactive content.

Scenario 11:

- **Persona:** Sarah Green
- **Scenario:** Sarah accesses the site on her iPad during her lunch break to find a 15-minute yoga session. She uses the site's video library to follow along with a guided session.

Scenario 12:

- **Persona:** Sarah Green
- **Scenario:** Sarah reads articles on creative ways to stay fit while maintaining a busy work schedule. She finds tips on integrating short workouts into her daily routine.

Persona 7: Michael Brown

- **Age:** 50
- **Gender:** Male
- **Occupation:** High School Teacher
- **Interests:** Hiking, reading, technology
- **Goals and Motives:** Michael is looking to stay active and healthy. He uses his smartphone to find quick and effective workout routines that he can do at home or while traveling.
- **Approach to Using the Site:** Michael prefers straightforward, easy-to-follow routines that he can access quickly on his smartphone. He appreciates clear instructions and minimalistic design.

Scenario 13:

- **Persona:** Michael Brown
- **Scenario:** Michael uses his smartphone to access the site and find a 10-minute workout he can do before starting his day. He looks for routines that require minimal equipment.

Scenario 14:

- **Persona:** Michael Brown

- **Scenario:** Michael reads an article on the site about the benefits of short, high-intensity workouts. He finds a list of suggested routines and decides to try one out.

Persona 8: Emily Clark

- **Age:** 27
- **Gender:** Female
- **Occupation:** Freelance Writer
- **Interests:** Running, cooking, traveling
- **Goals and Motives:** Emily wants to maintain her fitness while traveling. She needs portable workout routines and dietary advice that she can access on her laptop.
- **Approach to Using the Site:** Emily prefers routines that she can do in small spaces with minimal equipment. She enjoys reading about healthy recipes and fitness tips that she can incorporate into her travel routine.

Scenario 15:

- **Persona:** Emily Clark
- **Scenario:** Emily accesses the site on her laptop from a hotel room to find a 20-minute workout routine. She uses the workout generator to customize a routine that fits her available space and equipment.

Scenario 16:

- **Persona:** Emily Clark
- **Scenario:** Emily reads the blog section for new healthy recipes that she can try while on the road. She finds a recipe for a nutritious, easy-to-make meal and decides to incorporate it into her diet.

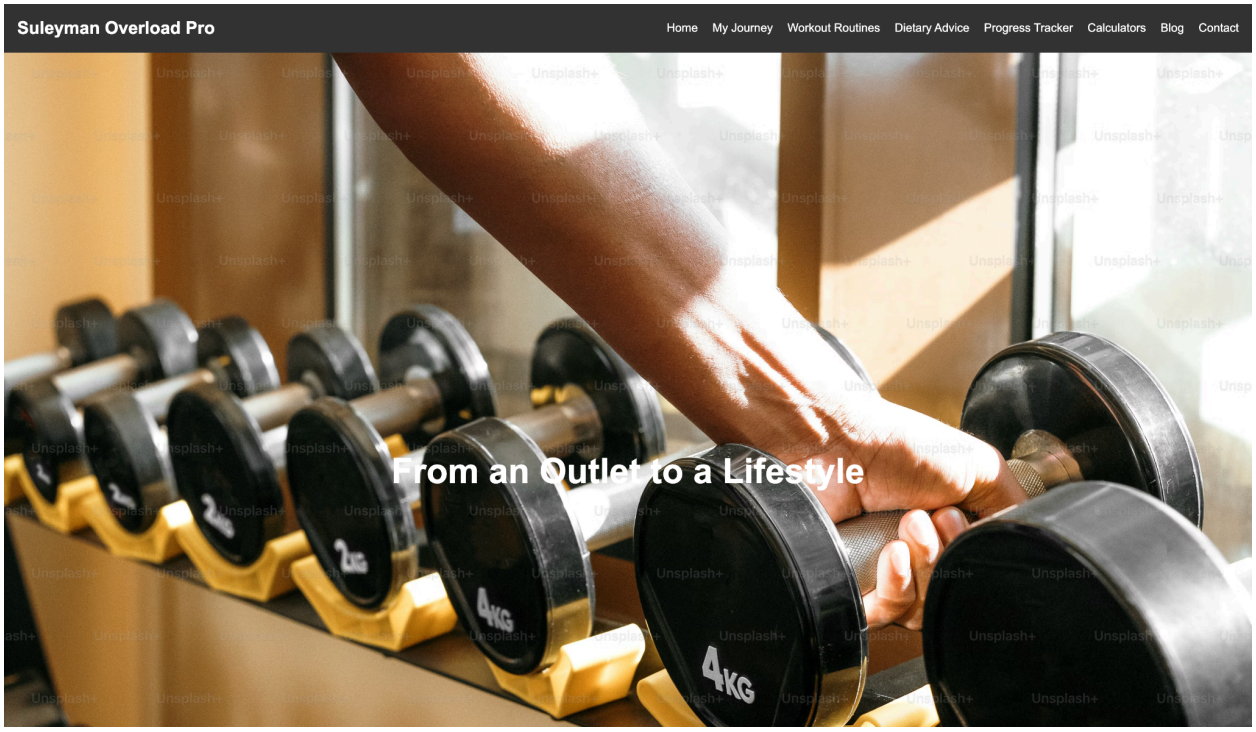
Mockup

Mockup: Create simple sketches of the main page and major sub-pages. Include major headings, menu items, and placeholders for content like workout routines, blog articles, and calculators. Use a tool like Balsamiq or Adobe XD to create a visual representation.

Home Page

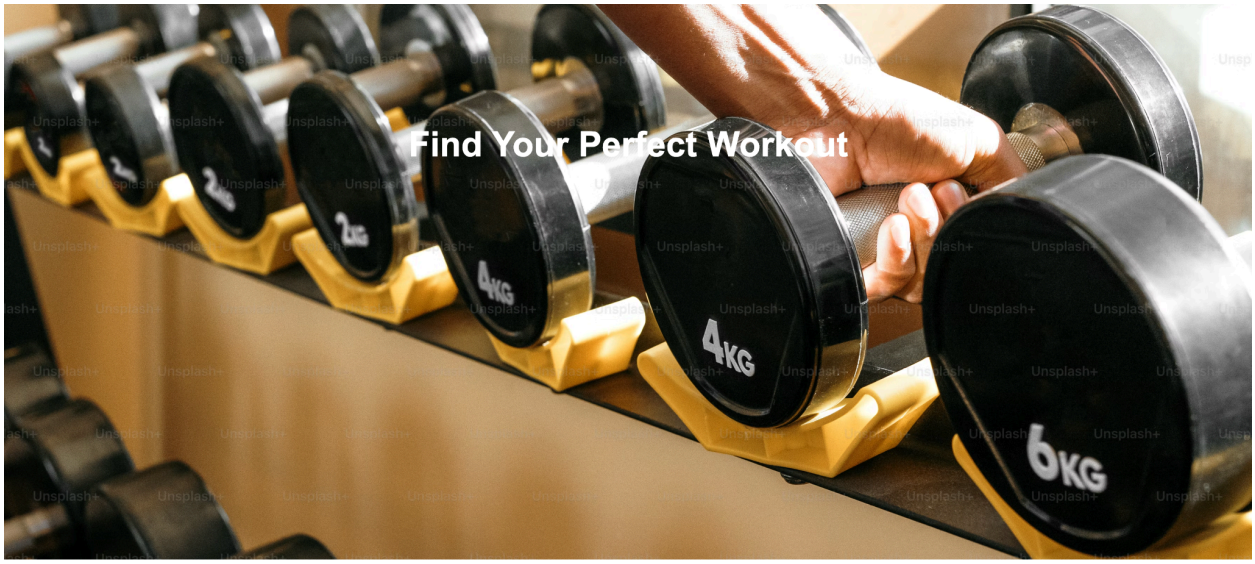
- **Header**: Logo, Navigation Menu (Home, Workout Routines, Dietary Advice, Progress Tracker, Motivational Tips, My Journey, Calculators, Blog, Contact)
- **Hero Section**: Engaging image, tagline, call-to-action button (Get Started)
- **Featured Articles**: Thumbnails and snippets of latest articles
- **Workout Routines**: Quick links to Beginner, Intermediate, Advanced sections
- **Dietary Advice**: Highlights of latest nutrition tips and recipes
- **Progress Tracker**: Brief introduction and link to track progress
- **Motivational Stories**: Featured stories and links to read more

- **Footer**: Links to social media, privacy policy, terms of service



Workout Routines Page

- **Header**: Consistent with Home Page
- **Hero Section**: Motivational image with overlay text (e.g., "Find Your Perfect Workout")
- **Beginner Section**: Thumbnails and descriptions of beginner routines
- **Intermediate Section**: Thumbnails and descriptions of intermediate routines
- **Advanced Section**: Thumbnails and descriptions of advanced routines
- **Footer**: Consistent with Home Page



Beginner Workouts

Intermediate Workouts

Advanced Workouts

Dietary Advice Page

- **Header**: Consistent with Home Page
- **Hero Section**: Image with overlay text (e.g., "Fuel Your Body Right")
- **Nutrition Tips**: Latest articles and tips
- **Recipes**: Thumbnails and brief descriptions of featured recipes
- **Footer**: Consistent with Home Page

Nutrition Tips

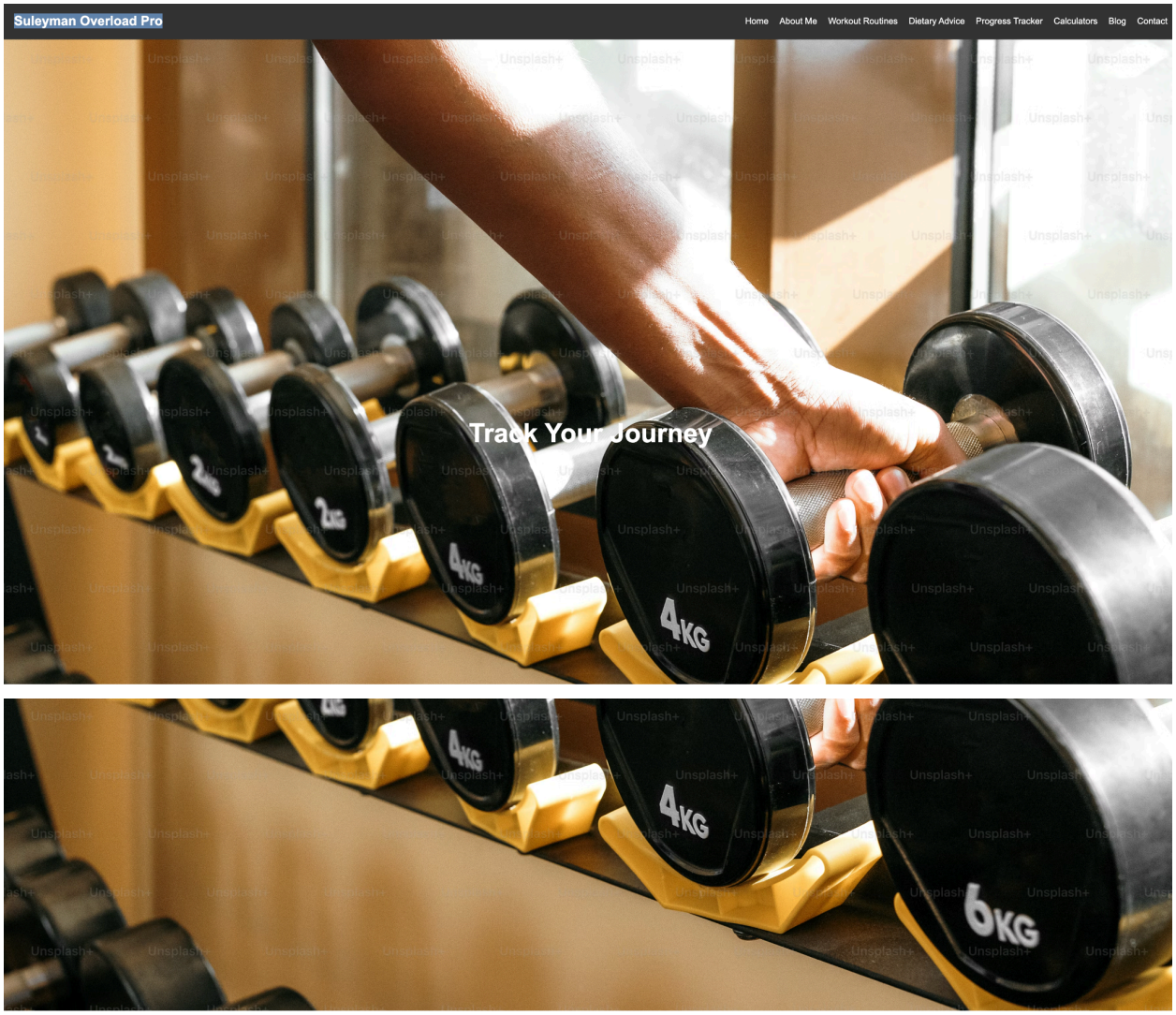
Recipes

Privacy Policy Terms of Service

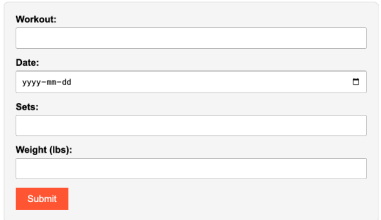
Progress Tracker Page

- **Header**: Consistent with Home Page
- **Hero Section**: Image with overlay text (e.g., "Track Your Journey")
- **Track Progress**: Form and input fields for tracking workout progress
- **Charts**: Visual representation of progress data

- **Footer**: Consistent with Home Page



Track Progress

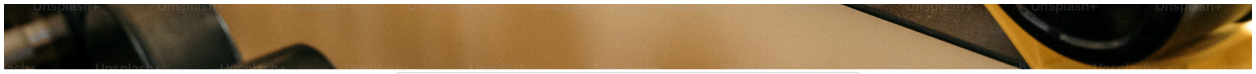


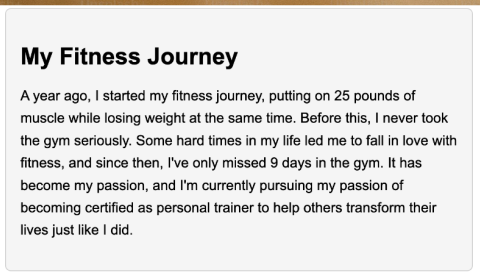
Progress Charts

My Journey Page

- **Header**: Consistent with Home Page
- **Hero Section**: Image with overlay text (e.g., "Share Your Journey")
- **Personal Achievements**: Form to input personal achievements

- **Gallery**: Image gallery of user achievements
- **Footer**: Consistent with Home Page



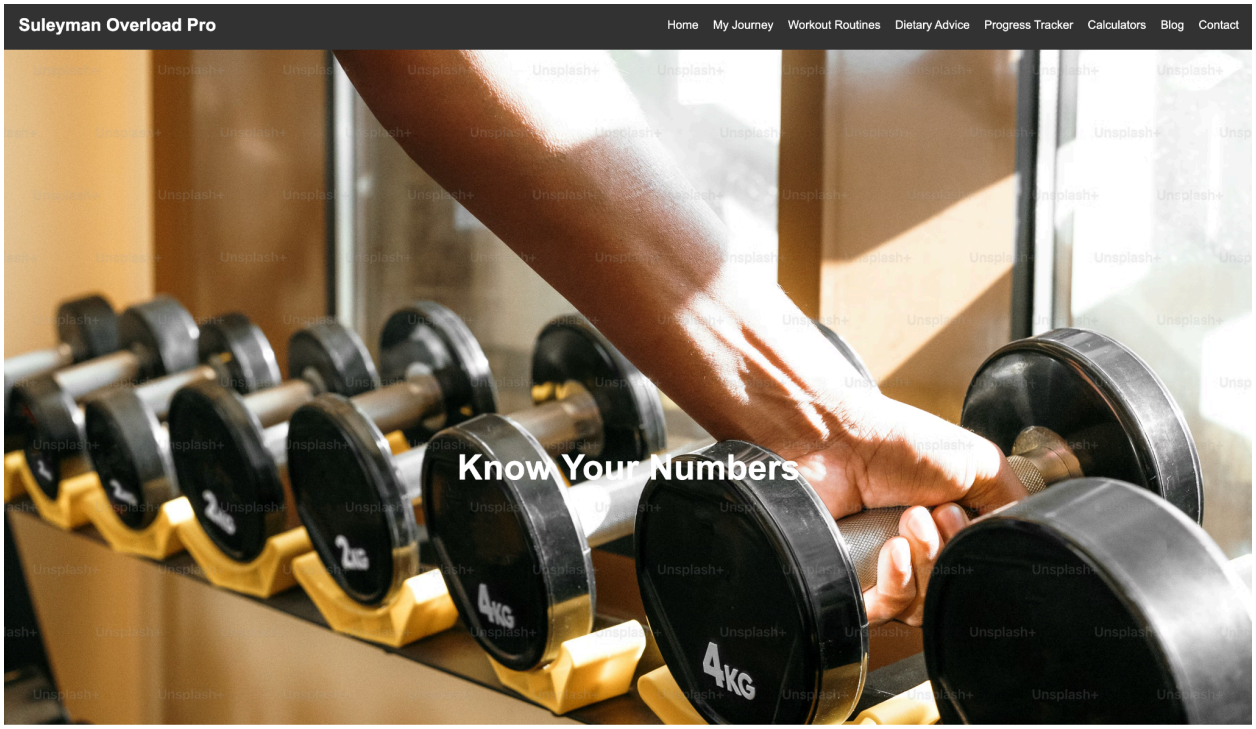


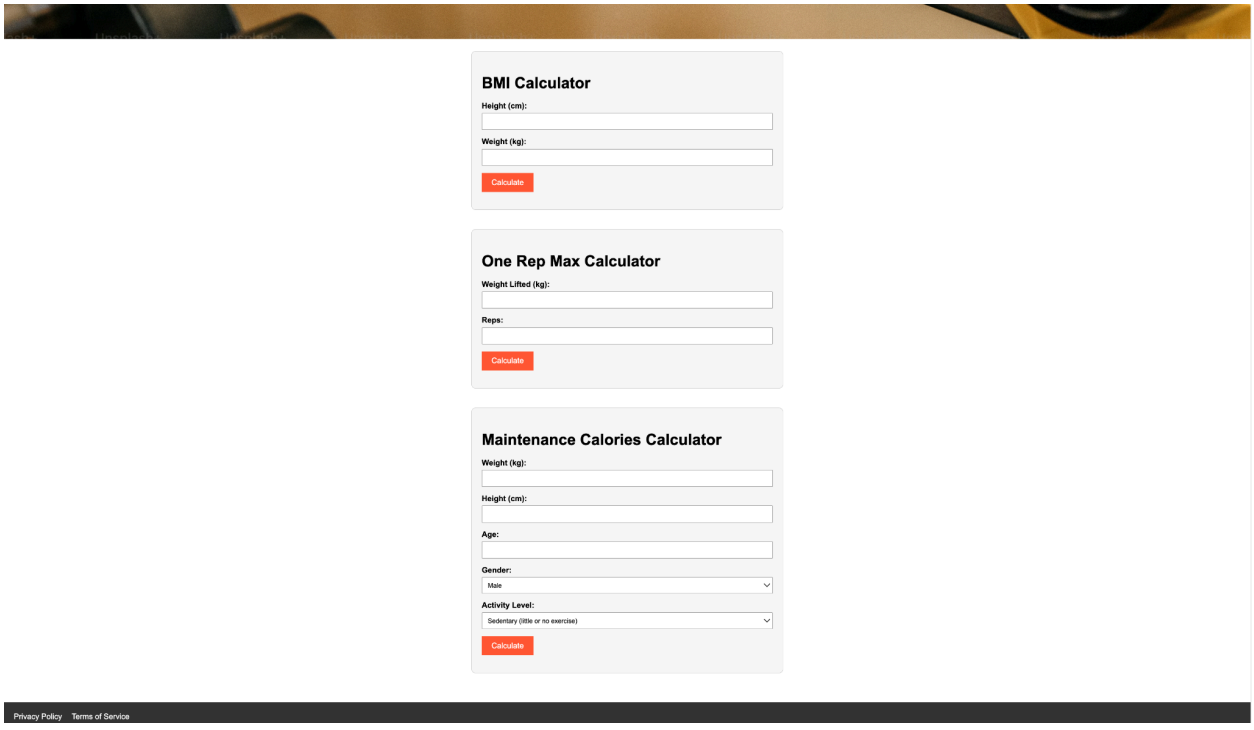
Month-by-Month Progress

Calculators Page

- **Header**: Consistent with Home Page
- **Hero Section**: Image with overlay text (e.g., "Know Your Numbers")
- **BMI Calculator**: Input fields and calculate button
- **One Rep Max Calculator**: Input fields and calculate button

- **Footer**: Consistent with Home Page





Blog Page

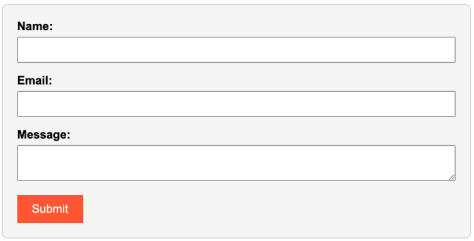
- **Header**: Consistent with Home Page
- **Hero Section**: Image with overlay text (e.g., "Stay Updated")

- **Latest Posts**: Thumbnails and snippets of latest blog posts
- **Archives**: Links to archived posts
- **Footer**: Consistent with Home Page

Contact Page

- **Header**: Consistent with Home Page
- **Hero Section**: Image with overlay text (e.g., "Get in Touch")
- **Contact Form**: Input fields for name, email, message, and submit button
- **Social Media Links**: Icons linking to social media profiles
- **Footer**: Consistent with Home Page

Contact Form



Follow Us

Sitemap

Sitemap:

1. **Home**
    - **Navigation Header**
    - Overview
    - Featured Articles
2. **Workout Routines**
    - Beginner
    - Intermediate
    - Advanced
3. **Dietary Advice**

    - Nutrition Tips
    - Recipes
4. **Progress Tracker**
    - Track Progress
    - Charts
5. **Motivational Tips**
    - Personal Stories
    - Success Stories
6. **My Journey**
    - Personal Achievements
    - Gallery
7. **Calculators**
    - BMI Calculator
    - One Rep Max Calculator
8. **Blog**
    - Latest Posts
    - Archives
9. **Contact**
    - Contact Form
    - Social Media Links

Reflective Commentary

The initial design phase of creating the fitness and health resource website has been a thorough and enlightening process. Each step, from brainstorming the idea to developing detailed personas and scenarios, has provided valuable insights and a deeper understanding of user-centered design principles.

Idea Generation and Research: The journey began with extensive brainstorming and research to determine the theme and purpose of the website. The decision to create a fitness and health resource stemmed from a personal passion for fitness and a desire to share useful information with others. Researching existing fitness websites was instrumental in identifying common features, understanding user expectations, and gathering inspiration. This research phase underscored the importance of delivering content effectively and catering to the needs of a diverse audience.

Persona and Scenario Development: Creating detailed personas was a crucial step in the design process. Developing five personas, each with unique backgrounds, fitness goals, and preferences, allowed for a comprehensive understanding of potential users. This exercise emphasized the importance of empathy in design, ensuring that the website would be inclusive and cater to a broad audience. Additionally, crafting sixteen scenarios provided practical insights into how different users would interact with the site, what features they would use, and what

content they would seek. This step was vital in anticipating user needs and designing a user-friendly interface.

Mockup and Sitemap Creation: Designing the mockup and sitemap involved multiple iterations to refine the layout and navigation. The mockup offered a visual representation of the website, helping to organize content logically and intuitively. Creating a sitemap provided a hierarchical view of the website structure, ensuring all essential pages and features were included and logically linked. This step was crucial in planning the overall user experience and ensuring seamless navigation.

Addressing Further Requirements: Considering the technical, legal, social, and organizational constraints was essential in shaping the design. Technical constraints required using JavaScript for dynamic content storage, avoiding server-side processing. This decision ensured user privacy and simplified implementation. Legal and social constraints were addressed by ensuring the website complied with privacy laws and was accessible to people with disabilities. Organizational constraints influenced the design's simplicity and maintainability, aligning with corporate styles and budget limitations. This comprehensive approach ensured the design was practical, inclusive, and sustainable.

Challenges and Solutions: One of the main challenges was balancing the inclusion of diverse features, such as workout generators and progress trackers, while maintaining a user-friendly interface. The solution lay in employing intuitive design principles and providing clear, concise instructions for each feature. Another challenge was ensuring the site's responsiveness across various devices. Using CSS media queries and testing the design on different devices helped achieve a seamless user experience regardless of screen size.

Learning Outcomes: This initial design phase has significantly deepened my understanding of user-centered design. Creating detailed personas and scenarios emphasized the importance of empathy in design, ensuring the website meets the needs of its intended audience. Additionally, the process of developing mockups and sitemaps has honed my skills in visualizing and planning website structures. Addressing various constraints has taught me to balance creativity with practical considerations, ensuring the final product is both innovative and feasible.

Future Steps: The next steps involve refining the design based on feedback from the professor and peers. This will include making any necessary adjustments to the personas, scenarios, mockup, and sitemap to ensure they align with the project goals and user needs. Additionally, further development will involve creating the actual website pages, implementing the dynamic features, and continuously testing the site to ensure usability and accessibility.

In conclusion, this initial design phase has been a comprehensive exercise in understanding and applying user-centered design principles. It has provided a solid foundation for the next stages of website development, ensuring the final product will be user-friendly, accessible, and engaging. The insights and skills gained from this phase will be invaluable in future web development projects, contributing significantly to my growth as a web developer and designer.

Reflective Learning Diary - Unit 2

Feedback for Unit 2:

Improvements this version:

HTML Structure and Compliance:
I made the structure of HTML according to HTML5. This entails the following: checking all nesting and closure of elements, applying the appropriate attributes. I structured content semantically using proper html tags.

The website is meant to be opened with the home (index.html) page.

Content Improvements:

- I added a table to the "My Journey" page, which would display tabular data effectively. This table holds data regarding monthly progress—weight and muscle gain—in an effective and clear way for the user to view.
- I have added proper alt attributes to all images in order to have better accessibility for differently abled persons.

Accessibility and Usability:
- Ensured that all forms have necessary labels, name, email, and message fields in a contact form are appropriately labelled and accessible. I did this to ensure all images had descriptive alt text.
- Improved the navigation structure for easy user movement between pages.

Effective Communication:

All text had to be read over for clarity and grammatical correctness. Headings were continuously used to break content up and make it more digestible.

Linking with Personas and Scenarios:
- The structure of the HTML and the content were directly aligned with the personas and scenarios developed in Unit 1. For instance, the progress table in the "My Journey" page will interest those persons wanting to see precise details of progress metrics. Making a contact form simple and accessible ensured that all personas, regardless of their technical proficiency, could get in touch quite easily.

Conclusion:
These choices have dramatically improved the maintainability, accessibility, and compliance of the website's HTML. This puts the site on good footing and prepares it for the future developments that will happen in the next units.

Reflective Learning Diary - Unit 3

<u>Learning Outcome Reflection</u>
Learning Outcome: Write well-structured, easily maintained, standards-compliant CSS code to present HTML pages in different ways.

Well-Structured CSS:

- **Reflection**: Throughout this unit, I have focused on organizing my CSS code in a logical and hierarchical manner. I have used comments to separate different sections, such as layout, typography, and components, which makes the code easier to navigate and maintain.
- **Example**: I have grouped related styles together and used meaningful class names that reflect their purpose.

Easily Maintained CSS:

- **Reflection**: I have adopted best practices for maintainability, such as using variables for colors and fonts, and employing a modular approach with reusable classes. This ensures that changes can be made efficiently without affecting unrelated parts of the code.

Standards-Compliant CSS:

- **Reflection**: I have adhered to W3C standards and guidelines to ensure my CSS is compliant. This includes using valid syntax, avoiding deprecated properties, and ensuring cross-browser compatibility.
- **Example**: I have validated my CSS using tools like the W3C CSS Validator and tested my pages across different browsers to ensure consistent presentation.

Present HTML Pages in Different Ways:

- **Reflection**: I have explored various CSS techniques to present HTML content in multiple ways, such as using Flexbox and Grid for layout, and media queries for responsive design. This allows the content to adapt seamlessly to different screen sizes and devices.
- **Example**: I have implemented a responsive design that adjusts the layout and typography based on the viewport size, providing an optimal user experience on both desktop and mobile devices.

Design Choices and Rationale:

The primary purpose of Unit 3 was to incorporate CSS into the website developed for the previous unit. The concentration was on keeping consistency and responsiveness in the design while structuring the CSS code in a way that it could be maintained. This is in consideration of the outcome of the units, as it was required to have a website with proper CSS for it to support compliance with web standards.

Colour Scheme:

I went with a dark color scheme and a bold accent color, #ff5733, as it puts across the themes of fitness and motivation that are "Suleyman Overload Pro." These color choices bring consistency to the site and create good visual hierarchy by drawing attention to key elements. The choice was therefore highly influenced by personas identified in Unit 1, who appreciate a lively and empowering aesthetic.

Typography:

I chose the sans-serif font Arial for easy readability and a clean look on different machines. The sizes and line heights of the font were manipulated to ensure better readability while keeping in mind that the site shall be accessible to all users, including those with visual impairments. This is part of the learning goals: to create maintainable, accessible, user-friendly CSS.

Layout:
I used CSS Flexbox to create the nav bars, footers, and main content areas. Flexbox was used because it allows for flexibility and responsiveness in design that easily adapts to different screen sizes. This means the site will be not only pleasing to the eye but also functional on a wide variety of devices, something that in the scenarios described in Unit 1 was very important.

Responsive Design:
Media queries adjusted the layout and font sizes on smaller screens for mobile devices. This ensured that the site was also usable on mobile devices, therefore adhering to the learning goal of creating a responsive design that caters to different personas and scenarios.

Reflection:
Applying CSS to this website gave me a chance to drastically improve it in terms of visual aspect and even the users' experience. These design decisions were based on fulfilling the requirements of the personas that I had come up with while keeping usability, readability, and accessibility in mind. Following best practices in CSS meant that my code was organized and maintainable for future updating and modification in the course of the project. It served to enlighten me on just how proper CSS design can enhance the user experience so much while meeting technical standards and requirements.

Learning Outcomes

When you have completed this unit, you should be able to

- **critique JavaScript code written by others, identifying examples of both good and bad practice.**
- **use JavaScript to add dynamic content to pages.**
- **modify existing JavaScript code to extend and alter its functionality and, where appropriate, to correct errors and cases of poor practice.**

Week 4 Learning Diary

Overview: This week, I focused on enhancing the functionality and content of my website to meet the requirements outlined in Unit 4. The updates were guided by the personas and scenarios developed in Unit 1, ensuring the site aligns with the needs and expectations of its intended users. The primary goals were to integrate JavaScript functionality, improve content engagement, and ensure that the site is both visually appealing and user-friendly.

1. Dynamic JavaScript Features

- **Home Page:**
  - **Change:** Added a dynamic welcome message that changes based on the time of day using JavaScript.
  - **Learning Goal:** This feature was implemented to provide a personalized experience for users, enhancing the interactivity of the site. It reflects an understanding of how JavaScript can be used to manipulate the DOM based on real-time data.
  - **Persona Connection:** This feature particularly benefits the persona of "James," who values personalized experiences and engaging content.
- **Workout Page:**
  - **Change:** Integrated interactive workout routines with dynamic content display using JavaScript. Users can now click to reveal details about beginner, intermediate, and advanced workouts.
  - **Learning Goal:** This task deepened my understanding of JavaScript event handling and content manipulation. It also improved the site's usability by allowing users to interact with content without page reloads.
  - **Persona Connection:** "Sarah," the fitness enthusiast, benefits from this feature as it helps her easily access workout routines that match her fitness level.

- **Progress Page:**
    - **Change:** Implemented a form with JavaScript functionality that displays user inputs as a progress summary. This simulates tracking their fitness journey and encourages ongoing engagement with the site.
    - **Learning Goal:** This exercise enhanced my ability to combine HTML forms with JavaScript to create dynamic, user-responsive content.
    - **Persona Connection:** The progress tracking feature is particularly valuable for "John," who is motivated by seeing his progress visually represented, helping him stay committed to his fitness goals.

2. Content Expansion

- **Dietary Advice Page:**
    - **Change:** Added detailed nutrition tips and featured recipes to enrich the content and provide valuable information to users.
    - **Learning Goal:** This addition was aimed at improving content relevance and user engagement by offering practical advice and resources. It also tested my ability to structure content in a user-friendly and visually appealing manner.
    - **Persona Connection:** The expanded content addresses the needs of "Emily," who is focused on healthy eating and looking for easy-to-follow dietary advice that complements her fitness routine.
- **Blog Page:**
    - **Change:** Expanded the blog section with additional posts and a "Read More" feature for better content interaction.
    - **Learning Goal:** This task helped me understand how to manage content that is initially hidden and revealed on user interaction, improving the overall user experience.
    - **Persona Connection:** The blog content is designed to keep "James" informed and motivated, aligning with his desire for ongoing education and inspiration in his fitness journey.

3. Visual and Functional Enhancements

- **Image Styling:**
    - **Change:** Updated image styling across the site, particularly on the My Journey page, to ensure a consistent and polished visual presentation.
    - **Learning Goal:** This update was an exercise in maintaining visual consistency and professionalism across the site, which is crucial for user trust and engagement.
    - **Persona Connection:** "Sarah" appreciates a visually appealing site that reflects the quality and care put into the content she consumes.

4. Social Media Integration

- **Change:** Integrated LinkedIn and Instagram links across all pages to enhance the site's connectivity and provide users with a way to engage further on social media.
- **Learning Goal:** This task helped reinforce the importance of cross-platform presence and how it can drive further user engagement.
- **Persona Connection:** All personas benefit from easy access to additional resources and community engagement through social media.

Conclusion: Throughout this week, I successfully integrated dynamic JavaScript features, expanded content, and maintained a consistent visual design across my website. These updates not only align with the technical learning goals of the unit but also reflect the needs and behaviors of the personas developed in Unit 1. The site is now more interactive, informative, and user-friendly, providing a better overall experience for its intended audience.

Moving forward, I plan to continue refining the user experience by analyzing feedback and making iterative improvements. This week has solidified my understanding of how to use JavaScript effectively in a web project, and I'm confident that these skills will continue to enhance my work in the upcoming units.

Unit 4 Submission: Critique and Code Improvement

For this project, I found a BMI calculator script that aligns well with my project's goal of providing fitness and health tools. The code allows users to calculate their BMI based on either metric or imperial units. The original code is from [Code Boxx](https://code-boxx.com/simple-bmi-calculator-in-javascript/).

1. Example 1 code

```
<html>
<form id="bmi-form" onsubmit="return calcBMI();">
  <div class="bmi-row">
    <label>
      <input type="radio" id="bmi-metric" name="bmi-measure" onchange="measureBMI()"
checked> Metric
    </label>
    <label>
      <input type="radio" id="bmi-imperial" name="bmi-measure" onchange="measureBMI()">
Imperial
    </label>
  </div>
  <div class="bmi-row">
```

```
    <input id="bmi-weight" type="number" min="1" max="635" required>
    <input id="bmi-height" type="number" min="54" max="272" required>
  </div>
  <input type="submit" value="Calculate BMI">
  <span id="bmi-results"></span>
</form>

<script>
function measureBMI () {
  let unit = document.getElementById("bmi-metric").checked,
      weight = document.getElementById("bmi-weight"),
      height = document.getElementById("bmi-height");

  if (unit) {
    weight.min = 1; weight.max = 635;
    height.min = 54; height.max = 272;
  } else {
    weight.min = 2; weight.max = 1400;
    height.min = 21; height.max = 107;
  }
}

function calcBMI () {
  let unit = document.getElementById("bmi-metric").checked,
      weight = parseInt(document.getElementById("bmi-weight").value),
      height = parseInt(document.getElementById("bmi-height").value),
      results = document.getElementById("bmi-results"),
      bmi;

  if (unit) {
    height = height / 100;
    bmi = weight / (height * height);
  } else {
    bmi = 703 * (weight / (height * height));
  }
  bmi = Math.round(bmi * 100) / 100;

  if (bmi < 18.5) {
    results.innerHTML = bmi + " - Underweight";
  } else if (bmi < 25) {
    results.innerHTML = bmi + " - Normal weight";
  } else {
    results.innerHTML = bmi + " - Overweight";
  }
```

```
  return false;
}
</script>
</html>
```

2. Critique of the Online Code

Good Practices:

- 1. **Unit Conversion Flexibility:** The code allows switching between metric and imperial units, making it user-friendly for a global audience. This is a good practice in creating adaptable interfaces for diverse users.
- 2. **Modularization:** The code effectively splits the BMI calculation (`calcBMI`) and the unit conversion (`measureBMI`) into two functions. This modular structure enhances maintainability and readability by separating concerns.
- 3. **Form Input Validation via HTML:** The form uses HTML5's `min`, `max`, and `required` attributes to validate inputs directly within the HTML. This approach ensures basic validation without needing additional JavaScript.
- 4. **BMI Categories:** The code provides feedback based on BMI ranges (e.g., "underweight," "normal weight"). This adds value by helping users understand the significance of the result, not just the numerical value.

Bad Practices:

1. **Lack of In-depth Input Validation:** While the HTML form handles basic validation, the JavaScript should handle more comprehensive error checking. For example, users can input zero, negative numbers, or nonsensical values (e.g., height or weight out of realistic bounds).

    - Suggested Improvement: Add detailed validation in JavaScript to ensure input values are sensible.
    - For example:

```
if (weight <= 0 || height <= 0) {
  results.innerHTML = "Please enter valid positive numbers for height and weight.";
  return false;
}
```

2. **No Feedback for Invalid Input:** If invalid values are entered, the function doesn't provide detailed feedback beyond an empty result or the page remaining unchanged.

- Suggested Improvement: Improve user feedback by providing detailed error messages for invalid inputs, alerting users when the values are outside the valid range.

3. **Absence of Comments:** The code lacks comments that explain its functions or logic. This makes it difficult for other developers to understand what the code does or why certain decisions were made.

- **Suggested Improvement:** Adding comments that explain the code's purpose and logic would make it more maintainable and easier to understand for future developers.

4. **Lack of Accessibility Considerations:** The script does not provide accessibility features such as ARIA attributes, which are essential for users relying on assistive technologies.

    - Suggested Improvement: Add accessibility features like `aria-live` for dynamic content updates.

```
<span id="bmi-results" aria-live="polite"></span>
```

5. **Imperial to Metric Conversion Limited to BMI:** The code handles BMI calculations for both unit systems but does not offer conversions beyond BMI (e.g., one-rep max, maintenance calories). This is a limitation if the application needs to handle more fitness-related metrics.

3. Improved Version of the Code (to be used in my project)

My Updated Version integrates more robust input validation, improved user feedback, and better accessibility. Additionally, it follows best practices by including comments and handling both BMI and other fitness-related calculations, such as maintenance calories and one-rep max.

```
<form id="bmi-form" onsubmit="return calcBMI();">
  <div class="bmi-row">
    <label>
      <input type="radio" id="bmi-metric" name="bmi-measure" onchange="measureBMI()"
checked> Metric
    </label>
    <label>
      <input type="radio" id="bmi-imperial" name="bmi-measure" onchange="measureBMI()">
Imperial
    </label>
  </div>
```

```
  <div class="bmi-row">
    <input id="bmi-weight" type="number" min="1" max="635" required placeholder="Weight">
    <input id="bmi-height" type="number" min="54" max="272" required placeholder="Height">
  </div>
  <input type="submit" value="Calculate BMI">
  <span id="bmi-results" aria-live="polite"></span>
</form>

<script>
// Function to switch between metric and imperial units
function measureBMI() {
  let unit = document.getElementById("bmi-metric").checked,
      weight = document.getElementById("bmi-weight"),
      height = document.getElementById("bmi-height");

  if (unit) {
    weight.min = 1; weight.max = 635;
    height.min = 54; height.max = 272;
  } else {
    weight.min = 2; weight.max = 1400;
    height.min = 21; height.max = 107;
  }
}

// Function to calculate BMI and display result
function calcBMI() {
  let unit = document.getElementById("bmi-metric").checked,
      weight = parseInt(document.getElementById("bmi-weight").value),
      height = parseInt(document.getElementById("bmi-height").value),
      results = document.getElementById("bmi-results"),
      bmi;

  // Validate input
  if (weight <= 0 || height <= 0) {
    results.innerHTML = "Please enter valid positive numbers for height and weight.";
    return false;
  }

  // Calculate BMI
  if (unit) {
    height = height / 100;  // Convert height to meters
    bmi = weight / (height * height);
  } else {
    bmi = 703 * (weight / (height * height)); // Imperial calculation
```

```
  }
  bmi = Math.round(bmi * 100) / 100;

  // Provide BMI category feedback
  if (bmi < 18.5) {
    results.innerHTML = `Your BMI is ${bmi} - Underweight`;
  } else if (bmi < 25) {
    results.innerHTML = `Your BMI is ${bmi} - Normal weight`;
  } else if (bmi < 30) {
    results.innerHTML = `Your BMI is ${bmi} - Overweight`;
  } else {
    results.innerHTML = `Your BMI is ${bmi} - Obesity`;
  }

  return false;
}
</script>
```

Conclusion:
This updated version integrates better validation, user feedback, and accessibility features while maintaining a clean and modular structure. It follows best practices learned from the critique and offers a better user experience. The improvements ensure compliance with the requirements outlined in Unit 4 and aim to demonstrate a thorough understanding, modification, and integration of JavaScript code.

2. Example 2 code

Personalized for users to meet personas

The online code from James1x0 on GitHub Gist provides a JavaScript function that returns a greeting based on the current time using the moment.js library.

Online code:

```
function getGreetingTime(m) {
        var g = null; //return g

        if(!m || !m.isValid()) { return; } //if we can't find a valid or filled moment, we return.

        var split_afternoon = 12 //24hr time to split the afternoon
        var split_evening = 17 //24hr time to split the evening
        var currentHour = parseFloat(m.format("HH"));
```

```
        if(currentHour >= split_afternoon && currentHour <= split_evening) {
                g = "afternoon";
        } else if(currentHour >= split_evening) {
                g = "evening";
        } else {
                g = "morning";
        }

        return g;
}
```

Good Practices:

1. **Use of** moment.js **for Time Handling**:
    - The code uses the moment.js library to handle time and format it appropriately. This is a good practice because moment.js simplifies date and time manipulation, ensuring cross-browser consistency and making the code easier to maintain.
2. **Logical Splitting of Time Periods**:
    - The function defines clear boundaries for “morning,” “afternoon,” and “evening,” which makes it easier to understand the intended behavior of the code.
3. **Input Validation**:
    - The function checks if the moment object m is valid before proceeding. This ensures that invalid or undefined inputs do not break the function

Bad Practices:

1. **Over-reliance on External Libraries**:
    - The code requires the moment.js library to function, which is an additional dependency. While moment.js is powerful, modern JavaScript now provides robust native date-handling methods (such as Date and Intl.DateTimeFormat), reducing the need for external libraries.
    - **Suggested Improvement**: Use native JavaScript for date and time manipulation to avoid the overhead of importing libraries.
    - let currentHour = new Date().getHours();

2. **Hardcoded Time Splits**:
    - The time splits (12 for afternoon and 17 for evening) are hardcoded, which limits flexibility. If a different application requires different splits (e.g., defining evening as starting at 18:00), the function would need to be manually modified.
    - **Suggested Improvement**: Use parameters to pass time splits for more flexibility, allowing the function to be reused in different contexts without modification.
    - function getGreetingTime(currentHour, afternoonStart = 12, eveningStart = 17) {
      // logic remains the same
3. **Limited Error Handling**:
    - While the function checks for a valid moment.js object, it does not provide any feedback in cases where the input is invalid, potentially leaving users unsure about what went wrong.

- **Suggested Improvement**: Provide informative error messages when the input is invalid, or consider returning a default value like “Invalid Time” instead of returning nothing.

4. **Overuse of Variables**:
   - The variable g is initialized as null, but it is reassigned immediately based on conditions. This extra initialization can be skipped by directly returning the result of the conditions, reducing the number of lines and improving readability.
   - **Suggested Improvement**: Simplify the function by directly returning the appropriate greeting based on conditions.
   - function getGreetingTime(m) {
   - if (!m || !m.isValid()) return "Invalid Time";
   - let currentHour = parseFloat(m.format("HH"));
   - return currentHour >= 17 ? "evening" : currentHour >= 12 ? "afternoon" : "morning";
   - }

My code based on critique:

```
<script>
   window.onload = function() {
      var greeting;
      var hour = new Date().getHours();

      if (hour < 12) {
         greeting = "Good Morning";
      } else if (hour < 18) {
         greeting = "Good Afternoon";
      } else {
         greeting = "Good Evening";
      }

      document.getElementById("welcome-message").innerText = greeting + ", My name is Suleyman Kiani, Welcome to my fitness Journey!!";
   };
</script>
```

Good Practices:

1. **Simplicity:**

    - Your code uses native JavaScript, which is efficient and does not require external libraries like moment.js. This is a good approach, as modern JavaScript has sufficient built-in functionality for time-related tasks.
2. **Clear and Readable Logic**:
    - The code is easy to follow, with clear conditions for morning, afternoon, and evening. It handles a basic use case without overcomplicating the logic.
3. **Integration with the DOM**:
    - The code effectively interacts with the DOM by updating the text of an HTML element (welcome-message) based on the greeting. This demonstrates good integration of JavaScript and HTML.

Bad Practices:

1. **Hardcoded Time Splits**:
    - Similar to the original code, your time splits (12:00 and 18:00) are hardcoded, which limits flexibility. If you want to reuse this code for a different time scheme, you would need to modify the values manually.
    - **Suggested Improvement**: Make the time splits configurable by using parameters or variables that can be adjusted dynamically.
    - var morningEnd = 12;
    - var afternoonEnd = 18;
2. **No Fallback for Invalid Time**:
    - While it is highly unlikely that new Date().getHours() would return an invalid time, edge cases or unexpected behavior could still occur, and the function does not provide any fallback in such cases.
    - **Suggested Improvement**:
    - Include a fallback or error-handling mechanism in case the time retrieval fails (even though it's rare).
    - var hour = new Date().getHours();
    - if (isNaN(hour)) {
    - greeting = "Hello";
    - }
3. **Global Variables**:
    - The variable greeting is declared without the let or const keyword, making it a global variable. Global variables should generally be avoided as they can cause unexpected behavior in larger scripts.
    - **Suggested Improvement**: Use let or const to limit the scope of variables to avoid polluting the global namespace.
    - let greeting;

Suggested Improved Version

Here's a refined version of the greeting script based on the critique. It includes configurable time splits, better variable handling, and improved error checking.

```
<script>
   window.onload = function() {
      const morningEnd = 12;
      const afternoonEnd = 18;
      let greeting;

      let hour = new Date().getHours();

      // Error handling if the hour is not valid
      if (isNaN(hour)) {
         greeting = "Hello";
      } else if (hour < morningEnd) {
         greeting = "Good Morning";
      } else if (hour < afternoonEnd) {
         greeting = "Good Afternoon";
      } else {
         greeting = "Good Evening";
      }

      document.getElementById("welcome-message").innerText =
         `${greeting}, My name is Suleyman Kiani, Welcome to my fitness Journey!!`;
   };
</script>
```

Key Improvements:

- **Configurable Time Splits**: You can now easily adjust the time splits for morning and afternoon.
- **Better Error Handling**: If new Date().getHours() fails, the code defaults to a generic greeting (“Hello”).
- **Scope Limiting**: I replaced the global greeting variable with let, preventing unnecessary pollution of the global namespace.
- **Improved Readability**: The code uses template literals for cleaner string concatenation.

This version improves flexibility, robustness, and readability while maintaining the core functionality of greeting users based on the current time.

Critique of the Online Code for Form Validation

Here’s an online form validation example found from W3Docs that handles basic input fields like name, email, password, and telephone.

Online Code Example:

```
function ValidationForm() {
  let username = document.forms["RegForm"]["Name"];
  let email = document.forms["RegForm"]["Email"];
  let phoneNumber = document.forms["RegForm"]["Telephone"];
  let select = document.forms["RegForm"]["Subject"];
  let pass = document.forms["RegForm"]["Password"];

  if (username.value == "") {
    alert("Please enter your name.");
    username.focus();
    return false;
  }
  if (email.value == "" || email.value.indexOf("@") < 0 || email.value.indexOf(".") < 0) {
    alert("Please enter a valid e-mail address.");
    email.focus();
    return false;
  }
  if (phoneNumber.value == "") {
    alert("Please enter your telephone number.");
    phoneNumber.focus();
    return false;
  }
  if (pass.value == "") {
    alert("Please enter your password.");
    pass.focus();
    return false;
  }
  if (select.selectedIndex < 1) {
    alert("Please enter your course.");
    select.focus();
    return false;
  }

  return true;
}
```

Good Practices:

1. **Basic Field Validation**: The function checks if the form fields are filled out before submission, ensuring that the form won't submit incomplete data. This is essential for user experience and maintaining data integrity.
1. **Field-Specific Validations**: For the email field, it checks for the presence of "@" and "." characters, preventing common email input errors.
2. **Focus for Error Fields**: When a field is invalid, the function sets focus to the invalid field (focus()), which is helpful in guiding the user to the exact input that needs correction.

Bad Practices:

1. **Weak Email Validation**: The validation for the email field only checks for the presence of "@" and ".", which is not sufficient for ensuring a valid email format. Modern email validation should include a regular expression (regex) to handle more complex email formats.

    - **Suggested Improvement**: Use a regex pattern for email validation, such as:
    -
    - let emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    - if (!emailPattern.test(email.value)) {
    - alert("Please enter a valid email address.");
    - email.focus();
    - return false;
    - }

2. **No Detailed Error Messages:** The function uses alert() for error feedback, which is disruptive and does not provide specific error messages. Using inline error messages next to the input fields would create a better user experience.
    - **Suggested Improvement**: Implement inline feedback messages by using innerHTML to display validation messages next to the form fields. This can be done by creating a function to display the error message next to the field.
3. **No Handling for Other Common Input Types**: The code doesn't validate input types such as numbers or characters in names, leaving potential for incorrect data input.
    - **Suggested Improvement**: Add checks using regular expressions for name and phone fields to ensure valid input:
    -
    - let namePattern = /^[a-zA-Z\s]+$/;
    - if (!namePattern.test(username.value)) {
    - alert("Please enter a valid name.");
    - username.focus();
    - return false;
    - }

Improved version of the code for my site:

```
<script>
  function validateForm() {
    var name = document.forms["contactForm"]["name"].value;
    var email = document.forms["contactForm"]["email"].value;
    var message = document.forms["contactForm"]["message"].value;

    // Error messages container
    var errorMessage = document.getElementById("error-message");
```

```
        // Validate empty fields
        if (name == "" || email == "" || message == "") {
            errorMessage.innerText = "All fields must be filled out.";
            return false;
        }

        // Validate name field (only letters and spaces)
        var namePattern = /^[a-zA-Z\s]+$/;
        if (!namePattern.test(name)) {
            errorMessage.innerText = "Please enter a valid name.";
            return false;
        }

        // Validate email format
        var emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
        if (!emailPattern.test(email)) {
            errorMessage.innerText = "Please enter a valid email address.";
            return false;
        }

        // Clear error message if validation passes
        errorMessage.innerText = "";
        return true;
    }
</script>
```

This version includes field-specific validation, inline error feedback, and regex-based validation for the email and name fields. It also clears the error message once the form passes validation, enhancing both functionality and user experience.

Review of JavaScript Program Designs for Suleyman Overload Pro

For the JavaScript integration into my fitness website, *Suleyman Overload Pro*, I have designed and implemented several features aimed at enhancing the user experience and addressing the needs of the personas developed in Unit 1. Below is an overview of each program design, including its purpose, functionality, and relation to the scenarios.

---

1. Toggle “Read More” Feature (Blog Page)

- **Purpose**: The primary goal of this feature is to give users control over the content they want to view. By adding a toggle function to each blog post, users can either expand the post to read more or collapse it to save space.
- **Design**:
    - JavaScript is used to control the visibility of additional text. The toggle button dynamically changes its label between "Read more" and "Read less" based on the content's state.
    - **Objects/Functions**: `toggleReadMore(id)` function; DOM elements such as buttons and paragraphs are targeted.
    - **Variables**: `dots`, `moreText`, `btnText` (all capturing specific sections of the content to be controlled).
- **Program Flow**: When a user clicks the “Read more” button, the JavaScript function hides the dots, reveals the extended content, and changes the button label to "Read less." When clicked again, the content is collapsed.
- **Relation to Personas**: This feature is essential for *Jane*, who likes to browse fitness articles without feeling overwhelmed by long text blocks. It improves usability by giving her control over how much she wants to read.
- **Code:**

```
      /**
       * Toggles the visibility of the full blog post content and updates the read
more/less button.
       *
       * @param {string} id - The unique identifier for the blog post.
       */
      function toggleReadMore(id) {
          // Get the DOM elements for the specific blog post
```

```
        var dots = document.getElementById("dots-" + id);

        var moreText = document.getElementById("more-" + id);

        var btnText = document.getElementById("read-more-btn-" + id);



        // Check if the full content is currently hidden

        if (dots.style.display === "none") {

            // If full content is visible, hide it and show the ellipsis

            dots.style.display = "inline";

            btnText.innerText = "Read more";

            moreText.style.display = "none";

        } else {

            // If full content is hidden, show it and hide the ellipsis

            dots.style.display = "none";

            btnText.innerText = "Read less";

            moreText.style.display = "inline";

        }

    }
```

---

2. Fitness Calculators (BMI, One-Rep Max, and Maintenance Calories)

- **Purpose**: These calculators are designed to offer users immediate feedback based on their personal fitness data. The BMI calculator, one-rep max calculator, and calorie maintenance calculator allow users to calculate important metrics without leaving the page.
- **Design**:

    - Each calculator is embedded in its respective section and responds to user inputs via event handlers (on button click).
    - **Objects/Functions**:
        - `calculateBMI()`: Calculates BMI using height and weight inputs.
        - `calculateOneRepMax()`: Calculates one-rep max using weight lifted and repetitions.
        - `calculateMaintenanceCalories()`: Calculates daily caloric maintenance based on weight, height, age, gender, and activity level.
    - **Variables**: Input fields such as `height`, `weight`, `age`, and dropdowns for gender and activity level. Results are displayed using `innerText`.
- **Program Flow**: Each function captures user inputs, performs the necessary calculations using formulas, and then displays the result below the form.
- **Relation to Personas**: For *John*, a beginner who wants personalized workout recommendations, and *Mark*, an advanced user looking for precise fitness data, these calculators provide essential tools to track progress and set goals.
- **BMI Calculator code:**

```
      /**
       * Calculates the Body Mass Index (BMI) based on user input.
       *
       * This function retrieves the height and weight values entered by the user,
       * validates the input, calculates the BMI using the formula: weight (kg) /
(height (m))^2,
       * and displays the result or an error message if the input is invalid.
       *
       * The BMI categories are also provided for context.
       */
      function calculateBMI() {
          // Retrieve height and weight values from input fields
          const height = parseFloat(document.getElementById('height').value);
          const weight = parseFloat(document.getElementById('weight').value);

          // Check if both height and weight are positive values and not NaN
          if (height > 0 && weight > 0 && !isNaN(height) && !isNaN(weight)) {
              // Calculate BMI
              // Note: Height is converted from cm to m by dividing by 100
              const bmi = (weight / ((height / 100) ** 2)).toFixed(1);

              // Determine BMI category
              let category;
              if (bmi < 18.5) {
                  category = "Underweight";
              } else if (bmi < 25) {
```

```
                category = "Normal weight";
            } else if (bmi < 30) {
                category = "Overweight";
            } else {
                category = "Obese";
            }

            // Display the calculated BMI and category
            document.getElementById('bmi-result').innerHTML = `Your BMI is
${bmi}<br>Category: ${category}`;
        } else {
            // Display an error message if input is invalid
            document.getElementById('bmi-result').innerText = 'Please enter valid
positive numbers for height and weight.';
        }
    }
```

- **One rep max Calculator:**

```
    /**
     * Calculates the One Rep Max (1RM) based on weight lifted and number of reps.
     *
     * @description
     * This function uses the Brzycki formula: 1RM = weight * (36 / (37 - reps))
     * It's considered accurate for rep ranges up to 10.
     *
     * @function
     * @name calculateOneRepMax
     *
     * @returns {void} - Updates the DOM with the calculation result or error
message
     */
    function calculateOneRepMax() {
        // Parse input values from the DOM
        const weight = parseFloat(document.getElementById('weight-lifted').value);
        const reps = parseInt(document.getElementById('reps').value);

        // Validate input and calculate 1RM
        if (weight > 0 && reps > 0 && reps <= 10) {
            // Apply Brzycki formula and round to 2 decimal places
            const oneRepMax = (weight * (36 / (37 - reps))).toFixed(2);
            // Display the calculated 1RM
```

```
            document.getElementById('one-rep-max-result').innerText = `Your
estimated One Rep Max is ${oneRepMax} kg`;
        } else if (weight > 0 && reps > 10) {
            // Warn user if reps exceed recommended range for accurate calculation
            document.getElementById('one-rep-max-result').innerText = 'For accurate
results, please enter 10 reps or fewer.';
        } else {
            // Display error message for invalid input
            document.getElementById('one-rep-max-result').innerText = 'Please enter
valid positive numbers for weight and reps.';
        }
    }
```

- **Calorie Maintainence Calculator:**

```
function calculateMaintenanceCalories
            // Parse input values from the DOM
            const weight =
parseFloat(document.getElementById('weight-maint').value);
            const height =
parseFloat(document.getElementById('height-maint').value);
            const age = parseInt(document.getElementById('age').value);
            const gender = document.getElementById('gender').value;
            const activityLevel =
parseFloat(document.getElementById('activity-level').value);

            // Validate input
            if (weight > 0 && height > 0 && age > 0 && !isNaN(activityLevel)) {
                // Calculate BMR using Mifflin-St Jeor Equation
                let bmr;
                if (gender === "male") {
                    // BMR formula for males
                    bmr = (10 * weight) + (6.25 * height) - (5 * age) + 5;
                } else if (gender === "female") {
                    // BMR formula for females
                    bmr = (10 * weight) + (6.25 * height) - (5 * age) - 161;
                } else {
                    // Handle invalid gender input
```

```
document.getElementById('maintenance-calories-result').innerText = 'Please
select a valid gender.';
                    return;
                }


                // Calculate maintenance calories by multiplying BMR with
activity level
                const maintenanceCalories = (bmr * activityLevel).toFixed(0);


                // Display the calculated maintenance calories
                document.getElementById('maintenance-calories-result').innerText
= `Your estimated Maintenance Calories are ${maintenanceCalories} kcal/day`;
            } else {
                // Display error message for invalid input
                document.getElementById('maintenance-calories-result').innerText
= 'Please enter valid positive numbers for all fields.';
            }
        }
```

---

3. Form Validation (Contact Page)

- **Purpose**: The goal of this design is to ensure that users submit valid data when contacting the site. It prevents incomplete or incorrect data entry, reducing potential errors and frustration for users.
- **Design**:
    - JavaScript checks for empty fields and verifies that the email is in the correct format using a regular expression pattern.
    - **Objects/Functions**: `validateForm()` function, which returns `false` if the data is invalid and prevents form submission.
    - **Variables**: Input fields for name, email, and message. Email validation is performed using the regular expression stored in the `emailPattern` variable.

- **Program Flow**: When the user submits the form, the JavaScript function checks each field for missing data and validates the email format. If any field is invalid, an alert message is displayed, and the form is not submitted.
- **Relation to Personas**: For users like *Sarah*, who may be using the contact form for inquiries or fitness advice, this feature ensures that their messages are correctly formatted and prevents errors that could lead to incomplete submissions.
- **Code:**

```
/**
 * Validates the contact form before submission.
 * Checks for empty fields and valid email format.
 * @returns {boolean} True if form is valid, false otherwise.
 *
 * Sample tests (run these in the browser console):
 *
 * // Test empty fields
 * document.forms["contactForm"]["name"].value = "";
 * document.forms["contactForm"]["email"].value = "";
 * document.forms["contactForm"]["message"].value = "";
 * console.log(validateForm()); // Should return false
 *
 * // Test invalid email
 * document.forms["contactForm"]["name"].value = "John Doe";
 * document.forms["contactForm"]["email"].value = "invalid.email";
 * document.forms["contactForm"]["message"].value = "Test message";
 * console.log(validateForm()); // Should return false
 *
 * // Test valid input
```

```
 * document.forms["contactForm"]["name"].value = "John Doe";
 * document.forms["contactForm"]["email"].value = "john@example.com";
 * document.forms["contactForm"]["message"].value = "Test message";
 * console.log(validateForm()); // Should return true
 */
function validateForm() {
    // Get form field values
    const name = document.forms["contactForm"]["name"].value.trim();
    const email = document.forms["contactForm"]["email"].value.trim();
    const message = document.forms["contactForm"]["message"].value.trim();

    // Check for empty fields
    if (!name || !email || !message) {
        alert("All fields must be filled out");
        return false; // Prevent form submission
    }

    // Validate email format using a regular expression
    const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!emailPattern.test(email)) {
        alert("Please enter a valid email address");
        return false; // Prevent form submission
    }
```

```
        // All validations passed

        return true;

    }
```

---

4. Time-Sensitive Greeting (Home Page)

- **Purpose**: This feature creates a welcoming, personalized touch by displaying a different greeting message based on the time of day. This simple JavaScript functionality enhances the user experience by making the site feel more dynamic and responsive.
- **Design**:
    - The script checks the current time when the page loads and assigns a greeting (morning, afternoon, or evening) accordingly.
    - **Objects/Functions**: The `window.onload` event triggers a function that uses the `Date()` object to retrieve the current hour.
    - **Variables**: `hour`, `greeting`, `welcome-message` (used to display the greeting text).
- **Program Flow**: Once the page loads, the JavaScript function determines the current hour and updates the content of the `welcome-message` element with an appropriate greeting based on the time of day.
- **Relation to Personas**: This feature subtly improves the experience for all users, especially returning visitors like *John*, by creating a friendly, time-sensitive greeting that adds a human touch to the site.

- **Code:**

```
<script>

    /**

     * This script sets a personalized greeting based on the time of day.

     * It runs when the window loads, determining the appropriate greeting

     * (morning, afternoon, or evening) and updates the welcome message.

     *

     * The script performs the following steps:
```

```
 * 1. Waits for the window to fully load.
 * 2. Gets the current hour using the Date object.
 * 3. Determines the appropriate greeting based on the hour.
 * 4. Finds the welcome message element in the DOM.
 * 5. Updates the welcome message with the greeting and introduction.
 *
 * Error handling is included to log an error if the welcome message
 * element is not found in the DOM.
 */
window.onload = function() {
    // Initialize greeting variable
    let greeting;
    // Get current hour (0-23)
    const hour = new Date().getHours();


    // Set greeting based on time of day
    if (hour < 12) {
        greeting = "Good Morning";
    } else if (hour < 18) {
        greeting = "Good Afternoon";
    } else {
        greeting = "Good Evening";
    }
```

```
            // Find the welcome message element
            const welcomeMessage = document.getElementById("welcome-message");



            // Update the welcome message with the appropriate greeting
            if (welcomeMessage) {
                welcomeMessage.textContent = `${greeting}, My name is Suleyman Kiani.
Welcome to my fitness journey!`;
            } else {
                // Log an error if the element is not found
                console.error("Element with id 'welcome-message' not found");
            }
        };
    </script>
```

Design Considerations

- **Flow Diagrams/Pseudo-Code**: Each of the program designs follows a clear flow:
    - User input is captured.
    - JavaScript performs the required logic (calculations, validation, toggling).
    - The result is displayed on the page or an action is triggered (e.g., preventing form submission).
- **Integration with HTML/CSS**: All JavaScript is well-integrated with the HTML structure and adheres to the design language of the website. The calculators, validation forms, and interactive buttons are consistent with the CSS styling, maintaining a seamless user experience.
- **Scope and Complexity**: Each design is crafted to balance functionality with ease of use. While the calculators involve basic mathematical logic, the form validation uses regular expressions, offering a higher level of technical complexity.

Conclusion

The JavaScript program designs were carefully crafted to fit the needs of my personas and improve user experience across different sections of the website. From interactive calculators to form validation and dynamic content, each feature aligns with the goals of the site and provides valuable functionality for users at various stages of their fitness journey. These designs enhance usability, accessibility, and interactivity, ensuring that the website remains engaging and user-friendly for a wide audience.

Reflective Learning Diary Unit 5:

Reflective Learning Diary: JavaScript Integration in Suleyman Overload Pro

For Unit 5, my focus was to enhance the functionality of my fitness website using JavaScript. My personas and scenarios, developed in Unit 1, were pivotal in guiding the features I implemented. Here's a breakdown of the JavaScript features I added and how they improve the site for my target users:

1. Toggle "Read More" Feature for Blog Posts

- **Idea**: I created a "Read more" button that expands and collapses content on the blog page. This keeps the blog posts visually clean while allowing users to explore more if they are interested.
- **Purpose**: This feature directly addresses *Jane*, who enjoys exploring different workout routines and reading articles. The ability to toggle content allows her to consume information at her own pace, without being overwhelmed by long posts.
- **Reflection**: This was a simple yet impactful feature. It keeps the interface clean and improves user engagement by giving them control over the content they wish to view. The feature aligns perfectly with the information-rich theme of my site, allowing users to access more details only when they are interested.

2. Interactive Calculators for Fitness Metrics

- **Idea**: I implemented interactive calculators, including a BMI calculator, a one-rep max calculator, and a calorie maintenance calculator. Users can input their data and receive immediate feedback.
- **Purpose**: The calculators were designed with *John* and *Mark* in mind. *John* uses the one-rep max calculator to get personalized lifting recommendations, while *Mark* uses the calorie maintenance calculator to refine his advanced nutritional plans. These tools help both beginners and advanced users tailor their fitness plans effectively.
- **Reflection**: The calculators enhance the site's interactive nature, making it more engaging and useful. This is especially critical for users like *John* and *Mark* who rely on these metrics to track their progress and adjust their fitness routines. This feature has been one of the key tools that bridges information with actionable insights.

3. Form Validation Using JavaScript

- **Idea**: I added form validation to the contact page, ensuring that all fields are filled out and that the email entered follows a correct format using regular expressions.
- **Purpose**: This feature caters to users like *Sarah*, who may be reaching out for fitness advice via the contact form. The validation reduces user frustration by preventing errors before form submission and ensuring the information provided is accurate.

- **Reflection**: Adding real-time validation using JavaScript improved the user experience by providing immediate feedback. This not only minimizes errors but also makes the form more accessible and user-friendly. It ensures smooth interaction, reinforcing the professional feel of the site.

4. Time-Sensitive Greeting

- **Idea**: I created a time-sensitive greeting message on the homepage that changes based on the time of day (morning, afternoon, evening). While I initially considered a personalized greeting for logged-in users, I focused on keeping the implementation simple and relevant to my site's scope.
- **Purpose**: This feature adds a personal touch for users like *John*, who visits the site regularly. A friendly greeting makes the site feel welcoming and adds a human touch to the otherwise structured information.
- **Reflection**: Although simple, this greeting feature makes the homepage more dynamic and engaging. It helps create a positive first impression, which is crucial for users navigating the site for fitness advice and motivation.

Overview of the Changes

The proposed changes for the fitness website focus on improving user interaction through the integration of JQuery features. These updates aim to provide a seamless, user-friendly experience across different parts of the site, enhancing the accessibility and usability of the tools and content. The goal is to make the site more engaging for users, particularly the personas developed in Unit 1: John Doe (beginner), Jane Smith (intermediate), and Mark Johnson (advanced).

Proposed JQuery Features:

1. **Collapsible Sections for Blog Posts and Nutritional Advice**:
    - **Implementation**: I will use JQuery to create collapsible sections in both the blog and dietary advice pages. This allows users to read through sections of interest without overwhelming them with too much content at once.
    - **Benefit for Personas**:
        - **John Doe**: As a beginner, John can focus on the basics first and expand content only when he feels comfortable. This prevents information overload.
        - **Mark Johnson**: For advanced users like Mark, this feature allows quick navigation to the details he needs without unnecessary scrolling.
2. **Enhanced Form Validation on Contact Page**:
    - **Implementation**: The contact form will use JQuery for real-time validation, ensuring that users are notified of any issues before submitting the form.
    - **Benefit for Personas**:
        - **Jane Smith**: As a busy freelance designer, Jane can quickly input her information without having to reload the page, saving her time and improving her experience.
3. **Smooth Animations for Progress Tracker**:
    - **Implementation**: Adding JQuery animations for workout progress entries ensures that new entries slide into view, providing a visually engaging experience.
    - **Benefit for Personas**:
        - **John Doe**: The visual feedback when adding new progress entries can help keep John motivated as he sees his progress reflected smoothly.
        - **Mark Johnson**: As someone who tracks advanced data, the smooth animation will make Mark's interaction with the site more polished and professional.
4. **Interactive Calculator Enhancements**:
    - **Implementation**: The fitness calculators (BMI and one-rep max) will be improved with real-time error handling and dynamic updates using JQuery.
    - **Benefit for Personas**:
        - **John and Mark**: Both beginner and advanced users can enjoy a more interactive and engaging experience when calculating their fitness metrics, with instant feedback and error checks preventing confusion.

Justification for Using JQuery:

- **Efficiency**: JQuery simplifies the development process, allowing these features to be implemented with less code than pure JavaScript.
- **User Experience**: These enhancements are in line with the goals of improving usability and engagement for all user types.
- **Mobile and Accessibility**: By integrating JQuery, I can ensure that animations and interactions remain smooth on both desktop and mobile platforms, enhancing the experience for users of all abilities.

Learning Diary: Reflections on the Process Unit 6

Introduction:

Working through this unit has been a rewarding experience, as I was able to significantly enhance the usability of my fitness website using JQuery. The ability to integrate pre-built libraries allowed me to create a more dynamic user experience without writing complex code from scratch. This process involved learning to balance the functionality needs of my personas (John, Jane, and Mark) with the technical possibilities of JQuery.

Challenges Faced:

One of the biggest challenges I encountered was ensuring that the new dynamic features did not interfere with the accessibility of the site. I had to ensure that the collapsible sections and form validation worked smoothly, even if users had JavaScript disabled. This led to a deeper understanding of progressive enhancement and how important it is to build a website that works well for all users, regardless of the technology they have access to.

Another obstacle was integrating JQuery animations into the progress tracker. Initially, I overcomplicated the implementation, resulting in slow and clunky animations. By simplifying the logic and relying more on JQuery's built-in functions, I was able to create smooth transitions that improved the user experience without slowing down the page.

What I Would Do Differently:

If I were to do this again, I would spend more time planning the implementation of JQuery features to ensure better integration with the site's existing functionality. Initially, I jumped into the coding process too quickly and encountered several integration issues. By taking more time to consider the technical requirements, I could have streamlined the process.

Additionally, I would explore more JQuery plugins, as they offer a wide range of features that could be beneficial. For instance, using a plugin for form validation could have saved time compared to writing custom code.

Strengths and Weaknesses of JQuery:

JQuery has proven to be an excellent tool for simplifying JavaScript code, especially when it comes to handling DOM manipulation and animations. Its ability to chain methods and simplify complex tasks made it easier to add dynamic elements to my site. For example, adding collapsible sections to the blog posts and nutritional advice was incredibly simple using JQuery

Reflective Learning Diary for Unit 7 – EmailJS Integration and External APIs

In **Unit 7**, the goal was to integrate external services into my website to enhance user interaction by using APIs or web services. This unit challenged me to leverage JavaScript for accessing dynamic content via external APIs and improve the overall functionality of the site. The main external service I used was **EmailJS**, which I integrated into the **Contact** page of my site to allow users to send messages directly to my email without needing server-side processing.

Personas Recap from Unit 1

When designing the **contact form** functionality, I focused on three key personas from **Unit 1** to ensure the experience aligned with their needs:

1. **John** – A beginner in fitness, aged 25, who uses the site to explore workout routines and ask specific fitness questions. John prefers easy navigation and quick, straightforward ways to contact me if he has questions.
2. **Jane** – A more advanced user, 30 years old, who is already familiar with fitness but wants detailed dietary advice. Jane would likely contact me for specific meal planning queries. She values professionalism, so having a reliable and seamless contact feature is essential.
3. **Sara** – A 40-year-old working professional who is new to fitness but very tech-savvy. She appreciates interactive elements on a website and would contact me to inquire about fitness plans tailored for people with a busy schedule. Sara's experience is enhanced by a smooth, interactive contact form like the one EmailJS offers.

Reflection on Implementation Process

1. Using EmailJS for Seamless Communication

I opted for **EmailJS** to fulfill my goal of enabling easy communication without needing server-side processing. The decision was motivated by the need to offer a form that matched the tech preferences of personas like **Sara** and the usability demands of **John** and **Jane**. This service uses client-side JavaScript, which works perfectly within the course's limitations and avoids server-side configurations.

In implementing **EmailJS**:

- I used the API's `send()` function to transmit the form data (name, email, and message) as an email. This aligns with **John's** goal of getting quick responses to workout queries, and it makes the site more user-friendly for **Jane** and **Sara**, who value professionalism and technology integration.
- I ensured form validation was in place to check for empty fields and improper email formats before allowing submission. This adds to the experience of **Sara**, who expects a technically smooth interaction with the site.

2. Choosing Additional API

In addition to EmailJS, I considered adding an embedded **Google Maps** feature that could help **Jane** and **Sara** find gyms or healthy food stores in their area. This functionality would add to the interactive experience and enhance the site's value for these personas, but I ran out of time to integrate this.

3. Challenges and Learning Outcomes

Initially, I encountered an error (HTTP 418), which was due to using an outdated EmailJS SDK. After researching the issue, I updated the SDK to the latest version, and the service worked as expected. This process taught me the importance of keeping external libraries updated and understanding API errors.

The integration with EmailJS allowed me to improve my JavaScript skills, particularly around asynchronous requests using the `emailjs.send()` function. I had to carefully manage the form data and ensure proper validation, which improves the experience for all personas by reducing the likelihood of submission errors.

How the Project Enhances the Personas' Experience

- **John's Experience**: The seamless, error-checked contact form makes it easy for John to submit workout-related questions. He values simplicity, and the EmailJS integration ensures he doesn't have to navigate complex processes.
- **Jane's Experience**: Jane, who values professionalism, is reassured by the smooth user interface and the fast response she receives via email after filling out the form. Her interactions are handled promptly and professionally, thanks to EmailJS.
- **Sara's Experience**: Sara enjoys the modern technology used to facilitate communication. The interactive contact form, paired with well-functioning validation, makes her feel that the site is trustworthy and up-to-date with the latest web technologies.

Conclusion

This unit greatly helped me understand how to work with external APIs to enhance user experience. The **EmailJS** integration was a key feature that added dynamic communication capabilities to the site. Although I only integrated one service, this was a valuable learning experience that strengthened my ability to manage client-side APIs and improve the overall functionality of the website based on the personas developed in Unit 1.

For future improvements, I plan to add additional external APIs like Google Maps, which could further enhance the experience for **Jane** and **Sara**, making the site even more interactive and useful for fitness enthusiasts.