**ICT 502 INTERNET AND WEB DEVLOPMENT**

**ASSESMENT-2 REPORT (Week-7)**

**GROUP MEMBERS**

|  |  |
| --- | --- |
| **STUDENT NAME** | **STUDENT ID** |
| Dhruvkumar Patel | S20242123 |
| Dhaval Patel | S20242124 |
| Pragneshkumar Patel | S20241349 |
| Jatin Patel | S20241894 |

Contents

[Introduction 3](#_Toc165815589)

[HTML Design 3](#_Toc165815590)

[Stylesheet 5](#_Toc165815591)

[Scripting 6](#_Toc165815592)

[Logic for Trip Calculator 8](#_Toc165815593)

[Logic for Animation 8](#_Toc165815594)

[Responsiveness 9](#_Toc165815595)

[Challenges Faced 10](#_Toc165815596)

# Introduction

Welcome to our Travel Agency website! We specialize in curating unforgettable travel experiences tailored to your desires. Whether you're dreaming of an adventurous journey through exotic landscapes or a serene retreat on pristine beaches, our Travel Agency is here to transform your wanderlust into reality. Discover expert guidance, personalized service, and a world of possibilities as you embark on a journey of exploration with us. From planning your dream vacation to calculating trip costs and reaching out for personalized travel inquiries, our website serves as your one-stop destination for all things travel-related.

# HTML Design

Document Type Declaration (<!DOCTYPE html>): Specifies the document type and version of HTML used.

**HTML Structure:**

<html>: The root element of the HTML document.

<head>: Contains metadata, such as character set, viewport settings, and the website title.

<meta>: Provides metadata about the HTML document, such as character encoding and viewport settings.

<title>: Sets the title of the website displayed in the browser tab.

<link>: Links an external CSS file for styling the website.

<body>: Contains the visible content of the website.

Header (<header>): Displays the logo or website name and navigation menu.

<h1>: Represents the main heading of the website (e.g., "Travel Agency").

<nav>: Contains navigation links to different sections of the website.

<ul> and <li>: Unordered list and list items for organizing navigation links.

Sections (<section>): Divides the content into distinct sections.

Each section has a unique ID (home, calculator, contact) for navigation purposes.

Contains descriptive content about the agency's services, trip cost calculator, and contact form.

Footer (<footer>): Displays copyright information.

Contains a paragraph (<p>) with copyright details.

Script (<script>): Links an external JavaScript file for adding interactivity to the website.

Navigation Bar:

Explanation:

<header>: This is the header section of the webpage, typically used for branding and navigation.

<h1>: This is the main heading of the webpage, which in this case, displays the name of the travel agency, "Travel Agency".

<nav>: This is a semantic HTML5 element used to define a section of navigation links.

<ul>: This is an unordered list element used to contain the navigation links.

<li>: These are list items that contain individual navigation links.

<a>: These are anchor elements used to create hyperlinks. Each anchor element has an href attribute specifying the target location within the webpage (using the # followed by the section id).

The navigation bar consists of three links:

Home: When clicked, it scrolls to the section with the id "home".

Trip Cost Calculator: When clicked, it scrolls to the section with the id "calculator".

Contact: When clicked, it scrolls to the section with the id "contact".

Footer Design:

The footer is designed using semantic HTML elements and CSS styling. Here's how it's structured:

<footer>

<p>@2024 Travel Agency. All rights reserved.</p>

</footer>

Explanation:

<footer>: This is a semantic HTML5 element used to define the footer section of the webpage. It typically contains information about the website, copyright notices, contact information, etc.

Inside the <footer> element, there is a <p> (paragraph) element containing the copyright information:

<p>: This is a paragraph element used to structure and format text content within the footer.

The text content inside the <p> element displays the copyright information, "© 2024 Travel Agency. All rights reserved."

In terms of styling, the footer is designed using CSS to achieve a visually appealing appearance. Here's a simplified version of the CSS styling that might be applied:

css

Copy code

footer {

background-color: #333;

color: #fff;

text-align: center;

padding: 20px;

}

Explanation of CSS styling:

background-color: Sets the background color of the footer to a dark shade (#333).

color: Sets the text color of the footer to white (#fff).

text-align: Aligns the text content within the footer to the center.

padding: Adds padding (space) around the content inside the footer to improve spacing and aesthetics.

# Stylesheet

In the CSS file, we have included the following styles:

**Resetting Default Margin and Padding:**

Resets the default margin and padding of all elements to 0 and ensures that the box-sizing property includes padding and borders in the element's total width and height calculations.

**Body Styling:**

Sets the font family for the entire document to Arial or sans-serif.

**Header Styling:**

Styles the header element with a dark background color, white text color, padding, and fixed positioning at the top of the viewport.

Uses flexbox to align items horizontally with space between them.

**Navigation Styling:**

Styles the navigation menu with list items displayed inline and margin-right spacing between them.

Sets link text color to white and removes underlines.

**Page Styling:**

Applies padding to elements with the class page.

**Footer Styling:**

Styles the footer element with a dark background color, white text color, padding, and center alignment.

**Fade-in Animation:**

Defines a fade-in animation using keyframes, transitioning the opacity of an element from 0 to 1 over 1 second.

Applies the fade-in animation to elements with the class fade-in.

**Home Page Styling:**

Styles the home section with a background image covering the viewport, centered text content, and vertical alignment using flexbox.

**Trip Cost Calculator Section Styling:**

Adds margin and padding to the trip cost calculator section.

**Table Styling:**

Styles the table with full width, collapsed borders, and alternating row colors.

Sets padding, text alignment, and border-bottom for table headers and cells.

Applies background colors to alternate rows and hover effect.

**Fade-in Animation for Sections:**

Initially hides sections with opacity 0.

Defines a smooth transition for opacity changes.

Shows sections with fade-in effect when they have the class show.

# Scripting

**DOMContentLoaded Event Listener:**

* Listens for the DOMContentLoaded event, which fires when the initial HTML document has been completely loaded and parsed.
* Ensures that the script executes after the HTML content is ready.

**Trip Cost Calculator:**

* Retrieves references to the trip calculator form (tripForm) and the element where the trip cost result will be displayed (costResult).
* Adds a submit event listener to the tripForm.
* Prevents the default form submission behavior.
* Retrieves the values entered by the user for destination, duration, and budget from the form.
* Calls the calculateTripCost function to calculate the trip cost based on the destination and duration.
* Updates the content of the costResult element with the calculated trip cost, indicating whether it is within the budget or not.

**calculateTripCost Function:**

* Defines a function calculateTripCost that takes destination and duration as parameters.
* Calculates the trip cost based on a predetermined logic (in this example, a fixed cost per day multiplied by the duration).

**Contact Form Submission:**

* Retrieves a reference to the contact form (contactForm).
* Adds a submit event listener to the contactForm.
* Prevents the default form submission behavior.
* Retrieves the values entered by the user for name, email, and message from the form using FormData.
* Logs the form data to the console (example: name, email, message).
* Clears the form after submission using contactForm.reset().
* Displays an alert confirming that the message has been submitted.

**Scrolling Animations:**

* Retrieves references to all sections of the website.
* Defines a scrollAnimation function that checks if each section is within view as the user scrolls.
* Adds a class show to sections that are 30% visible in the viewport, triggering a fade-in animation.
* Attaches the scrollAnimation function to the scroll and resize events of the window to update animations accordingly.
* Calls scrollAnimation() once to check for initial animations when the page loads.

# Logic for Trip Calculator

**Retrieve Form Elements:** The script starts by retrieving references to the trip calculator form (tripForm) and the element where the trip cost result will be displayed (costResult). This is done using document.getElementById().

**Form Submission Event Listener:** An event listener is added to the tripForm for the 'submit' event. When the form is submitted, the event listener callback function is executed.

**Prevent Default Form Submission:** The first action within the event listener is to prevent the default form submission behavior using event.preventDefault(). This ensures that the form does not perform its default action, which would cause the page to reload.

**Retrieve Form Data:** The script retrieves the values entered by the user for destination, duration, and budget from the form elements using tripForm.elements['...'].value.

**Calculate Trip Cost:** The calculateTripCost function is called with the destination and duration as parameters. This function calculates the trip cost based on the input values. In the provided example, the calculation logic is simple: it multiplies a fixed cost per day by the duration of the trip. The result is stored in the tripCost variable.

**Display Trip Cost Result:** The script updates the content of the costResult element with the calculated trip cost. If the trip cost is within the budget (i.e., less than or equal to the budget entered by the user), a message indicating that the trip is within the budget is displayed. If the trip cost exceeds the budget, a message indicating that the trip exceeds the budget is displayed.

# Logic for Animation

**Retrieve Sections:** The script starts by retrieving references to all sections of the website using document.querySelectorAll('section'). This selects all HTML elements with the <section> tag.

**Scroll Animation Function:** The scrollAnimation function is defined to handle the scrolling animation logic. This function is responsible for checking if each section is within view as the user scrolls down the page.

**Calculate Section Position:** For each section, the function calculates its position relative to the viewport. It gets the section's top offset from the top of the document (section.offsetTop), its height (section.clientHeight), and the height of the viewport (window.innerHeight).

**Check If Section Is Visible:** It then compares the section's position with the scroll position (window.scrollY). If the section is within approximately 30% of the viewport height from the top of the viewport, it is considered to be in view. The condition if (window.scrollY > sectionTop - windowHeight + sectionHeight \* 0.3) checks if the section is sufficiently visible in the viewport.

**Add Animation Class:** If a section is deemed to be in view, the function adds the CSS class show to that section. This class triggers the fade-in animation defined in the CSS stylesheet. Adding the show class applies the fade-in effect to the section, gradually transitioning its opacity from 0 to 1.

**Event Listeners:** The scrollAnimation function is called: When the window is scrolled (window.addEventListener('scroll', scrollAnimation)). When the window is resized (window.addEventListener('resize', scrollAnimation)). Additionally, scrollAnimation() is called once initially to check for any sections already in view when the page loads.

**CSS Fade-in Animation:** The CSS stylesheet (styles.css) contains a fade-in animation (@keyframes fade-in) that gradually changes the opacity of an element from 0 to 1. The show class is used to apply this animation to sections when they enter the viewport.

# Responsiveness

* Media Queries: Media queries are used in CSS to apply different styles based on the characteristics of the device, such as its screen width, height, and orientation.
* Fluid Layouts: Elements on the webpage are sized using relative units like percentages (%) or viewport units (vw, vh) instead of fixed pixel values. This allows them to scale proportionally based on the size of the viewport.
* Flexible Images: Images are styled to be fluid, meaning they resize proportionally to fit within their containing elements. This prevents images from overflowing or becoming distorted on smaller screens.
* Viewport Meta Tag: The viewport meta tag is included in the HTML <head> section to control the viewport behavior and ensure proper scaling on mobile devices.
* Responsive Typography: Font sizes are specified using relative units like ems or rems, allowing text to scale appropriately across different screen sizes.

We thoroughly tested the Travel Agency website's responsiveness using Google Chrome Developer Tools, specifically simulating the viewport of an iPad Mini and an iPhone 12 Pro. By emulating these devices, we ensured that the website's layout, content, and styling adapt appropriately to different screen sizes and resolutions. We verified that all elements were displayed correctly, fonts were legible, images were appropriately scaled, and navigation remained accessible and functional. This meticulous testing process confirmed that users accessing the website from these devices would have a seamless and optimized browsing experience, reinforcing the website's usability and accessibility across various platforms.

# Challenges Faced

|  |  |  |
| --- | --- | --- |
| **Challenge** | **Description** | **Resolution** |
| Cross-browser Compatibility | Ensuring consistent behavior and appearance across different web browsers. | Test extensively on various browsers and versions. Use feature detection and vendor prefixes. |
| Responsive Design | Creating a layout that adapts seamlessly to different screen sizes and devices. | Implement responsive design techniques like media queries. Test on multiple devices and screen resolutions. |
| Performance Optimization | Optimizing code and assets to ensure fast loading times and smooth user experience. | Minimize file sizes, reduce HTTP requests, and optimize code for efficiency. Use CSS animations instead of JavaScript. |
| Animation Timing and Smoothness | Ensuring animations start and end at the right times, and move smoothly without lag or stuttering. | Fine-tune animation timing and easing functions. Test on different devices and network conditions. |
| Form Validation and Error Handling | Validating user input and providing informative error messages for invalid submissions. | Implement robust form validation logic using HTML5 attributes or JavaScript. Display clear error messages to users. |
| Scroll Event Optimization | Managing scroll event listeners efficiently to minimize performance overhead. | Use debouncing or throttling techniques to limit the frequency of scroll events. Consider using the Intersection Observer API for more efficient scroll detection. |
| Accessibility Considerations | Ensuring the website is accessible to users with disabilities and compatible with assistive technologies. | Follow accessibility guidelines (e.g., WCAG) and implement features like keyboard navigation and screen reader compatibility. |