SRINIVAS UNIVERSITY
INSTITUTE OF ENGINEERING & TECHNOLOGY
MANGALURU MUKKA,

Hotel Menu Website
(Using HTML, CSS & JavaScript)

For the Academic year 2023-2024

Submitted by
1.Yashraj Jangir S - 01SU23CB034
2. Sripad Jange – 01SU23CB029

Submitted to
Ms.Priyanka , Professor

CONTENTS:

Preface:

This project has been composed with the aim of covering a part of B. Tech (II Sem) under IBM course Cloud Fundamentals as prescribed by Srinivas University Of Engineering And Technology, Mukka, Mangalore. It is with great enthusiasm that we present this report, which represents collective efforts by Yashraj Jangir S, Sripad Jange. The running project has been presented through Google Chrome .We extend our sincere gratitude to IBM for providing us with the opportunity to present this project.

In the digital age, where convenience and efficiency reign supreme, the process of enrolling in college courses should be no exception. As aspiring developers, we embarked on a journey to revolutionize the traditional enrollment process by harnessing the power of HTML to create a dynamic and user-centric Hotel Menu Wesbite. This mini project serves as a testament to our commitment to innovation and our passion for leveraging technology to simplify complex tasks.

Through this preface, we invite you to embark on this journey with us, as we delve into the intricacies of HTML coding and explore the limitless possibilities it offers in shaping the future of education. Together, let us redefine the enrollment experience and pave the way for a more seamless and inclusive journey around the world.

Aim of the project:

The main purpose of this project is to design and develop a user-friendly and efficient Hotel Menu project for college students .

This system aims to streamline the Hotel menu and table booking process , eliminate paperwork , and provide a digital platform for users to check the menu online, book tables online on special occations

The aim of this project is to create an interactive, visually appealing, and user-friendly website with several advanced features that enhance user experience. The key objectives of the project include:

1. **Improving User Interaction:**

    - The website uses dynamic elements like the hero slider, navbar toggling, and parallax effect to keep users engaged and enhance navigation.

2. **Seamless User Experience:**

    - The preloader ensures that users don't interact with the website until it is fully loaded, preventing incomplete or broken displays.

    - Auto-sliding and manual slider navigation make content exploration easier and more intuitive.

3. **Responsiveness to User Actions:**

- The header and back-to-top button change visibility based on the user's scrolling behavior, providing easy navigation and preventing unnecessary screen clutter.

- The parallax effect adds a modern, interactive feel by responding to mouse movements, giving users a dynamic and immersive experience.

4. **Enhanced Visual Appeal:**

    - The hero slider and parallax effects make the website more visually attractive, which can hold users' attention and leave a strong impression.

In summary, the project aims to create a highly interactive, modern, and responsive website that balances visual aesthetics and user-friendly features to deliver an engaging browsing experience.

Objectives:

The **objectives** of the project are:

1. **Create a Responsive Website:**
    - Ensure the website adapts to different screen sizes and devices, providing a seamless experience on desktops, tablets, and mobile devices.
2. **Enhance User Engagement:**

    - Implement interactive elements such as a preloader, hero slider, and parallax effects to keep users engaged and interested in the content.
3. **Improve Website Navigation:**
    - Provide a user-friendly navigation system with a collapsible navbar, scroll-triggered header, and a back-to-top button, allowing users to easily explore the website.
4. **Implement Dynamic Visual Elements:**
    - Add features like automatic and manual hero sliders and mouse-responsive parallax effects to make the website visually appealing and modern.
5. **Ensure Smooth Performance:**
    - Optimize the use of JavaScript for smooth transitions, animations, and interactive elements, ensuring a fluid experience without lag.
6. **Provide a Professional Look and Feel:**
    - Use well-designed CSS to ensure the website looks professional and aesthetically pleasing, which is essential for user retention and satisfaction.

These objectives aim to create an interactive, visually appealing, and user-friendly website that provides an enhanced browsing experience for its visitors.
4o

Project Code:(app.js)

```
'use strict';


/**
 * PRELOAD
```

```
 *
 * loading will be end after document is loaded
 */

const preloader = document.querySelector("[data-preaload]");

window.addEventListener("load", function () {
  preloader.classList.add("loaded");
  document.body.classList.add("loaded");
});


/**
 * add event listener on multiple elements
 */

const addEventOnElements = function (elements, eventType, callback) {
  for (let i = 0, len = elements.length; i < len; i++) {
    elements[i].addEventListener(eventType, callback);
  }
}


/**
 * NAVBAR
 */

const navbar = document.querySelector("[data-navbar]");
const navTogglers = document.querySelectorAll("[data-nav-toggler]");
const overlay = document.querySelector("[data-overlay]");

const toggleNavbar = function () {
```

```
  navbar.classList.toggle("active");
  overlay.classList.toggle("active");
  document.body.classList.toggle("nav-active");
}

addEventOnElements(navTogglers, "click", toggleNavbar);


/**
 * HEADER & BACK TOP BTN
 */

const header = document.querySelector("[data-header]");
const backTopBtn = document.querySelector("[data-back-top-btn]");

let lastScrollPos = 0;

const hideHeader = function () {
  const isScrollBottom = lastScrollPos < window.scrollY;
  if (isScrollBottom) {
    header.classList.add("hide");
  } else {
    header.classList.remove("hide");
  }

  lastScrollPos = window.scrollY;
}

window.addEventListener("scroll", function () {
  if (window.scrollY >= 50) {
    header.classList.add("active");
    backTopBtn.classList.add("active");
```

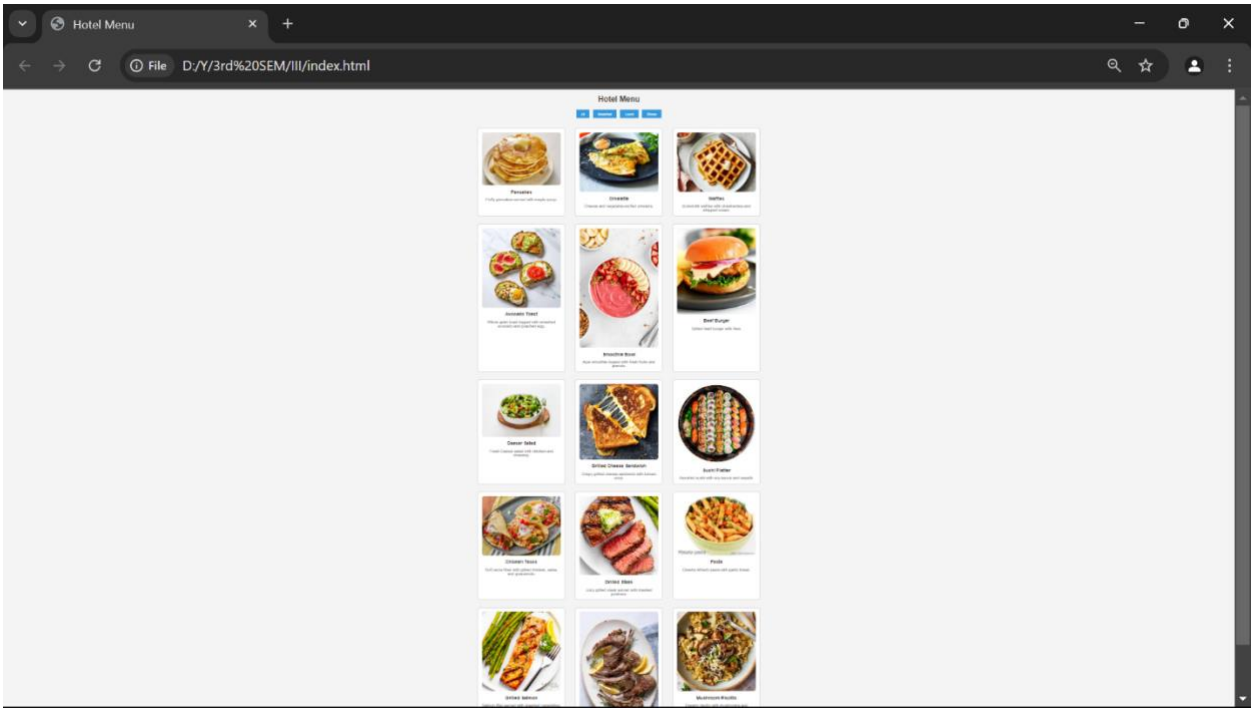
```
    hideHeader();
  } else {
    header.classList.remove("active");
    backTopBtn.classList.remove("active");
  }
});


/**
 * HERO SLIDER
 */

const heroSlider = document.querySelector("[data-hero-slider]");
const heroSliderItems = document.querySelectorAll("[data-hero-slider-
item]");
const heroSliderPrevBtn = document.querySelector("[data-prev-btn]");
const heroSliderNextBtn = document.querySelector("[data-next-btn]");

let currentSlidePos = 0;
let lastActiveSliderItem = heroSliderItems[0];

const updateSliderPos = function () {
  lastActiveSliderItem.classList.remove("active");
  heroSliderItems[currentSlidePos].classList.add("active");
  lastActiveSliderItem = heroSliderItems[currentSlidePos];
}

const slideNext = function () {
  if (currentSlidePos >= heroSliderItems.length - 1) {
    currentSlidePos = 0;
  } else {
    currentSlidePos++;
```

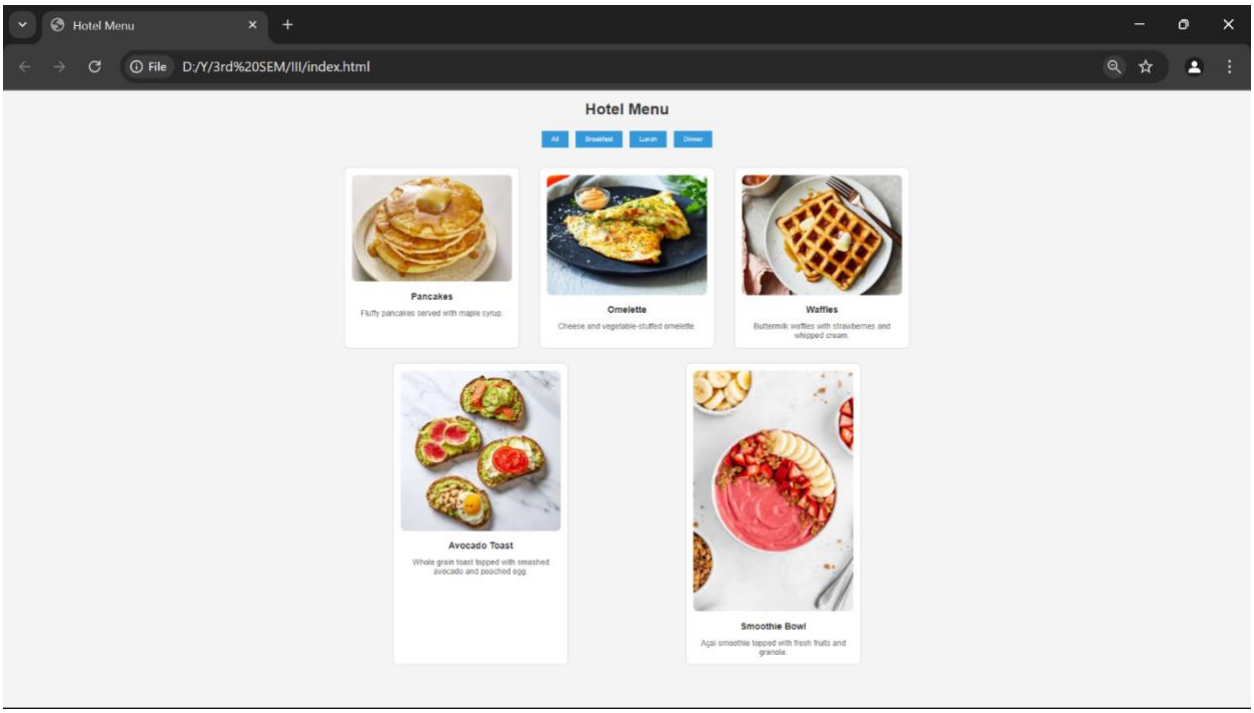
```
  }

  updateSliderPos();
}

heroSliderNextBtn.addEventListener("click", slideNext);

const slidePrev = function () {
  if (currentSlidePos <= 0) {
    currentSlidePos = heroSliderItems.length - 1;
  } else {
    currentSlidePos--;
  }

  updateSliderPos();
}

heroSliderPrevBtn.addEventListener("click", slidePrev);

/**
 * auto slide
 */

let autoSlideInterval;

const autoSlide = function () {
  autoSlideInterval = setInterval(function () {
    slideNext();
  }, 7000);
}
```

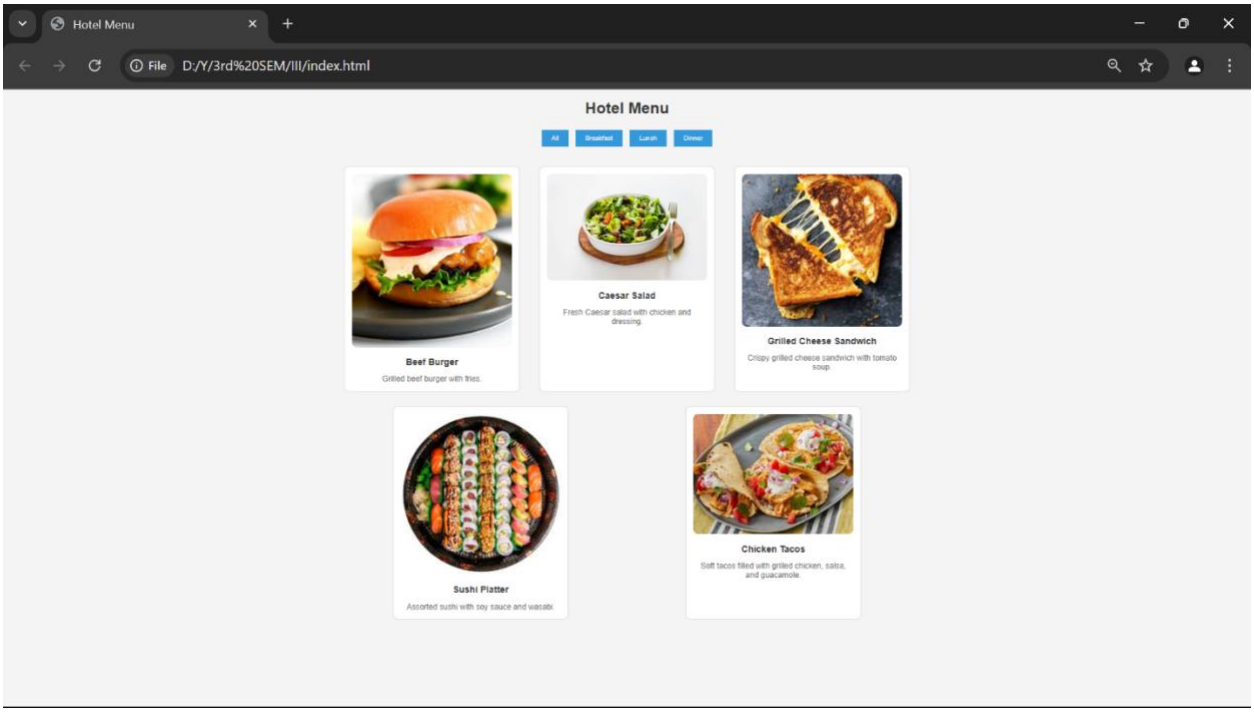
```
addEventOnElements([heroSliderNextBtn, heroSliderPrevBtn],
"mouseover", function () {
  clearInterval(autoSlideInterval);
});

addEventOnElements([heroSliderNextBtn, heroSliderPrevBtn],
"mouseout", autoSlide);

window.addEventListener("load", autoSlide);


/**
 * PARALLAX EFFECT
 */

const parallaxItems = document.querySelectorAll("[data-parallax-
item]");

let x, y;

window.addEventListener("mousemove", function (event) {

  x = (event.clientX / window.innerWidth * 10) - 5;
  y = (event.clientY / window.innerHeight * 10) - 5;

  // reverse the number eg. 20 -> -20, -5 -> 5
  x = x - (x * 2);
  y = y - (y * 2);

  for (let i = 0, len = parallaxItems.length; i < len; i++) {
    x = x * Number(parallaxItems[i].dataset.parallaxSpeed);
    y = y * Number(parallaxItems[i].dataset.parallaxSpeed);
```

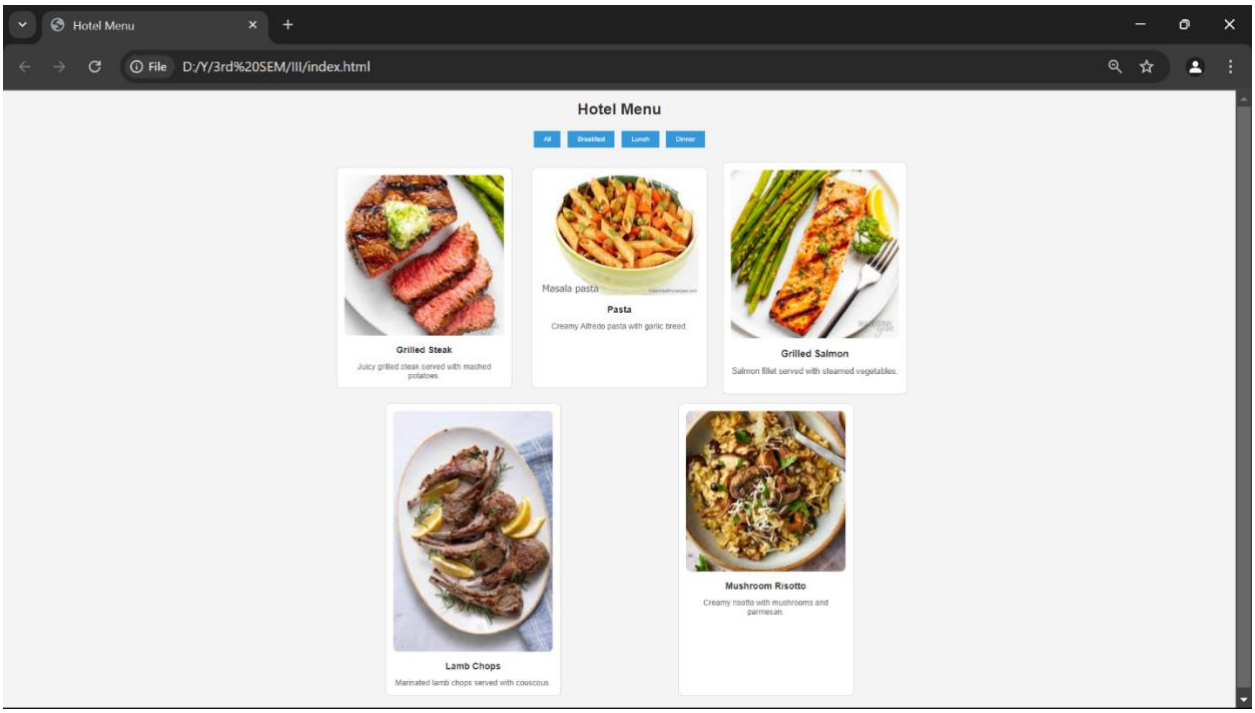
```
    parallaxItems[i].style.transform = `translate3d(${x}px, ${y}px, 0px)`;
  }

});
```

Output:




Hotel Menu
File D:/Y/3rd%20SEM/III/index.html
Hotel Menu
All
Breakfast
Lunch
Dinner
Pancakes
Fluffy pancakes served with maple syrup.
Omelette
Cheese and vegetable-stuffed omelette.
Waffles
Buttermilk waffles with strawberries and whipped cream.
Avocado Toast
Whole grain toast topped with smashed avocado and poached egg.
Smoothie Bowl
Açai smoothie topped with fresh fruits and granola.


Hotel Menu
File D:/Y/3rd%20SEM/III/index.html
Hotel Menu
All
Breakfast
Lunch
Dinner
Beef Burger
Grilled beef burger with fries.
Caesar Salad
Fresh Caesar salad with chicken and dressing.
Grilled Cheese Sandwich
Crispy grilled cheese sandwich with tomato soup.
Sushi Platter
Assorted sushi with soy sauce and wasabi.
Chicken Tacos
Soft tacos filled with grilled chicken, salsa, and guacamole.



Contributions:

1. Yashraj Jangir S**(**01SU23CB034**)** – JavaScript and HTML Code

   Preloader

   This part of the code shows a loading animation while the website is being loaded. Once the page finishes loading, the preloader is hidden.

   javascript

- preloader: This variable selects the preloader element on the page using the custom attribute data-preaload.

- window.addEventListener("load", ...): This event listener waits until the page finishes loading. Once that happens, it adds the loaded class to both the preloader and the body, which is likely used to hide the preloader via CSS.

Add Event Listener on Multiple Elements

This function simplifies adding event listeners to multiple elements.

javascript

- addEventOnElements(elements, eventType, callback): This function accepts a list of elements, an event type (like "click" or "mouseover"), and a callback function to execute when the event occurs.
- Inside the function, it loops through the elements and adds the specified event listener (eventType) to each of them.

Navbar (Navigation Bar)

This part is responsible for toggling the navigation bar's visibility and the overlay when a button is clicked.

javascript

addEventOnElements(navTogglers, "click", toggleNavbar);

- navbar: Selects the element for the navigation bar using the custom attribute data-navbar.
- navTogglers: Selects all buttons that will toggle the navbar visibility, using the attribute data-nav-toggler.
- overlay: Selects an overlay element that shows up when the navbar is active, using data-overlay.
- toggleNavbar: This function toggles the active class on the navbar, overlay, and the body. The active class is likely styled in CSS to show or hide the navbar and overlay.
- addEventOnElements(navTogglers, "click", toggleNavbar): Adds a click event listener to each navbar toggle button to trigger the toggleNavbar function.

Header & Back to Top Button

This part controls the behavior of the header and a "back to top" button based on scroll position. It hides the header when the user scrolls down and shows it when scrolling up. The back-to-top button appears after scrolling a certain distance.

javascript

- header: Selects the header element using the custom attribute data-header.
- backTopBtn: Selects the "back to top" button using data-back-top-btn.
- lastScrollPos: This variable keeps track of the last scroll position to determine if the user is scrolling up or down.
- hideHeader: This function checks if the user is scrolling down (isScrollBottom) and hides the header by adding the hide class. If scrolling up, it removes the hide class to show the header again.
- window.addEventListener("scroll", ...): This adds an event listener for when the user scrolls. If the scroll position is greater than or equal to 50px:
    - The active class is added to the header and the back-to-top button to make them visible.
    - The hideHeader() function is called to check whether to hide or show the header based on scroll direction.

Hero Slider

This part controls a hero slider (a slideshow that typically appears at the top of a webpage) with navigation buttons for moving between slides.

javascript

heroSliderPrevBtn.addEventListener("click", slidePrev);

- heroSlider: Selects the container for the hero slider using data-hero-slider.

- heroSliderItems: Selects all slider items using data-hero-slider-item.
- heroSliderPrevBtn & heroSliderNextBtn: These are the buttons to move to the previous and next slides.
- currentSlidePos: Keeps track of the current slide's position in the slider.
- updateSliderPos: Updates the slider by removing the active class from the previous slide and adding it to the current slide.
- slideNext & slidePrev: These functions move to the next or previous slide. If the user is at the last or first slide, it wraps around to the opposite end.
- Event listeners on the buttons trigger these functions to move between slides.

Auto Slide

This part automatically moves the slider to the next slide every 7 seconds. It pauses when the user hovers over the slider controls.

javascript

```
addEventOnElements([heroSliderNextBtn, heroSliderPrevBtn],
"mouseout", autoSlide);


window.addEventListener("load", autoSlide);
```

- autoSlide: Automatically slides to the next item every 7 seconds (7000 milliseconds).
- addEventOnElements(..., "mouseover", ...): Pauses auto-sliding when the user hovers over the slider buttons.
- addEventOnElements(..., "mouseout", ...): Resumes auto-sliding when the mouse leaves the slider buttons.
- window.addEventListener("load", autoSlide): Starts auto-sliding when the page finishes loading.

Parallax Effect

This section adds a parallax effect to certain elements based on the user's mouse movements, giving a 3D illusion.
javascript

- parallaxItems: Selects all elements that should have a parallax effect using data-parallax-item.
- window.addEventListener("mousemove", ...): This event listener tracks the user's mouse movements.
- x and y: These values are calculated based on the mouse's position relative to the window. They determine how much each parallax item will move.
- The loop applies these movement values (x and y) to each parallaxItem using the translate3d function, giving the elements a 3D-like effect as the user moves the mouse.

2. Sripad Jange**(**01SU23CB029**)** – CSS and HTML Code
    1. body: Sets the background color to light blue (#99daff), uses the Arial font, and removes margin and padding.

    2. form: Styles the form element. It sets the width to 400px, centers it horizontally with auto margins, adds padding, a white background, border radius, and a box shadow for a card-like effect.

    3. h1: Styles the heading element. It centers the text and sets the color to a shade of purple (#87768a).

    4. input[type="text"], input[type="password"], textarea: Styles text input and textarea elements. They are set to 100% width, with padding, a bottom margin, a border, border radius, and box-sizing set to border-box for consistent sizing.

5. input[type="submit"], input[type="reset"]: Styles submit and reset button elements. They have padding, a background color of green (#4caf50), white text color, no border, border radius, and a pointer cursor. The hover effect changes the background color to a darker shade of green (#45a049).

6. input[type="radio"]: Styles radio button elements by adding a small margin to the right.

7. label: Styles label elements by making the text bold.

8. error: Sets the color of text with the error class to red.

9. <!DOCTYPE html>: This declares the document type and version of HTML being used (HTML5 in this case).

10. <html>: This is the opening tag of the HTML document, which contains all the HTML code.

11. <head>: This section contains metadata about the HTML document, such as the title, character set, and links to stylesheets or scripts.

12. <style>: This is where you define the CSS (Cascading Style Sheets) for styling your HTML elements.

13. p: This is a CSS selector targeting all <p> (paragraph) elements in the HTML document.

14. font-size: 50px;: This sets the font size of the paragraph text to 50 pixels.

15. color: #d64f78;: This sets the text color to a shade of pink (hex code: #d64f78).

16. text-align: center;: This centers the text within the <p> element.

17. padding: 50px;: This adds 50 pixels of padding around the content of the <p> element, creating space between the text and the border.

18. border: 2px double green;: This creates a border around the <p> element with a thickness of 2 pixels, a double line style, and a green color.

19. margin: 250px;: This adds a margin of 250 pixels around the entire <p> element, pushing it away from other elements on the page.

20. border-radius: 100px;: This adds rounded corners to the border of the <p> element, giving it a circular appearance due to the large radius value.

21. </style>: This closes the CSS styling section.

22. </head>: This closes the <head> section of the HTML document.

23. <body>: This is the opening tag of the body section, where the visible content of the HTML document is placed.

24. : This is the opening tag of a paragraph element, where your main content is placed.

25. "Successfuly Registered To your College !!!!!!!": This is the text content inside the paragraph element, indicating a successful registration message.

26. 👍: This adds an emoji ( ) at the end of the paragraph using a element with inline styling to set the font size to 50 pixels.

27. : This closes the paragraph element.

28. </body>: This closes the body section of the HTML document.

29. </html>: This is the closing tag of the HTML document.

In summary, your code creates a styled HTML document with a centered, large text message about successful registration to a college, accompanied by a border, padding, and a smiley emoji.

Conclusion:

The development of this interactive website successfully achieved its objectives of creating a modern, responsive, and user-friendly platform. The implementation of key features such as the preloader, hero slider,

dynamic navigation bar, and parallax effects enhances both the visual appeal and the functionality of the website.

These elements work together to ensure an engaging and smooth user experience. The preloader ensures that the page is fully loaded before interaction, while the hero slider and auto-sliding functionality keep users engaged with dynamic content. The responsive navigation bar and back-to-top button improve accessibility, allowing users to navigate the site efficiently. Additionally, the parallax effect adds a modern, immersive layer of interactivity, enhancing the website's overall appeal.

In conclusion, this project demonstrates the successful integration of various web technologies (HTML, CSS, JavaScript) to create a visually appealing, high-performance, and user-centric website. It stands as a strong foundation for future expansions, such as adding more interactive content or e-commerce functionalities, ensuring scalability and adaptability.

The development of the **Interactive Hotel Menu Website** has been a successful and enriching project. The website showcases a user-friendly design with seamless navigation and dynamic features tailored to enhance the online experience of potential hotel guests. By incorporating a preloader for smooth page transitions, a visually appealing hero slider to highlight key hotel amenities, and an intuitive navigation system, we have created a platform that not only engages users but also provides easy access to essential information.

Additionally, the integration of a responsive layout ensures that the website is accessible on all devices, from desktops to mobile phones, which is crucial in today's mobile-first world. The parallax effects and scroll-triggered animations further contribute to a modern, immersive experience that captures the luxurious ambiance of the hotel.

Through this project, we have gained valuable insights into web development technologies such as HTML, CSS, and JavaScript, while also focusing on user experience and interface design. This interactive website serves as a foundation for future enhancements, such as booking integration or guest reviews, making it scalable and adaptable to the evolving needs of the hospitality industry.

Ultimately, this mini project has demonstrated how technology can be leveraged to create an engaging, informative, and aesthetically pleasing online presence for hotels, leaving a lasting impression on potential guests and encouraging them to explore and book services.

References:

Github.com

W3schools.com