**Option 1: Landing Page Creation**

For the landing page, you’ll be creating a simple and visually appealing page that could be for any website or business. The goal is to make a clean, responsive layout using HTML, CSS, and possibly some JavaScript if you want to add interactive elements.

**Steps to create a landing page:**

1. **Plan the Design:**
   * **Purpose:** Define the goal of the landing page. Is it to promote a product, collect leads, showcase a service, or just provide information?
   * **Content:** List all the sections you want to include (e.g., Hero section, Features, Testimonials, Contact Form, Footer).
   * **Layout:** Sketch the layout on paper or use a tool like Figma or Adobe XD.
2. **HTML Structure:**
   * Create a basic HTML structure with the following:
     + header with navigation (links to sections like Home, Features, etc.).
     + main section for the hero section (big, eye-catching image and call-to-action).
     + footer with contact info or social media links.

Example:

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Landing Page</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<header>

<nav>

<ul>

<li><a href="#home">Home</a></li>

<li><a href="#features">Features</a></li>

<li><a href="#contact">Contact</a></li>

</ul>

</nav>

</header>

<section id="hero">

<h1>Welcome to Our Service</h1>

<p>Transform your business with our amazing product!</p>

<button>Get Started</button>

</section>

<section id="features">

<h2>Features</h2>

<ul>

<li>Easy to Use</li>

<li>High Performance</li>

<li>24/7 Support</li>

</ul>

</section>

<footer id="contact">

<p>Contact us: email@example.com</p>

</footer>

</body>

</html>

1. **CSS for Styling:**
   * Create a separate CSS file (style.css) to style the landing page.
   * Focus on typography, colors, spacing, and layout using Flexbox or Grid.

Example:

css

Copy code

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

header {

background-color: #333;

color: white;

padding: 1rem;

}

header nav ul {

list-style: none;

display: flex;

justify-content: center;

}

header nav ul li {

margin: 0 15px;

}

header nav ul li a {

color: white;

text-decoration: none;

}

#hero {

background-image: url('hero.jpg');

background-size: cover;

text-align: center;

padding: 100px 20px;

}

#hero h1 {

font-size: 3rem;

color: white;

}

#hero button {

padding: 10px 20px;

font-size: 1rem;

background-color: #28a745;

color: white;

border: none;

cursor: pointer;

}

#features ul {

display: flex;

justify-content: space-around;

padding: 20px;

}

footer {

background-color: #333;

color: white;

text-align: center;

padding: 1rem;

}

1. **JavaScript (Optional):**
   * Add interactivity, like smooth scrolling for navigation links or a form validation script.

Example:

javascript

Copy code

document.querySelectorAll('a[href^="#"]').forEach(anchor => {

anchor.addEventListener('click', function(e) {

e.preventDefault();

document.querySelector(this.getAttribute('href')).scrollIntoView({

behavior: 'smooth'

});

});

});

1. **Test the Page:**
   * Make sure it’s responsive by testing on different screen sizes (mobile, tablet, desktop).
   * Validate your HTML and CSS using tools like [W3C Validator](https://validator.w3.org/).

**Option 2: To-Do List App**

If you prefer building a To-Do list application, this would be a fun way to practice your HTML, CSS, and JavaScript skills. You’ll create an interactive list where users can add, delete, and mark tasks as completed.

**Steps to create a To-Do List:**

1. **HTML Structure:**
   * Set up the basic structure for the to-do list with an input field, a button to add tasks, and a list to display tasks.

Example:

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>To-Do List</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<div id="app">

<h1>My To-Do List</h1>

<input type="text" id="taskInput" placeholder="Add a new task">

<button id="addButton">Add Task</button>

<ul id="taskList"></ul>

</div>

<script src="script.js"></script>

</body>

</html>

1. **CSS Styling:**
   * Style the input, button, and task list.

Example:

css

Copy code

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

background-color: #f4f4f9;

}

#app {

text-align: center;

width: 300px;

background-color: white;

padding: 20px;

border-radius: 10px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

}

input {

width: 100%;

padding: 10px;

margin: 10px 0;

border-radius: 5px;

border: 1px solid #ccc;

}

button {

padding: 10px 20px;

background-color: #28a745;

color: white;

border: none;

cursor: pointer;

border-radius: 5px;

}

button:hover {

background-color: #218838;

}

ul {

list-style-type: none;

padding: 0;

}

li {

display: flex;

justify-content: space-between;

padding: 10px;

background-color: #f9f9f9;

margin: 5px 0;

border-radius: 5px;

}

li.completed {

text-decoration: line-through;

background-color: #d3f9d8;

}

1. **JavaScript for Functionality:**
   * Add JavaScript to allow users to add tasks, mark tasks as completed, and delete tasks.

Example:

javascript

Copy code

document.getElementById('addButton').addEventListener('click', function() {

const taskInput = document.getElementById('taskInput');

const taskValue = taskInput.value.trim();

if (taskValue) {

const li = document.createElement('li');

li.textContent = taskValue;

const deleteButton = document.createElement('button');

deleteButton.textContent = 'Delete';

deleteButton.addEventListener('click', () => li.remove());

li.appendChild(deleteButton);

li.addEventListener('click', () => li.classList.toggle('completed'));

document.getElementById('taskList').appendChild(li);

taskInput.value = ''; // Clear the input

}

});

1. **Test and Polish:**
   * Ensure tasks can be added, marked as completed, and deleted.
   * Test responsiveness and improve UI if needed.

**First Report Submission**

For the report, you’ll likely need to:

1. **Explain the Design & Functionality:**
   * Describe the purpose of the project (Landing Page or To-Do List).
   * Provide details about the key features, like interactive buttons, smooth scrolling, or task management.
2. **Document Your Process:**
   * Outline the steps you took to complete the project.
   * Discuss any challenges or learning moments during development.
3. **Future Improvements:**
   * Mention areas for future enhancements, such as adding local storage for the

Top of Form

Bottom of Form