**Project Documentation: Travel and Tourism Registration Form**

**Objective**

To create a user registration form for a travel and tourism website with real-time validation, dynamic feedback, and a visually appealing design using Tailwind CSS, custom styles, and animations.

**File Contents**

**1. registration-form.html**

This HTML file includes the form structure, uses Tailwind CSS for styling, and links to the custom CSS and JavaScript files.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Travel and Tourism Registration</title>

<script src="https://cdn.tailwindcss.com"></script>

<link rel="stylesheet" href="styles.css"> <!-- Link to external CSS -->

</head>

<body class="flex items-center justify-center h-screen p-6">

<div class="form-container p-8 rounded-lg shadow-lg max-w-md w-full">

<h2 class="text-2xl font-bold text-center mb-6 text-blue-700 animate\_\_animated animate\_\_fadeIn">Join Our Travel Community</h2>

<form id="registrationForm" novalidate>

<!-- Form fields here -->

</form>

</div>

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

</body>

</html>

**2. styles.css**

This CSS file handles the styling of the form, including the background image, transitions, and animations.

body {

background-image: url('images/background.jpg');

background-size: cover;

background-position: center;

background-attachment: fixed;

margin: 0;

font-family: Arial, sans-serif;

}

.form-container {

background: rgba(255, 255, 255, 0.9);

animation: fadeIn 1s ease-in-out;

}

@keyframes fadeIn {

from {

opacity: 0;

transform: translateY(-20px);

}

to {

opacity: 1;

transform: translateY(0);

}

}

.input-field {

transition: border-color 0.3s ease;

}

.input-field.valid {

border-color: #48bb78;

}

.input-field.invalid {

border-color: #f56565;

}

.error-message {

transition: opacity 0.3s ease;

}

.error-message.hidden {

opacity: 0;

}

.error-message.visible {

opacity: 1;

}

**3. validation.js**

This JavaScript file manages form validation, real-time feedback, and form submission logic.

document.addEventListener('DOMContentLoaded', () => {

const form = document.getElementById('registrationForm');

const nameInput = document.getElementById('name');

const emailInput = document.getElementById('email');

const passwordInput = document.getElementById('password');

const confirmPasswordInput = document.getElementById('confirmPassword');

const dobInput = document.getElementById('dob');

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

nameInput.addEventListener('input', () => {

const nameValue = nameInput.value.trim();

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

if (nameValue.length >= 3 && /^[A-Za-z\s]+$/.test(nameValue)) {

nameInput.classList.add('valid');

nameInput.classList.remove('invalid');

nameError.classList.add('hidden');

} else {

nameInput.classList.add('invalid');

nameInput.classList.remove('valid');

nameError.classList.remove('hidden');

}

});

emailInput.addEventListener('input', () => {

const emailValue = emailInput.value.trim();

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

const emailPattern = /^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;

if (emailPattern.test(emailValue)) {

emailInput.classList.add('valid');

emailInput.classList.remove('invalid');

emailError.classList.add('hidden');

} else {

emailInput.classList.add('invalid');

emailInput.classList.remove('valid');

emailError.classList.remove('hidden');

}

});

passwordInput.addEventListener('input', () => {

const passwordValue = passwordInput.value.trim();

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

const passwordPattern = /^(?=.\*[A-Za-z])(?=.\*\d)[A-Za-z\d]{8,}$/;

if (passwordPattern.test(passwordValue)) {

passwordInput.classList.add('valid');

passwordInput.classList.remove('invalid');

passwordError.classList.add('hidden');

} else {

passwordInput.classList.add('invalid');

passwordInput.classList.remove('valid');

passwordError.classList.remove('hidden');

}

});

confirmPasswordInput.addEventListener('input', () => {

const confirmPasswordValue = confirmPasswordInput.value.trim();

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

if (confirmPasswordValue === passwordInput.value.trim()) {

confirmPasswordInput.classList.add('valid');

confirmPasswordInput.classList.remove('invalid');

confirmPasswordError.classList.add('hidden');

} else {

confirmPasswordInput.classList.add('invalid');

confirmPasswordInput.classList.remove('valid');

confirmPasswordError.classList.remove('hidden');

}

});

dobInput.addEventListener('input', () => {

const dobValue = new Date(dobInput.value);

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

const today = new Date();

const age = today.getFullYear() - dobValue.getFullYear();

const m = today.getMonth() - dobValue.getMonth();

if (m < 0 || (m === 0 && today.getDate() < dobValue.getDate())) {

age--;

}

if (age >= 18) {

dobInput.classList.add('valid');

dobInput.classList.remove('invalid');

dobError.classList.add('hidden');

} else {

dobInput.classList.add('invalid');

dobInput.classList.remove('valid');

dobError.classList.remove('hidden');

}

});

form.addEventListener('submit', (event) => {

event.preventDefault(); // Prevent form from submitting

const isNameValid = !nameInput.classList.contains('invalid');

const isEmailValid = !emailInput.classList.contains('invalid');

const isPasswordValid = !passwordInput.classList.contains('invalid');

const isConfirmPasswordValid = !confirmPasswordInput.classList.contains('invalid');

const isDobValid = !dobInput.classList.contains('invalid');

if (isNameValid && isEmailValid && isPasswordValid && isConfirmPasswordValid && isDobValid) {

alert('Registration successful!');

// You can add further code here to actually submit the form, such as form.submit();

} else {

alert('Please correct the errors before submitting the form.');

}

});

});

**Summary**

* **Objective:** Build a user-friendly, visually appealing registration form for a travel and tourism website.
* **Files:** HTML, CSS, and JavaScript files manage the form layout, styling, and validation.
* **Styling:** Background image, form container styling, and animations are handled through CSS.
* **Validation:** JavaScript ensures that user inputs meet specified criteria, providing real-time feedback and preventing invalid form submissions.

This documentation covers the essentials of your project setup and implementation, providing a comprehensive overview of how to integrate local images and add custom styles and animations to enhance the user experience.