**MERN Expt 4**

**Name:** Rohit Deshpande

**Roll No.:** 16010122041

**Batch:** TY A2 (Comps)

### **Task: User Profile Editor**

**1. Create a form with fields for the user's name, age, and email.**

**2. Display the entered information below the form.**

**3. Add a button to toggle the visibility of the form.**

**4. Add a button to reset the form fields to their initial values.**

App.js

import React, { useState } from "react";

import "./App.css"; // Importing the CSS file

function App() {

// State to hold the form data

const [formData, setFormData] = useState({

name: '',

age: '',

email: ''

});

// State to control form visibility

const [formVisible, setFormVisible] = useState(true);

// State to manage form errors

const [errors, setErrors] = useState({});

// Handle input changes

const handleChange = (e) => {

const { name, value } = e.target;

setFormData((prevData) => ({

...prevData,

[name]: value

}));

};

// Validate form fields

const validateForm = () => {

let validationErrors = {};

if (!formData.name.trim()) {

validationErrors.name = "Name is required";

}

if (!formData.age || formData.age <= 0) {

validationErrors.age = "Age must be a positive number";

}

const emailRegex = /^[^\s@]+@[^\s@]+**\.**[^\s@]+$/;

if (!formData.email || !emailRegex.test(formData.email)) {

validationErrors.email = "Invalid email format";

}

setErrors(validationErrors);

return Object.keys(validationErrors).length === 0;

};

// Handle form submission (for demonstration purposes, not used)

const handleSubmit = (e) => {

e.preventDefault();

if (validateForm()) {

alert("Form is valid!");

} else {

alert("Form has errors");

}

};

// Toggle form visibility

const toggleFormVisibility = () => {

setFormVisible((prevVisible) => !prevVisible);

};

// Reset form fields to their initial values

const resetForm = () => {

setFormData({

name: '',

age: '',

email: ''

});

setErrors({});

};

return (

<div className="App">

<h1>User Profile Editor</h1>

{/\* Toggle form visibility \*/}

<button className="toggle-btn" onClick={toggleFormVisibility}>

{formVisible ? "Hide Form" : "Show Form"}

</button>

{/\* Form \*/}

{formVisible && (

<form onSubmit={handleSubmit}>

<div className="form-group">

<label>Name: </label>

<input

type="text"

name="name"

value={formData.name}

onChange={handleChange}

className={errors.name ? "input-error" : ""}

/>

{errors.name && <span className="error-text">{errors.name}</span>}

</div>

<div className="form-group">

<label>Age: </label>

<input

type="number"

name="age"

value={formData.age}

onChange={handleChange}

className={errors.age ? "input-error" : ""}

/>

{errors.age && <span className="error-text">{errors.age}</span>}

</div>

<div className="form-group">

<label>Email: </label>

<input

type="email"

name="email"

value={formData.email}

onChange={handleChange}

className={errors.email ? "input-error" : ""}

/>

{errors.email && <span className="error-text">{errors.email}</span>}

</div>

{/\* Reset form button \*/}

<button type="button" onClick={resetForm}>

Reset Form

</button>

</form>

)}

{/\* Display entered information \*/}

<div className="user-info">

<h2>User Information</h2>

<p><strong>Name:</strong> {formData.name || "N/A"}</p>

<p><strong>Age:</strong> {formData.age || "N/A"}</p>

<p><strong>Email:</strong> {formData.email || "N/A"}</p>

</div>

</div>

);

}

export default App;

App.css

.App {

font-family: Arial, sans-serif;

max-width: 400px;

margin: 0 auto;

padding: 20px;

background-color: #f8f8f8;

border-radius: 8px;

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

}

h1 {

text-align: center;

}

.form-group {

margin-bottom: 15px;

}

label {

display: block;

font-weight: bold;

margin-bottom: 5px;

}

input {

width: 100%;

padding: 8px;

border: 1px solid #ccc;

border-radius: 4px;

font-size: 14px;

}

input.input-error {

border-color: red;

}

.error-text {

color: red;

font-size: 12px;

}

button {

background-color: #4CAF50;

color: white;

padding: 10px 15px;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 16px;

}

button.toggle-btn {

background-color: #f39c12;

margin-bottom: 20px;

}

button:hover {

background-color: #45a049;

}

.user-info {

margin-top: 20px;

background-color: #fff;

padding: 15px;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

.user-info h2 {

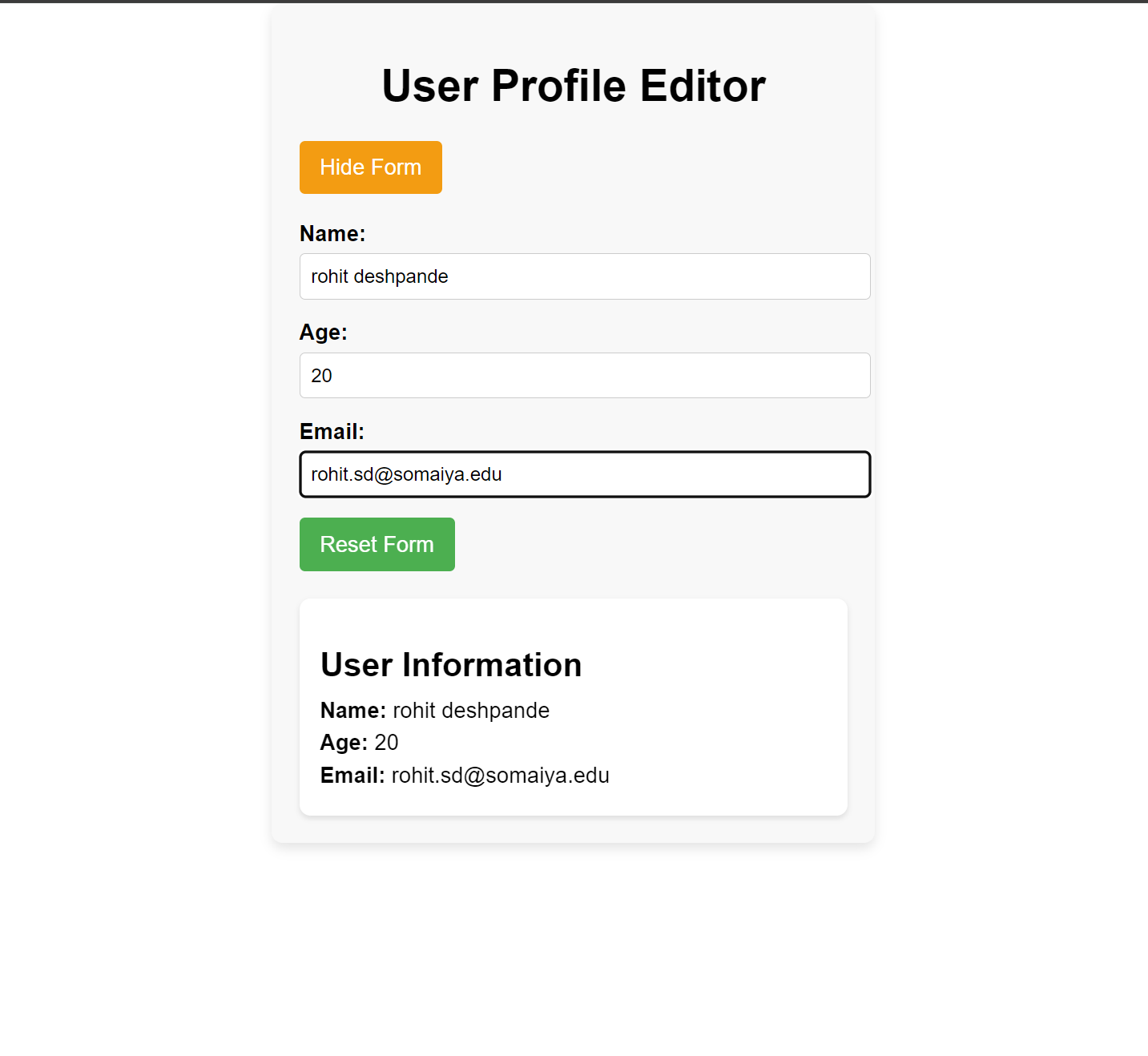
margin-bottom: 10px;

}

.user-info p {

margin: 5px 0;

}



### **Task: Task Manager**

1. **Create a task manager with the following features:**
   * **An input field to add a new task.**
   * **A list to display added tasks.**
   * **A button to mark tasks as complete.**
   * **A button to toggle the visibility of completed tasks.**
   * **A button to reset the task list.**

App.js

import React, { useState } from "react";

import "./App.css"; // Importing the CSS file

function App() {

// State for tasks

const [tasks, setTasks] = useState([]);

const [newTask, setNewTask] = useState("");

const [showCompleted, setShowCompleted] = useState(true);

// Add a new task

const addTask = () => {

if (newTask.trim() === "") return;

const task = {

id: Date.now(),

text: newTask,

completed: false,

};

setTasks([...tasks, task]);

setNewTask("");

};

// Mark task as completed

const toggleComplete = (id) => {

setTasks(tasks.map((task) =>

task.id === id ? { ...task, completed: !task.completed } : task

));

};

// Toggle visibility of completed tasks

const toggleVisibility = () => {

setShowCompleted((prev) => !prev);

};

// Reset the task list

const resetTasks = () => {

setTasks([]);

};

return (

<div className="App">

<h1>Task Manager</h1>

<div className="input-area">

<input

type="text"

placeholder="Enter a new task"

value={newTask}

onChange={(e) => setNewTask(e.target.value)}

/>

<button onClick={addTask}>Add Task</button>

</div>

<button className="toggle-btn" onClick={toggleVisibility}>

{showCompleted ? "Hide Completed Tasks" : "Show Completed Tasks"}

</button>

<button className="reset-btn" onClick={resetTasks}>Reset Tasks</button>

{/\* Task List \*/}

<ul className="task-list">

{tasks

.filter((task) => (showCompleted ? true : !task.completed))

.map((task) => (

<li key={task.id} className={task.completed ? "completed" : ""}>

<span>{task.text}</span>

<button onClick={() => toggleComplete(task.id)}>

{task.completed ? "Undo" : "Complete"}

</button>

</li>

))}

</ul>

</div>

);

}

export default App;

App.css

.App {

font-family: Arial, sans-serif;

max-width: 500px;

margin: 0 auto;

padding: 20px;

background-color: #f9f9f9;

border-radius: 8px;

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

}

h1 {

text-align: center;

}

.input-area {

display: flex;

justify-content: space-between;

margin-bottom: 15px;

}

input {

width: 70%;

padding: 10px;

font-size: 16px;

border: 1px solid #ccc;

border-radius: 4px;

}

button {

padding: 10px 10px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

button.toggle-btn {

background-color: #f39c12;

margin-top: 10px;

margin-right: 10px;

}

button.reset-btn {

background-color: #e74c3c;

margin-top: 10px;

margin-left: 10px;

}

button:hover {

opacity: 0.9;

}

.task-list {

list-style: none;

padding: 0;

margin-top: 20px;

}

.task-list li {

display: flex;

justify-content: space-between;

padding: 10px;

margin-bottom: 10px;

background-color: #fff;

border-radius: 4px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

.task-list li.completed span {

text-decoration: line-through;

color: #888;

}

.task-list li button {

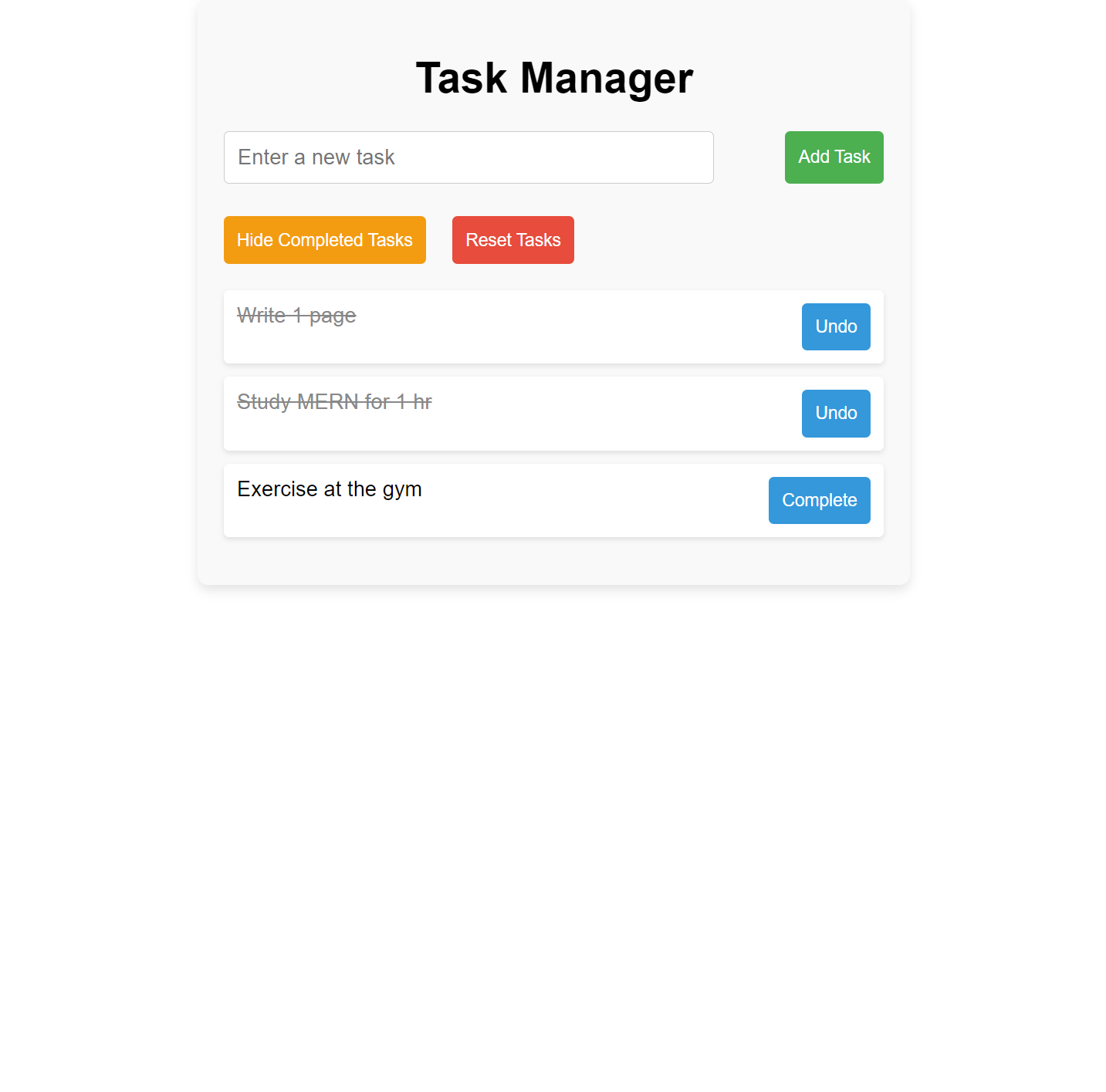
background-color: #3498db;

}

.task-list li button:hover {

opacity: 0.8;

}



### **Task: User Profile Manager**

1. **Create a form with fields for the user's name, age, and email.**
2. **Fetch initial profile data when the component mounts.**
3. **Display the fetched profile data in the form fields.**
4. **Allow the user to update their profile information.**
5. **Display the updated profile information below the form.**
6. **Log a message to the console whenever the profile data is updated.**

App.js

import React, { useState, useEffect } from "react";

import "./App.css";

function App() {

const [profile, setProfile] = useState({

name: '',

age: '',

email: ''

});

const [updatedProfile, setUpdatedProfile] = useState(null);

const [errors, setErrors] = useState({});

const [isSaving, setIsSaving] = useState(false);

const [errorMessage, setErrorMessage] = useState('');

useEffect(() => {

// Load profile data from localStorage on mount

const savedProfile = localStorage.getItem("profile");

if (savedProfile) {

setProfile(JSON.parse(savedProfile));

} else {

// Simulate fetching initial profile data from an API

const fetchProfile = async () => {

const fetchedProfile = {

name: "John Doe",

age: 28,

email: "johndoe@example.com"

};

setTimeout(() => setProfile(fetchedProfile), 1000);

};

fetchProfile();

}

}, []);

const validateForm = () => {

const newErrors = {};

if (!profile.name.trim()) {

newErrors.name = "Name is required.";

}

if (!profile.age || profile.age <= 0) {

newErrors.age = "Age must be a positive number.";

}

const emailRegex = /^[^\s@]+@[^\s@]+**\.**[^\s@]+$/;

if (!emailRegex.test(profile.email)) {

newErrors.email = "Invalid email format.";

}

setErrors(newErrors);

return Object.keys(newErrors).length === 0;

};

const handleChange = (e) => {

const { name, value } = e.target;

setProfile((prevProfile) => ({

...prevProfile,

[name]: value

}));

};

const handleSubmit = (e) => {

e.preventDefault();

if (!validateForm()) return;

setIsSaving(true);

setErrorMessage("");

// Simulate an API call to save the profile

setTimeout(() => {

const apiSuccess = Math.random() > 0.2; // Simulate 80% success rate

if (apiSuccess) {

setUpdatedProfile(profile);

localStorage.setItem("profile", JSON.stringify(profile));

console.log("Profile successfully updated:", profile);

} else {

setErrorMessage("Failed to save profile. Please try again.");

console.error("Profile update failed");

}

setIsSaving(false);

}, 1000);

};

return (

<div className="App">

<h1>User Profile Manager</h1>

<form onSubmit={handleSubmit}>

<div className="form-group">

<label>Name: </label>

<input

type="text"

name="name"

value={profile.name}

onChange={handleChange}

className={errors.name ? "input-error" : ""}

/>

{errors.name && <span className="error-text">{errors.name}</span>}

</div>

<div className="form-group">

<label>Age: </label>

<input

type="number"

name="age"

value={profile.age}

onChange={handleChange}

className={errors.age ? "input-error" : ""}

/>

{errors.age && <span className="error-text">{errors.age}</span>}

</div>

<div className="form-group">

<label>Email: </label>

<input

type="email"

name="email"

value={profile.email}

onChange={handleChange}

className={errors.email ? "input-error" : ""}

/>

{errors.email && <span className="error-text">{errors.email}</span>}

</div>

{errorMessage && <p className="error-message">{errorMessage}</p>}

<button type="submit" disabled={isSaving}>

{isSaving ? "Saving..." : "Update Profile"}

</button>

</form>

{updatedProfile && (

<div className="profile-info">

<h2>Updated Profile Information</h2>

<p><strong>Name:</strong> {updatedProfile.name}</p>

<p><strong>Age:</strong> {updatedProfile.age}</p>

<p><strong>Email:</strong> {updatedProfile.email}</p>

</div>

)}

</div>

);

}

export default App;

App.css

.App {

font-family: Arial, sans-serif;

max-width: 400px;

margin: 0 auto;

padding: 20px;

background-color: #f9f9f9;

border-radius: 8px;

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

}

h1 {

text-align: center;

}

form {

margin-bottom: 20px;

}

.form-group {

margin-bottom: 15px;

}

label {

display: block;

font-weight: bold;

margin-bottom: 5px;

}

input {

width: 100%;

padding: 8px;

border: 1px solid #ccc;

border-radius: 4px;

font-size: 14px;

}

input.input-error {

border-color: red;

}

.error-text {

color: red;

font-size: 12px;

}

button {

padding: 10px 15px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

button:disabled {

background-color: #888;

cursor: not-allowed;

}

button:hover:not(:disabled) {

background-color: #45a049;

}

.profile-info {

background-color: #fff;

padding: 15px;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

.error-message {

color: red;

font-size: 14px;

margin-bottom: 10px;

}

