<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>Advanced Event Management Portal - Interactive Placeholders</title>

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link href="https://fonts.googleapis.com/css2?family=Lato:wght@400;700&family=Poppins:wght@500;700&display=swap" rel="stylesheet">

<style>

/\* --- Keyframes for Animations --- \*/

@keyframes fadeIn { from { opacity: 0; transform: translateY(10px); } to { opacity: 1; transform: translateY(0); } }

@keyframes backgroundGradient { 0% { background-position: 0% 50%; } 50% { background-position: 100% 50%; } 100% { background-position: 0% 50%; } }

@keyframes pulse { 0% { transform: scale(1); } 50% { transform: scale(1.03); } 100% { transform: scale(1); } }

/\* --- Global Styles --- \*/

\* { box-sizing: border-box; margin: 0; padding: 0; }

body {

font-family: 'Lato', sans-serif;

line-height: 1.6;

background-color: #f8f9fa; /\* Light grey background \*/

color: #343a40; /\* Dark grey text \*/

display: flex;

min-height: 100vh;

overflow-x: hidden; /\* Prevent horizontal scroll \*/

}

/\* --- Layout --- \*/

.sidebar {

width: 250px;

background: linear-gradient(135deg, #4a5568, #2d3748); /\* Dark gradient \*/

color: #e2e8f0; /\* Light text \*/

padding: 25px 15px;

display: flex;

flex-direction: column;

height: 100vh; /\* Full height \*/

position: fixed; /\* Fixed sidebar \*/

left: 0;

top: 0;

box-shadow: 2px 0 10px rgba(0,0,0,0.1);

transition: width 0.3s ease;

z-index: 100; /\* Ensure sidebar is above content \*/

}

.sidebar h2 {

text-align: center;

margin-bottom: 30px;

font-family: 'Poppins', sans-serif;

font-weight: 700;

color: #ffffff;

letter-spacing: 1px;

}

.sidebar nav ul { list-style: none; padding: 0; flex-grow: 1; }

.sidebar nav ul li a {

display: flex; /\* Use flexbox for alignment \*/

align-items: center; /\* Vertically center icon and text \*/

color: #e2e8f0;

text-decoration: none;

padding: 12px 15px;

margin-bottom: 8px;

border-radius: 6px;

transition: background-color 0.3s ease, color 0.3s ease, padding-left 0.3s ease;

font-weight: 500; /\* Poppins medium \*/

font-family: 'Poppins', sans-serif;

}

.sidebar nav ul li a:hover,

.sidebar nav ul li a.active {

background-color: rgba(255, 255, 255, 0.15); /\* Lighter background on hover/active \*/

color: #ffffff; /\* White text \*/

padding-left: 20px; /\* Indent on hover/active \*/

}

.sidebar nav ul li a svg {

margin-right: 12px; /\* Space between icon and text \*/

width: 18px;

height: 18px;

vertical-align: middle; /\* Align icon nicely \*/

fill: currentColor; /\* Icon color matches text \*/

}

.sidebar .auth-buttons { margin-top: auto; /\* Pushes auth buttons to bottom \*/ }

.sidebar .auth-buttons button {

display: block;

width: 100%;

padding: 10px;

margin-bottom: 10px;

background-color: #4CAF50; /\* Green \*/

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s ease;

font-family: 'Poppins', sans-serif;

}

.sidebar .auth-buttons button.logout { background-color: #f44336; /\* Red \*/ }

.sidebar .auth-buttons button:hover { background-color: #45a049; }

.sidebar .auth-buttons button.logout:hover { background-color: #e53935; }

.main-content {

margin-left: 250px; /\* Account for fixed sidebar \*/

flex-grow: 1;

padding: 30px;

background-color: #ffffff; /\* White content background \*/

transition: margin-left 0.3s ease;

min-height: 100vh;

}

/\* Style for when sidebar might be collapsed (optional) \*/

/\* body.sidebar-collapsed .sidebar { width: 80px; } \*/

/\* body.sidebar-collapsed .main-content { margin-left: 80px; } \*/

/\* --- Section Styles --- \*/

section {

display: none; /\* Hide sections by default \*/

animation: fadeIn 0.5s ease-out;

background-color: #fff;

padding: 25px;

border-radius: 8px;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.08);

margin-bottom: 25px;

}

section.active { display: block; /\* Show active section \*/ }

section h3 {

font-family: 'Poppins', sans-serif;

font-weight: 700;

color: #343a40;

margin-bottom: 20px;

padding-bottom: 10px;

border-bottom: 2px solid #e9ecef; /\* Light border under heading \*/

}

/\* --- Dashboard Specific Styles --- \*/

.dashboard-stats {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

gap: 20px;

margin-bottom: 30px;

}

.stat-card {

background-color: #f8f9fa; /\* Light background for cards \*/

padding: 20px;

border-radius: 8px;

display: flex;

align-items: center;

box-shadow: 0 1px 4px rgba(0,0,0,0.05);

transition: transform 0.2s ease, box-shadow 0.2s ease;

}

.stat-card:hover {

transform: translateY(-3px);

box-shadow: 0 4px 12px rgba(0,0,0,0.08);

}

.stat-card img { /\* Changed from i to img \*/

width: 40px; /\* Adjust size as needed \*/

height: 40px;

margin-right: 15px;

opacity: 0.7;

}

.stat-card div h4 { font-size: 1.5em; font-weight: 700; color: #4a5568; }

.stat-card div p { color: #6c757d; }

/\* --- Event Card Styles --- \*/

.event-list {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));

gap: 25px;

}

.event-card {

background-color: #fff;

border-radius: 8px;

overflow: hidden;

box-shadow: 0 4px 15px rgba(0, 0, 0, 0.1);

transition: transform 0.3s ease, box-shadow 0.3s ease;

display: flex;

flex-direction: column;

}

.event-card:hover {

transform: translateY(-5px);

box-shadow: 0 8px 25px rgba(0, 0, 0, 0.15);

}

.event-card img {

width: 100%;

height: 180px;

object-fit: cover;

display: block;

background-color: #eee; /\* Background for broken images \*/

}

.event-card-content {

padding: 20px;

flex-grow: 1; /\* Allows content to fill space \*/

display: flex;

flex-direction: column;

}

.event-card h4 {

font-family: 'Poppins', sans-serif;

font-size: 1.2em;

margin-bottom: 10px;

color: #2d3748;

}

.event-card p { font-size: 0.95em; margin-bottom: 8px; color: #4a5568; }

.event-card p strong { color: #2d3748; }

.event-card-footer {

margin-top: auto; /\* Pushes footer to bottom \*/

padding-top: 15px;

border-top: 1px solid #e9ecef;

display: flex;

justify-content: space-between;

align-items: center;

}

.event-card .btn-book, .event-card .btn-wishlist {

padding: 8px 15px;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s ease, transform 0.2s ease;

font-size: 0.9em;

font-weight: 500;

}

.event-card .btn-book { background-color: #007bff; color: white; }

.event-card .btn-book:hover { background-color: #0056b3; transform: scale(1.05); }

.event-card .btn-wishlist { background: none; border: none; font-size: 1.5em; cursor: pointer; color: #adb5bd; transition: color 0.2s ease, transform 0.2s ease; padding: 0; line-height: 1; } /\* Adjust wishlist button \*/

.event-card .btn-wishlist:hover { color: #ff4d6d; transform: scale(1.1); }

.event-card .btn-wishlist.active { color: #ff4d6d; /\* Active pink color \*/ }

/\* --- Form Styles --- \*/

.form-group { margin-bottom: 15px; }

.form-group label {

display: block;

margin-bottom: 5px;

font-weight: 700;

color: #495057;

}

.form-group input[type="text"],

.form-group input[type="date"],

.form-group input[type="time"],

.form-group input[type="number"],

.form-group input[type="email"],

.form-group input[type="password"],

.form-group textarea,

.form-group select {

width: 100%;

padding: 10px 12px;

border: 1px solid #ced4da;

border-radius: 5px;

font-size: 1em;

transition: border-color 0.2s ease, box-shadow 0.2s ease;

font-family: inherit; /\* Ensure form elements inherit body font \*/

}

.form-group input:focus,

.form-group textarea:focus,

.form-group select:focus {

border-color: #80bdff;

outline: 0;

box-shadow: 0 0 0 0.2rem rgba(0, 123, 255, 0.25);

}

.form-group textarea { resize: vertical; min-height: 100px; }

.form-actions { margin-top: 20px; text-align: right; }

.btn {

padding: 10px 20px;

border: none;

border-radius: 5px;

cursor: pointer;

font-size: 1em;

font-weight: 500;

transition: background-color 0.3s ease, transform 0.2s ease;

margin-left: 10px;

font-family: 'Poppins', sans-serif; /\* Consistent button font \*/

}

.btn-primary { background-color: #007bff; color: white; }

.btn-primary:hover { background-color: #0056b3; transform: translateY(-1px); }

.btn-secondary { background-color: #6c757d; color: white; }

.btn-secondary:hover { background-color: #5a6268; transform: translateY(-1px); }

.btn-danger { background-color: #dc3545; color: white; }

.btn-danger:hover { background-color: #c82333; transform: translateY(-1px); }

.btn-success { background-color: #28a745; color: white; }

.btn-success:hover { background-color: #218838; transform: translateY(-1px); }

.btn-sm { padding: 5px 10px; font-size: 0.9em; margin-left: 5px; } /\* Smaller buttons for tables \*/

/\* --- Table Styles --- \*/

.table-container { overflow-x: auto; /\* Responsive tables \*/ }

table {

width: 100%;

border-collapse: collapse;

margin-top: 20px;

}

th, td {

padding: 12px 15px;

text-align: left;

border-bottom: 1px solid #e9ecef;

vertical-align: middle; /\* Align content vertically \*/

}

th {

background-color: #f8f9fa;

font-weight: 700;

color: #495057;

font-family: 'Poppins', sans-serif;

}

tbody tr:hover { background-color: #f1f3f5; }

td .btn { margin-left: 0; margin-right: 5px; } /\* Adjust button margins in tables \*/

/\* --- Filter/Search Bar --- \*/

.filter-bar {

display: flex;

gap: 15px;

margin-bottom: 20px;

flex-wrap: wrap; /\* Allow wrapping on smaller screens \*/

align-items: center; /\* Align items vertically \*/

}

.filter-bar input, .filter-bar select, .filter-bar button {

padding: 8px 10px;

border: 1px solid #ced4da;

border-radius: 5px;

font-size: 0.95em;

height: 38px; /\* Consistent height \*/

}

.filter-bar input[type="text"] { flex-grow: 1; min-width: 150px; } /\* Allow search to grow \*/

.filter-bar select { min-width: 120px; }

.filter-bar button { margin-left: 0; /\* Remove default btn margin \*/ }

/\* --- Payment Section --- \*/

.payment-methods { margin-top: 20px; }

.payment-option {

background-color: #f8f9fa;

padding: 15px;

border-radius: 5px;

margin-bottom: 10px;

cursor: pointer;

border: 1px solid #e9ecef;

transition: background-color 0.2s ease, border-color 0.2s ease;

font-weight: 500;

}

.payment-option:hover { background-color: #e9ecef; }

.payment-option.active { border-color: #007bff; background-color: #e7f3ff; }

.payment-details { display: none; margin-top: 15px; padding: 20px; background-color: #f8f9fa; border: 1px solid #e9ecef; border-radius: 5px; animation: fadeIn 0.3s; }

.payment-details.active { display: block; }

/\* --- Chatbot Styles --- \*/

#chatbot {

position: fixed;

bottom: 20px;

right: 20px;

width: 350px;

max-width: 90%;

background-color: #ffffff;

border-radius: 10px;

box-shadow: 0 5px 20px rgba(0,0,0,0.15);

overflow: hidden;

z-index: 1000;

transition: all 0.3s ease-in-out;

font-size: 0.95em;

}

#chatbot.minimized {

height: 50px; /\* Height of header \*/

width: 60px;

overflow: hidden;

border-radius: 50%; /\* Make it circular when minimized \*/

padding: 0;

}

.chatbot-header {

background: linear-gradient(135deg, #4a5568, #2d3748);

color: white;

padding: 10px 15px;

display: flex;

justify-content: space-between;

align-items: center;

cursor: pointer;

height: 50px; /\* Fixed header height \*/

}

#chatbot.minimized .chatbot-header { padding: 0; justify-content: center; } /\* Center icon when minimized \*/

.chatbot-header h4 { margin: 0; font-size: 1.1em; font-family: 'Poppins', sans-serif; }

#chatbot.minimized .chatbot-header h4 { display: none; } /\* Hide title when minimized \*/

.chatbot-header button {

background: none;

border: none;

color: white;

font-size: 1.5em;

cursor: pointer;

line-height: 1;

padding: 5px;

}

#chatbot.minimized .chatbot-header button { font-size: 1.2em; } /\* Adjust icon size when minimized \*/

#chatbot-body {

height: 300px;

overflow-y: auto;

padding: 15px;

background-color: #f8f9fa; /\* Light background for chat body \*/

scroll-behavior: smooth;

}

#chatbot-body p {

margin-bottom: 12px; /\* Increased spacing \*/

line-height: 1.4;

word-wrap: break-word;

padding: 8px 12px; /\* Add padding to messages \*/

border-radius: 15px; /\* Rounded corners for messages \*/

max-width: 85%; /\* Limit message width \*/

}

#chatbot-body p strong { font-family: 'Poppins', sans-serif; display: block; margin-bottom: 3px; font-size: 0.9em; color: #555; } /\* Style sender \*/

/\* Style 'You' messages \*/

#chatbot-body p:has(strong:contains("You")) {

background-color: #007bff;

color: white;

margin-left: auto; /\* Align right \*/

border-bottom-right-radius: 5px; /\* Speech bubble effect \*/

}

#chatbot-body p:has(strong:contains("You")) strong { color: #e0e0e0; }

/\* Style 'Bot' messages \*/

#chatbot-body p:has(strong:contains("Bot")) {

background-color: #e9ecef;

color: #343a40;

margin-right: auto; /\* Align left \*/

border-bottom-left-radius: 5px; /\* Speech bubble effect \*/

}

#chatbot-input-container {

display: flex;

border-top: 1px solid #e9ecef;

}

#chatbot-input {

flex-grow: 1;

border: none;

padding: 12px 15px;

outline: none;

font-size: 1em;

font-family: 'Lato', sans-serif;

}

#chatbot-send {

background-color: #007bff;

color: white;

border: none;

padding: 0 18px;

cursor: pointer;

font-size: 1em;

transition: background-color 0.2s ease;

}

#chatbot-send:hover { background-color: #0056b3; }

#chatbot.minimized #chatbot-body,

#chatbot.minimized #chatbot-input-container { display: none; }

/\* --- Responsive Design --- \*/

@media (max-width: 992px) {

.dashboard-stats { grid-template-columns: repeat(auto-fit, minmax(180px, 1fr)); }

}

@media (max-width: 768px) {

body { flex-direction: column; } /\* Stack sidebar and content \*/

.sidebar {

width: 100%;

height: auto;

position: relative; /\* Change from fixed \*/

box-shadow: none;

flex-direction: row; /\* Arrange items horizontally \*/

align-items: center; /\* Center items vertically \*/

padding: 10px 15px;

z-index: 100; /\* Keep above content \*/

}

.sidebar h2 { margin-bottom: 0; margin-right: auto; font-size: 1.2em; } /\* Adjust heading \*/

.sidebar nav { display: none; } /\* Hide nav links in simple horizontal bar \*/

/\* TODO: Add a hamburger menu toggle for mobile nav \*/

.sidebar .auth-buttons { margin-top: 0; margin-left: auto; display: flex; gap: 10px; } /\* Move auth buttons right \*/

.sidebar .auth-buttons button { width: auto; padding: 8px 12px; margin-bottom: 0; }

.main-content {

margin-left: 0; /\* No margin needed \*/

padding: 20px;

width: 100%;

}

.event-list { grid-template-columns: 1fr; } /\* Single column on smaller screens \*/

.filter-bar { flex-direction: column; gap: 10px; align-items: stretch; }

#chatbot { width: 90%; max-width: 300px; bottom: 10px; right: 10px; }

#chatbot.minimized { width: 50px; height: 50px; } /\* Slightly smaller minimized button \*/

}

@media (max-width: 480px) {

.dashboard-stats { grid-template-columns: 1fr; } /\* Single column stats \*/

.btn { /\* Full width buttons with exceptions \*/

width: 100%;

margin-left: 0;

margin-bottom: 10px;

}

.btn-sm, .filter-bar button, .event-card-footer .btn, .sidebar .auth-buttons button {

width: auto; /\* Keep small/inline buttons auto width \*/

margin-bottom: 0;

}

.form-actions { text-align: center; }

.form-actions .btn { margin-bottom: 10px; } /\* Ensure spacing in form actions \*/

.form-actions .btn:last-child { margin-bottom: 0; }

}

/\* Utility Classes \*/

.hidden { display: none !important; } /\* Use important to override \*/

.text-center { text-align: center; }

.mt-2 { margin-top: 0.5rem; }

.mb-2 { margin-bottom: 0.5rem; }

.p-2 { padding: 0.5rem; }

.text-danger { color: #dc3545; }

.text-success { color: #28a745; }

.text-info { color: #17a2b8; }

</style>

</head>

<body>

<aside class="sidebar">

<h2>EventPortal</h2>

<nav>

<ul>

<li><a href="#" class="active" onclick="showSection('dashboard', this)">

<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24"><path d="M3 13h8V3H3v10zm0 8h8v-6H3v6zm10 0h8V11h-8v10zm0-18v6h8V3h-8z"/></svg>

Dashboard

</a></li>

<li><a href="#" onclick="showSection('events', this)">

<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24"><path d="M19 3h-1V1h-2v2H8V1H6v2H5c-1.11 0-1.99.9-1.99 2L3 19c0 1.1.89 2 2 2h14c1.1 0 2-.9 2-2V5c0-1.1-.9-2-2-2zm0 16H5V8h14v11zM7 10h5v5H7z"/></svg>

Events

</a></li>

<li><a href="#" onclick="showSection('booking', this)">

<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24"><path d="M20 6h-4V4c0-1.11-.89-2-2-2h-4c-1.11 0-2 .89-2 2v2H4c-1.11 0-1.99.89-1.99 2L2 19c0 1.11.89 2 2 2h16c1.11 0 2-.89 2-2V8c0-1.11-.89-2-2-2zm-6 0h-4V4h4v2z"/></svg>

Booking

</a></li>

<li><a href="#" onclick="showSection('attendees', this)">

<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24"><path d="M16 11c1.66 0 2.99-1.34 2.99-3S17.66 5 16 5c-1.66 0-3 1.34-3 3s1.34 3 3 3zm-8 0c1.66 0 2.99-1.34 2.99-3S9.66 5 8 5C6.34 5 5 6.34 5 8s1.34 3 3 3zm0 2c-2.33 0-7 1.17-7 3.5V19h14v-2.5c0-2.33-4.67-3.5-7-3.5zm8 0c-.29 0-.62.02-.97.05 1.16.84 1.97 1.97 1.97 3.45V19h6v-2.5c0-2.33-4.67-3.5-7-3.5z"/></svg>

Attendees

</a></li>

<li><a href="#" onclick="showSection('wishlist', this)">

<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24"><path d="M12 21.35l-1.45-1.32C5.4 15.36 2 12.28 2 8.5 2 5.42 4.42 3 7.5 3c1.74 0 3.41.81 4.5 2.09C13.09 3.81 14.76 3 16.5 3 19.58 3 22 5.42 22 8.5c0 3.78-3.4 6.86-8.55 11.54L12 21.35z"/></svg>

Wishlist

</a></li>

<li><a href="#" onclick="showSection('sponsors', this)">

<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24"><path d="M12 2C6.48 2 2 6.48 2 12s4.48 10 10 10 10-4.48 10-10S17.52 2 12 2zm0 18c-4.41 0-8-3.59-8-8s3.59-8 8-8 8 3.59 8 8-3.59 8-8 8zm-5.5-2.5l7-4.5-7-4.5v9z"/></svg> Sponsors

</a></li>

<li><a href="#" onclick="showSection('analytics', this)">

<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24"><path d="M19 3H5c-1.1 0-2 .9-2 2v14c0 1.1.9 2 2 2h14c1.1 0 2-.9 2-2V5c0-1.1-.9-2-2-2zM9 17H7v-7h2v7zm4 0h-2V7h2v10zm4 0h-2v-4h2v4z"/></svg>

Analytics

</a></li>

</ul>

</nav>

<div class="auth-buttons">

<button id="loginBtn" onclick="showSection('auth', this); showAuthForm('login');">Login</button>

<button id="registerBtn" onclick="showSection('auth', this); showAuthForm('register');">Register</button>

<button id="logoutBtn" class="hidden logout" onclick="logoutUser()">Logout</button>

</div>

</aside>

<main class="main-content">

<section id="dashboard" class="active">

<h3>Dashboard Overview</h3>

<div class="dashboard-stats">

<div class="stat-card">

<img src="https://placehold.co/40x40/007bff/ffffff?text=E" alt="Events Icon"> <div>

<h4 id="total-events">0</h4>

<p>Total Events</p>

</div>

</div>

<div class="stat-card">

<img src="https://placehold.co/40x40/28a745/ffffff?text=A" alt="Attendees Icon"> <div>

<h4 id="total-attendees">0</h4>

<p>Total Attendees</p>

</div>

</div>

<div class="stat-card">

<img src="https://placehold.co/40x40/ffc107/ffffff?text=S" alt="Sponsors Icon"> <div>

<h4 id="total-sponsors">0</h4>

<p>Sponsors</p>

</div>

</div>

<div class="stat-card">

<img src="https://placehold.co/40x40/17a2b8/ffffff?text=U" alt="Users Icon"> <div>

<h4 id="total-users">0</h4>

<p>Registered Users</p>

</div>

</div>

</div>

<p>Welcome to the Event Management Portal. Use the sidebar to navigate through different sections.</p>

</section>

<section id="events">

<h3>Manage Events</h3>

<button class="btn btn-primary mb-2" onclick="toggleCreateEventForm()">Create New Event</button>

<div id="create-event-form" class="hidden" style="background-color: #f8f9fa; padding: 20px; border-radius: 8px; margin-bottom: 20px; border: 1px solid #eee;">

<h4 id="form-title">Create New Event</h4>

<form id="event-form" onsubmit="saveEvent(event)">

<input type="hidden" id="event-id"> <div class="form-group">

<label for="event-name">Event Name</label>

<input type="text" id="event-name" required>

</div>

<div class="form-group">

<label for="event-description">Description</label>

<textarea id="event-description" required></textarea>

</div>

<div class="form-group">

<label for="event-date">Date</label>

<input type="date" id="event-date" required>

</div>

<div class="form-group">

<label for="event-time">Time</label>

<input type="time" id="event-time" required>

</div>

<div class="form-group">

<label for="event-location">Location</label>

<input type="text" id="event-location" required>

</div>

<div class="form-group">

<label for="event-category">Category</label>

<select id="event-category">

<option value="Conference">Conference</option>

<option value="Workshop">Workshop</option>

<option value="Meetup">Meetup</option>

<option value="Webinar">Webinar</option>

<option value="Social">Social</option>

</select>

</div>

<div class="form-group">

<label for="event-price">Ticket Price ($)</label>

<input type="number" id="event-price" min="0" step="0.01" value="0" required>

</div>

<div class="form-group">

<label for="event-image-url">Image URL</label>

<input type="text" id="event-image-url" placeholder="https://example.com/image.jpg">

<small>Use a valid web URL or leave blank for default.</small>

</div>

<div class="form-actions">

<button type="submit" class="btn btn-success">Save Event</button>

<button type="button" class="btn btn-secondary" onclick="toggleCreateEventForm(true)">Cancel</button>

</div>

</form>

</div>

<div class="filter-bar">

<input type="text" id="event-search" placeholder="Search events..." oninput="filterEvents()">

<select id="event-category-filter" onchange="filterEvents()">

<option value="">All Categories</option>

<option value="Conference">Conference</option>

<option value="Workshop">Workshop</option>

<option value="Meetup">Meetup</option>

<option value="Webinar">Webinar</option>

<option value="Social">Social</option>

</select>

<input type="date" id="event-date-filter" onchange="filterEvents()">

<button class="btn btn-secondary" onclick="clearEventFilters()">Clear Filters</button>

</div>

<div id="event-list" class="event-list">

<p>Loading events...</p>

</div>

</section>

<section id="booking">

<h3>Event Booking</h3>

<div id="booking-details" class="hidden">

<h4>Book Tickets for: <span id="booking-event-name">Event Name</span></h4>

<p>Date: <span id="booking-event-date">Date</span> | Time: <span id="booking-event-time">Time</span> | Location: <span id="booking-event-location">Location</span></p>

<p>Price per ticket: $<span id="booking-ticket-price">0.00</span></p>

<form id="booking-form" onsubmit="processBooking(event)">

<input type="hidden" id="booking-event-id">

<div class="form-group">

<label for="num-tickets">Number of Tickets</label>

<input type="number" id="num-tickets" min="1" value="1" required oninput="updateTotalCost()">

</div>

<div class="form-group">

<label for="attendee-name">Your Name</label>

<input type="text" id="attendee-name" required>

</div>

<div class="form-group">

<label for="attendee-email">Your Email</label>

<input type="email" id="attendee-email" required>

</div>

<h4>Total Cost: $<span id="total-cost">0.00</span></h4>

<div class="payment-methods">

<h4>Select Payment Method (Simulation)</h4>

<div class="payment-option" onclick="selectPaymentMethod('card', this)">Credit/Debit Card</div>

<div id="card-details" class="payment-details">

<div class="form-group">

<label for="card-number">Card Number</label>

<input type="text" id="card-number" placeholder="\*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\*">

</div>

<div class="form-group" style="display: flex; gap: 10px;">

<div style="flex: 1;">

<label for="expiry-date">Expiry (MM/YY)</label>

<input type="text" id="expiry-date" placeholder="MM/YY">

</div>

<div style="flex: 1;">

<label for="cvv">CVV</label>

<input type="text" id="cvv" placeholder="\*\*\*">

</div>

</div>

<button type="button" class="btn btn-primary" onclick="processCardPayment()">Pay with Card</button>

</div>

<div class="payment-option" onclick="selectPaymentMethod('paypal', this)">PayPal</div>

<div id="paypal-details" class="payment-details">

<p>You will be redirected to PayPal (Simulation).</p>

<button type="button" class="btn btn-primary" onclick="processPayPalPayment()">Pay with PayPal</button>

</div>

<div class="payment-option" onclick="selectPaymentMethod('free', this)">Free Event / Pay Later</div>

<div id="free-details" class="payment-details">

<p>Confirm your booking for this free event or choose to pay later if applicable.</p>

<button type="button" class="btn btn-primary" onclick="processFreeOrPayLater()">Confirm Booking</button>

</div>

</div>

<p id="booking-message" class="mt-2"></p>

</form>

</div>

<div id="no-booking-selected">

<p>Select an event from the 'Events' section and click 'Book Now' to start the booking process.</p>

</div>

</section>

<section id="attendees">

<h3>Manage Attendees</h3>

<div class="filter-bar">

<select id="attendee-event-filter" onchange="filterAttendees()">

<option value="">Filter by Event</option>

</select>

<input type="text" id="attendee-search" placeholder="Search attendees by name or email..." oninput="filterAttendees()">

</div>

<div class="table-container">

<table id="attendees-table">

<thead>

<tr>

<th>Name</th>

<th>Email</th>

<th>Event</th>

<th>Tickets</th>

<th>Checked In</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

<tr><td colspan="6">Loading attendees...</td></tr>

</tbody>

</table>

</div>

</section>

<section id="wishlist">

<h3>My Wishlist & Reminders</h3>

<div id="wishlist-content">

<p>Your wishlist is empty. Add events using the ♡ icon.</p>

</div>

</section>

<section id="sponsors">

<h3>Manage Sponsors</h3>

<button class="btn btn-primary mb-2" onclick="showSponsorForm()">Add New Sponsor</button>

<div id="sponsor-form-container" class="hidden" style="background-color: #f8f9fa; padding: 20px; border-radius: 8px; margin-bottom: 20px; border: 1px solid #eee;">

<h4 id="sponsor-form-title">Add New Sponsor</h4>

<form id="sponsor-form" onsubmit="addOrUpdateSponsor(event)">

<input type="hidden" id="sponsor-id">

<div class="form-group">

<label for="sponsor-name">Sponsor Name</label>

<input type="text" id="sponsor-name" required>

</div>

<div class="form-group">

<label for="sponsor-level">Sponsorship Level</label>

<select id="sponsor-level">

<option value="Platinum">Platinum</option>

<option value="Gold">Gold</option>

<option value="Silver">Silver</option>

<option value="Bronze">Bronze</option>

</select>

</div>

<div class="form-group">

<label for="sponsor-contact">Contact Email</label>

<input type="email" id="sponsor-contact">

</div>

<div class="form-group">

<label for="sponsor-logo-url">Logo URL</label>

<input type="text" id="sponsor-logo-url" placeholder="https://example.com/logo.png">

</div>

<div class="form-actions">

<button type="submit" class="btn btn-success">Save Sponsor</button>

<button type="button" class="btn btn-secondary" onclick="hideSponsorForm()">Cancel</button>

</div>

</form>

</div>

<div class="table-container">

<table id="sponsors-table">

<thead>

<tr>

<th>Name</th>

<th>Level</th>

<th>Contact</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

<tr><td colspan="4">Loading sponsors...</td></tr>

</tbody>

</table>

</div>

</section>

<section id="analytics">

<h3>Analytics & Reports</h3>

<p>Generate reports on event performance, attendance, and revenue.</p>

<button class="btn btn-primary" onclick="generateAnalytics()">Generate Report (Simulation)</button>

<div id="analytics-output" class="mt-2" style="background-color: #e9ecef; padding: 15px; border-radius: 5px; min-height: 100px;">

Report data will be shown here after generation.

</div>

</section>

<section id="auth">

<h3>User Authentication</h3>

<div style="max-width: 400px; margin: 20px auto; padding: 25px; background-color: #f8f9fa; border-radius: 8px; box-shadow: 0 2px 8px rgba(0,0,0,0.1);">

<img src="https://placehold.co/60x60/007bff/ffffff?text=AUTH" alt="Auth Icon" style="display: block; margin: 0 auto 15px;"> <form id="login-form" onsubmit="loginUser(event)">

<h4>Login</h4>

<div class="form-group">

<label for="login-email">Email</label>

<input type="email" id="login-email" required>

</div>

<div class="form-group">

<label for="login-password">Password</label>

<input type="password" id="login-password" required>

</div>

<button type="submit" class="btn btn-primary" style="width: 100%;">Login</button>

<p class="text-center mt-2">Don't have an account? <a href="#" onclick="showAuthForm('register'); return false;">Register here</a></p>

<p id="login-message" class="text-danger text-center mt-2"></p>

</form>

<form id="register-form" class="hidden" onsubmit="registerUser(event)">

<h4>Register</h4>

<div class="form-group">

<label for="register-name">Full Name</label>

<input type="text" id="register-name" required>

</div>

<div class="form-group">

<label for="register-email">Email</label>

<input type="email" id="register-email" required>

</div>

<div class="form-group">

<label for="register-password">Password</label>

<input type="password" id="register-password" required>

</div>

<button type="submit" class="btn btn-success" style="width: 100%;">Register</button>

<p class="text-center mt-2">Already have an account? <a href="#" onclick="showAuthForm('login'); return false;">Login here</a></p>

<p id="register-message" class="text-success text-center mt-2"></p>

</form>

</div>

</section>

</main>

<div id="chatbot">

<div class="chatbot-header" onclick="minimizeChatbot()">

<h4>Chat Assistant</h4>

<button onclick="minimizeChatbot(event)">\_</button> </div>

<div id="chatbot-body">

<p><strong>Bot:</strong> Hello! How can I help you with the event portal today?</p>

</div>

<div id="chatbot-input-container">

<input type="text" id="chatbot-input" placeholder="Ask a question..." onkeydown="if(event.key === 'Enter') sendChatMessage()">

<button id="chatbot-send" onclick="sendChatMessage()">Send</button>

</div>

</div>

<script>

// --- Global Variables & Initial State ---

let currentSection = 'dashboard'; // Default section

let loggedInUser = null; // Track logged-in user state

let events = []; // Array to hold event data (simulated)

let attendees = []; // Array to hold attendee data (simulated)

let sponsors = []; // Array to hold sponsor data (simulated)

let wishlist = []; // Array to hold wishlist event IDs

let lastContext = { type: null, data: null }; // For chatbot context

// --- Utility Functions ---

/\*\*

\* Shows the specified section and hides others. Updates sidebar active state.

\* @param {string} sectionId - The ID of the section to show.

\* @param {HTMLElement} [linkElement] - The sidebar link element that was clicked.

\*/

function showSection(sectionId, linkElement) {

console.log(`Navigating to section: ${sectionId}`);

// Hide all sections

document.querySelectorAll('main section').forEach(section => {

section.classList.remove('active');

});

// Show the target section

const targetSection = document.getElementById(sectionId);

if (targetSection) {

targetSection.classList.add('active');

currentSection = sectionId;

} else {

console.error(`Section with ID "${sectionId}" not found.`);

document.getElementById('dashboard').classList.add('active'); // Fallback to dashboard

currentSection = 'dashboard';

}

// Update active link in sidebar

document.querySelectorAll('.sidebar nav ul li a').forEach(link => {

link.classList.remove('active');

});

if (linkElement) {

linkElement.classList.add('active');

} else {

// If called without a link (e.g., on login), find the link corresponding to the section

const correspondingLink = document.querySelector(`.sidebar nav ul li a[onclick\*="'${sectionId}'"]`);

if (correspondingLink) {

correspondingLink.classList.add('active');

} else {

// Fallback for sections without direct nav links (like auth) or if link not found

const dashboardLink = document.querySelector(`.sidebar nav ul li a[onclick\*="'dashboard'"]`);

if (dashboardLink) dashboardLink.classList.add('active');

}

}

// Special handling for booking section visibility

const bookingDetails = document.getElementById('booking-details');

const noBookingSelected = document.getElementById('no-booking-selected');

if (sectionId === 'booking') {

// Check if a booking is in progress (check if event ID has value)

if (document.getElementById('booking-event-id').value) {

bookingDetails.classList.remove('hidden');

noBookingSelected.classList.add('hidden');

} else {

bookingDetails.classList.add('hidden');

noBookingSelected.classList.remove('hidden');

}

} else {

// Ensure booking details are hidden if navigating away from booking section without completing

if (bookingDetails) bookingDetails.classList.add('hidden');

if (noBookingSelected) noBookingSelected.classList.remove('hidden'); // Show default message

}

// Refresh data when navigating to certain sections

if (sectionId === 'events') loadEvents();

if (sectionId === 'attendees') loadAttendees();

if (sectionId === 'sponsors') loadSponsors();

if (sectionId === 'wishlist') loadWishlist();

if (sectionId === 'dashboard') updateDashboardStats(); // Update stats when returning to dashboard

}

/\*\*

\* Toggles the visibility of the create/edit event form.

\* @param {boolean} [forceHide=false] - If true, always hides the form.

\*/

function toggleCreateEventForm(forceHide = false) {

const formDiv = document.getElementById('create-event-form');

const formTitle = document.getElementById('form-title');

const eventForm = document.getElementById('event-form');

const eventIdInput = document.getElementById('event-id');

if (forceHide || !formDiv.classList.contains('hidden')) {

formDiv.classList.add('hidden');

console.log('Hiding create/edit event form.');

} else {

formDiv.classList.remove('hidden');

// Reset form for 'Create' unless an ID is present (meaning 'Edit')

if (!eventIdInput.value) {

formTitle.textContent = 'Create New Event';

eventForm.reset();

eventIdInput.value = ''; // Ensure ID is cleared

console.log('Showing create event form.');

} else {

// Keep existing values if editing

console.log('Showing edit event form for ID:', eventIdInput.value);

}

}

}

/\*\*

\* Shows the specified authentication form (login or register).

\* @param {'login' | 'register'} formToShow - The form to display.

\*/

function showAuthForm(formToShow) {

console.log(`Showing auth form: ${formToShow}`);

const loginForm = document.getElementById('login-form');

const registerForm = document.getElementById('register-form');

const loginMessage = document.getElementById('login-message');

const registerMessage = document.getElementById('register-message');

loginMessage.textContent = ''; // Clear messages

registerMessage.textContent = '';

if (formToShow === 'login') {

loginForm.classList.remove('hidden');

registerForm.classList.add('hidden');

} else {

loginForm.classList.add('hidden');

registerForm.classList.remove('hidden');

}

}

/\*\*

\* Updates the total cost display in the booking form.

\*/

function updateTotalCost() {

const numTicketsInput = document.getElementById('num-tickets');

const pricePerTicketSpan = document.getElementById('booking-ticket-price');

const totalCostSpan = document.getElementById('total-cost');

const numTickets = parseInt(numTicketsInput.value) || 0;

const pricePerTicket = parseFloat(pricePerTicketSpan.textContent) || 0;

// Ensure non-negative tickets

if (numTickets < 0) {

numTicketsInput.value = 0;

totalCostSpan.textContent = '0.00';

return;

}

const totalCost = (numTickets \* pricePerTicket).toFixed(2);

totalCostSpan.textContent = totalCost;

console.log(`Updated total cost: $${totalCost}`);

}

/\*\*

\* Handles selection of payment methods, showing relevant details fields.

\* @param {string} method - The selected payment method ('card', 'paypal', 'free').

\* @param {HTMLElement} element - The clicked payment option element.

\*/

function selectPaymentMethod(method, element) {

console.log(`Payment method selected: ${method}`);

// Remove active class from all options and hide all details

document.querySelectorAll('.payment-option').forEach(opt => opt.classList.remove('active'));

document.querySelectorAll('.payment-details').forEach(det => det.classList.remove('active'));

// Add active class to the selected option and show its details

element.classList.add('active');

const detailsDiv = document.getElementById(`${method}-details`);

if (detailsDiv) {

detailsDiv.classList.add('active');

}

}

/\*\*

\* Clears filters in the Events section and reloads events.

\*/

function clearEventFilters() {

console.log('Clearing event filters.');

document.getElementById('event-search').value = '';

document.getElementById('event-category-filter').value = '';

document.getElementById('event-date-filter').value = '';

filterEvents(); // Re-apply filters (which should now show all)

}

/\*\*

\* Shows the add/edit sponsor form.

\* @param {object} [sponsorData=null] - Optional sponsor data for editing.

\*/

function showSponsorForm(sponsorData = null) {

const formContainer = document.getElementById('sponsor-form-container');

const formTitle = document.getElementById('sponsor-form-title');

const sponsorForm = document.getElementById('sponsor-form');

const sponsorIdInput = document.getElementById('sponsor-id');

sponsorForm.reset(); // Clear previous data

sponsorIdInput.value = ''; // Clear ID

if (sponsorData) {

console.log('Populating sponsor form for editing:', sponsorData);

formTitle.textContent = 'Edit Sponsor';

sponsorIdInput.value = sponsorData.id;

document.getElementById('sponsor-name').value = sponsorData.name;

document.getElementById('sponsor-level').value = sponsorData.level;

document.getElementById('sponsor-contact').value = sponsorData.contact || '';

document.getElementById('sponsor-logo-url').value = sponsorData.logoUrl || '';

} else {

console.log('Showing add sponsor form.');

formTitle.textContent = 'Add New Sponsor';

}

formContainer.classList.remove('hidden');

}

/\*\*

\* Hides the add/edit sponsor form.

\*/

function hideSponsorForm() {

console.log('Hiding sponsor form.');

document.getElementById('sponsor-form-container').classList.add('hidden');

document.getElementById('sponsor-form').reset();

document.getElementById('sponsor-id').value = '';

}

// --- Placeholder Data Loading & Display Functions ---

/\*\*

\* Loads and displays sample events (Placeholder). Uses localStorage.

\*/

function loadEvents() {

console.log("Loading events (placeholder)...");

// Simulate fetching events (replace with actual API call/localStorage retrieval)

const defaultEvents = [

{ id: 'evt1', name: 'Tech Conference 2025', description: 'The annual gathering of tech enthusiasts and professionals. Explore the future of technology.', date: '2025-10-15', time: '09:00', location: 'Grand Convention Center', category: 'Conference', price: 100, imageUrl: 'https://placehold.co/600x360/007bff/ffffff?text=Tech+Conf' },

{ id: 'evt2', name: 'Web Development Workshop', description: 'A hands-on workshop covering the latest in frontend and backend development.', date: '2025-11-05', time: '10:00', location: 'Online', category: 'Workshop', price: 50, imageUrl: 'https://placehold.co/600x360/28a745/ffffff?text=Web+Workshop' },

{ id: 'evt3', name: 'Startup Networking Meetup', description: 'Connect with fellow entrepreneurs, investors, and innovators in the local startup scene.', date: '2025-11-20', time: '18:30', location: 'Community Hub Room A', category: 'Meetup', price: 0, imageUrl: 'https://placehold.co/600x360/ffc107/000000?text=Meetup' },

{ id: 'evt4', name: 'Digital Marketing Trends Webinar', description: 'Learn about the latest strategies and trends shaping the digital marketing landscape.', date: '2025-12-01', time: '14:00', location: 'Online', category: 'Webinar', price: 25, imageUrl: 'https://placehold.co/600x360/17a2b8/ffffff?text=Webinar' }

];

// Get events from localStorage or use defaults if none exist

events = JSON.parse(localStorage.getItem('events')) || defaultEvents;

// If localStorage was empty, save the defaults

if (!localStorage.getItem('events')) {

localStorage.setItem('events', JSON.stringify(events));

}

filterEvents(); // Apply any existing filters initially

populateAttendeeEventFilter(); // Update filter dropdown

}

/\*\*

\* Renders the list of events based on the provided data.

\* @param {Array} eventsToRender - Array of event objects.

\*/

function renderEventList(eventsToRender) {

const eventListDiv = document.getElementById('event-list');

eventListDiv.innerHTML = ''; // Clear previous list

if (!eventsToRender || eventsToRender.length === 0) {

eventListDiv.innerHTML = '<p>No events found matching your criteria.</p>';

return;

}

// Load wishlist from localStorage to check active status

const currentWishlist = JSON.parse(localStorage.getItem('wishlist')) || [];

eventsToRender.forEach(event => {

const card = document.createElement('div');

card.className = 'event-card';

card.dataset.eventId = event.id; // Store ID for easy access

const isWishlisted = currentWishlist.includes(event.id);

const wishlistClass = isWishlisted ? 'active' : '';

const wishlistIcon = isWishlisted ? '❤️' : '♡'; // Use filled/empty heart

card.innerHTML = `

<img src="${event.imageUrl || 'https://placehold.co/600x360/6c757d/ffffff?text=Event'}" alt="${event.name}" onerror="this.onerror=null; this.src='https://placehold.co/600x360/6c757d/ffffff?text=Image+Error';">

<div class="event-card-content">

<h4>${event.name}</h4>

<p>${event.description}</p>

<p><strong>Date:</strong> ${event.date}</p>

<p><strong>Time:</strong> ${event.time}</p>

<p><strong>Location:</strong> ${event.location}</p>

<p><strong>Category:</strong> ${event.category}</p>

<p><strong>Price:</strong> $${event.price.toFixed(2)}</p>

<div class="event-card-footer">

<button class="btn btn-sm btn-book" onclick="startBooking('${event.id}')">Book Now</button>

<button class="btn-wishlist ${wishlistClass}" onclick="toggleWishlist('${event.id}', this)">${wishlistIcon}</button>

</div>

</div>

`;

eventListDiv.appendChild(card);

});

console.log(`Rendered ${eventsToRender.length} events.`);

}

/\*\*

\* Filters the displayed events based on search, category, and date.

\*/

function filterEvents() {

const searchTerm = document.getElementById('event-search').value.toLowerCase();

const categoryFilter = document.getElementById('event-category-filter').value;

const dateFilter = document.getElementById('event-date-filter').value;

console.log(`Filtering events: Search='${searchTerm}', Category='${categoryFilter}', Date='${dateFilter}'`);

const filteredEvents = events.filter(event => {

const nameMatch = event.name.toLowerCase().includes(searchTerm);

const descriptionMatch = event.description.toLowerCase().includes(searchTerm);

const locationMatch = event.location.toLowerCase().includes(searchTerm);

const categoryMatch = !categoryFilter || event.category === categoryFilter;

const dateMatch = !dateFilter || event.date === dateFilter;

return (nameMatch || descriptionMatch || locationMatch) && categoryMatch && dateMatch;

});

renderEventList(filteredEvents);

}

/\*\*

\* Loads and displays sample attendees (Placeholder). Uses localStorage.

\*/

function loadAttendees() {

console.log("Loading attendees (placeholder)...");

// Simulate fetching attendees

const defaultAttendees = [

{ id: 'att1', name: 'Alice Smith', email: 'alice@example.com', eventId: 'evt1', eventName: 'Tech Conference 2025', tickets: 1, checkedIn: false },

{ id: 'att2', name: 'Bob Johnson', email: 'bob@example.com', eventId: 'evt2', eventName: 'Web Development Workshop', tickets: 2, checkedIn: true },

{ id: 'att3', name: 'Charlie Brown', email: 'charlie@example.com', eventId: 'evt1', eventName: 'Tech Conference 2025', tickets: 1, checkedIn: false }

];

attendees = JSON.parse(localStorage.getItem('attendees')) || defaultAttendees;

if (!localStorage.getItem('attendees')) {

localStorage.setItem('attendees', JSON.stringify(attendees));

}

filterAttendees(); // Apply filters

}

/\*\*

\* Renders the list of attendees based on the provided data.

\* @param {Array} attendeesToRender - Array of attendee objects.

\*/

function renderAttendeeList(attendeesToRender) {

const tableBody = document.getElementById('attendees-table').querySelector('tbody');

tableBody.innerHTML = ''; // Clear previous list

if (!attendeesToRender || attendeesToRender.length === 0) {

tableBody.innerHTML = '<tr><td colspan="6">No attendees found matching your criteria.</td></tr>';

return;

}

attendeesToRender.forEach(attendee => {

const row = tableBody.insertRow();

row.dataset.attendeeId = attendee.id; // Store ID

row.innerHTML = `

<td>${attendee.name}</td>

<td>${attendee.email}</td>

<td>${attendee.eventName}</td>

<td>${attendee.tickets}</td>

<td>${attendee.checkedIn ? '<span class="text-success">Yes</span>' : 'No'}</td>

<td>

<button class="btn btn-sm ${attendee.checkedIn ? 'btn-secondary' : 'btn-success'}" onclick="checkInAttendee('${attendee.id}', ${!attendee.checkedIn})">

${attendee.checkedIn ? 'Undo Check-In' : 'Check-In'}

</button>

<button class="btn btn-sm btn-danger" onclick="removeAttendee('${attendee.id}')">Remove</button>

</td>

`;

});

console.log(`Rendered ${attendeesToRender.length} attendees.`);

}

/\*\*

\* Populates the event filter dropdown in the Attendees section.

\*/

function populateAttendeeEventFilter() {

const filterSelect = document.getElementById('attendee-event-filter');

const currentFilterValue = filterSelect.value; // Preserve selection if possible

// Preserve the "Filter by Event" option

filterSelect.innerHTML = '<option value="">Filter by Event</option>';

// Get unique event names and IDs from the loaded events

const eventOptions = [...new Map(events.map(event => [event.id, {id: event.id, name: event.name}])).values()];

eventOptions.sort((a, b) => a.name.localeCompare(b.name)); // Sort alphabetically

eventOptions.forEach(eventInfo => {

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

option.value = eventInfo.id; // Use event ID as value

option.textContent = eventInfo.name;

if (eventInfo.id === currentFilterValue) {

option.selected = true; // Restore selection

}

filterSelect.appendChild(option);

});

console.log('Populated attendee event filter dropdown.');

}

/\*\*

\* Filters the displayed attendees based on event and search term.

\*/

function filterAttendees() {

const eventFilter = document.getElementById('attendee-event-filter').value;

const searchTerm = document.getElementById('attendee-search').value.toLowerCase();

console.log(`Filtering attendees: EventID='${eventFilter}', Search='${searchTerm}'`);

const filteredAttendees = attendees.filter(attendee => {

const eventMatch = !eventFilter || attendee.eventId === eventFilter;

const nameMatch = attendee.name.toLowerCase().includes(searchTerm);

const emailMatch = attendee.email.toLowerCase().includes(searchTerm);

return eventMatch && (nameMatch || emailMatch);

});

renderAttendeeList(filteredAttendees);

}

/\*\*

\* Loads and displays sample sponsors (Placeholder). Uses localStorage.

\*/

function loadSponsors() {

console.log("Loading sponsors (placeholder)...");

// Simulate fetching sponsors

const defaultSponsors = [

{ id: 'spn1', name: 'Gold Sponsor Inc.', level: 'Gold', contact: 'contact@goldsponsor.com', logoUrl: 'https://placehold.co/100x50/ffd700/000000?text=Gold' },

{ id: 'spn2', name: 'Silver Partner Co.', level: 'Silver', contact: 'info@silverpartner.co', logoUrl: 'https://placehold.co/100x50/c0c0c0/000000?text=Silver' }

];

sponsors = JSON.parse(localStorage.getItem('sponsors')) || defaultSponsors;

if (!localStorage.getItem('sponsors')) {

localStorage.setItem('sponsors', JSON.stringify(sponsors));

}

renderSponsorList(sponsors);

}

/\*\*

\* Renders the list of sponsors based on the provided data.

\* @param {Array} sponsorsToRender - Array of sponsor objects.

\*/

function renderSponsorList(sponsorsToRender) {

const tableBody = document.getElementById('sponsors-table').querySelector('tbody');

tableBody.innerHTML = ''; // Clear previous list

if (!sponsorsToRender || sponsorsToRender.length === 0) {

tableBody.innerHTML = '<tr><td colspan="4">No sponsors found.</td></tr>';

return;

}

sponsorsToRender.forEach(sponsor => {

const row = tableBody.insertRow();

row.dataset.sponsorId = sponsor.id; // Store ID

row.innerHTML = `

<td>${sponsor.name} ${sponsor.logoUrl ? `<img src="${sponsor.logoUrl}" alt="${sponsor.name} Logo" style="height: 20px; vertical-align: middle; margin-left: 5px;" onerror="this.style.display='none'">` : ''}</td>

<td>${sponsor.level}</td>

<td>${sponsor.contact || 'N/A'}</td>

<td>

<button class="btn btn-sm btn-secondary" onclick="editSponsor('${sponsor.id}')">Edit</button>

<button class="btn btn-sm btn-danger" onclick="removeSponsor('${sponsor.id}')">Remove</button>

</td>

`;

});

console.log(`Rendered ${sponsorsToRender.length} sponsors.`);

}

/\*\*

\* Loads and displays the user's wishlist (Placeholder). Uses localStorage.

\*/

function loadWishlist() {

console.log("Loading wishlist (placeholder)...");

wishlist = JSON.parse(localStorage.getItem('wishlist')) || [];

const wishlistContent = document.getElementById('wishlist-content');

wishlistContent.innerHTML = ''; // Clear previous content

if (wishlist.length === 0) {

wishlistContent.innerHTML = '<p>Your wishlist is empty. Add events using the ♡ icon on the event cards.</p>';

return;

}

wishlistContent.innerHTML = '<h4>Your Wishlisted Events:</h4>';

const list = document.createElement('ul');

list.style.listStyle = 'none'; // Remove default bullets

list.style.padding = '0';

let itemsFound = 0;

wishlist.forEach(eventId => {

const event = events.find(e => e.id === eventId); // Find event details in the currently loaded events array

if (event) {

itemsFound++;

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

listItem.style.marginBottom = '15px';

listItem.style.padding = '10px';

listItem.style.border = '1px solid #eee';

listItem.style.borderRadius = '5px';

listItem.style.display = 'flex'; // Use flex for alignment

listItem.style.justifyContent = 'space-between';

listItem.style.alignItems = 'center';

listItem.innerHTML = `

<div>

<strong>${event.name}</strong> (${event.date})<br>

<small>${event.location}</small>

</div>

<div>

<button class="btn btn-sm btn-primary" style="margin-left: 10px;" onclick="setReminder('${eventId}')">Set Reminder (Sim)</button>

<button class="btn btn-sm btn-danger" style="margin-left: 10px;" onclick="removeFromWishlist('${eventId}')">Remove</button>

</div>

`;

list.appendChild(listItem);

} else {

console.warn(`Event ID ${eventId} found in wishlist but not in loaded events.`);

// Optionally remove orphaned wishlist items here by modifying the wishlist array and saving back to localStorage

}

});

if (itemsFound === 0) {

wishlistContent.innerHTML = '<p>Your wishlist is empty or contains events that are no longer available.</p>';

} else {

wishlistContent.appendChild(list);

}

}

/\*\*

\* Updates the statistics shown on the dashboard. Uses localStorage.

\*/

function updateDashboardStats() {

console.log("Updating dashboard stats...");

// Ensure data is loaded (or use stored values)

const totalEvents = (JSON.parse(localStorage.getItem('events')) || []).length;

const totalAttendees = (JSON.parse(localStorage.getItem('attendees')) || []).length;

const totalSponsors = (JSON.parse(localStorage.getItem('sponsors')) || []).length;

const totalUsers = (JSON.parse(localStorage.getItem('users')) || []).length; // Assuming users are stored

document.getElementById('total-events').textContent = totalEvents;

document.getElementById('total-attendees').textContent = totalAttendees;

document.getElementById('total-sponsors').textContent = totalSponsors;

document.getElementById('total-users').textContent = totalUsers;

}

// --- Placeholder Action Functions ---

/\*\*

\* Saves or updates an event (Placeholder - uses localStorage).

\* @param {Event} e - The form submission event.

\*/

function saveEvent(e) {

e.preventDefault(); // Prevent default form submission

const eventId = document.getElementById('event-id').value;

const eventData = {

id: eventId || `evt${Date.now()}`, // Generate new ID if creating

name: document.getElementById('event-name').value.trim(),

description: document.getElementById('event-description').value.trim(),

date: document.getElementById('event-date').value,

time: document.getElementById('event-time').value,

location: document.getElementById('event-location').value.trim(),

category: document.getElementById('event-category').value,

price: parseFloat(document.getElementById('event-price').value) || 0,

imageUrl: document.getElementById('event-image-url').value.trim()

};

// Basic validation

if (!eventData.name || !eventData.description || !eventData.date || !eventData.time || !eventData.location) {

alert("Please fill in all required event fields.");

return;

}

if (eventId) {

// Update existing event

const index = events.findIndex(ev => ev.id === eventId);

if (index > -1) {

events[index] = eventData;

console.log("Updating event (placeholder):", eventData);

} else {

console.error("Event ID provided for update, but event not found:", eventId);

alert("Error: Could not find the event to update.");

return;

}

} else {

// Add new event

events.push(eventData);

console.log("Saving new event (placeholder):", eventData);

}

localStorage.setItem('events', JSON.stringify(events)); // Save to localStorage

alert(`Placeholder: Event "${eventData.name}" ${eventId ? 'updated' : 'saved'} successfully!`);

toggleCreateEventForm(true); // Hide form

loadEvents(); // Reload event list

updateDashboardStats(); // Update dashboard

}

/\*\*

\* Initiates the booking process for a selected event.

\* @param {string} eventId - The ID of the event to book.

\*/

function startBooking(eventId) {

if (!loggedInUser) {

alert("Please log in to book tickets.");

showSection('auth');

showAuthForm('login');

return;

}

const event = events.find(e => e.id === eventId);

if (!event) {

alert('Error: Event not found!');

return;

}

console.log(`Starting booking process for event: ${event.name} (ID: ${eventId})`);

// Populate booking form

document.getElementById('booking-event-id').value = event.id;

document.getElementById('booking-event-name').textContent = event.name;

document.getElementById('booking-event-date').textContent = event.date;

document.getElementById('booking-event-time').textContent = event.time;

document.getElementById('booking-event-location').textContent = event.location;

document.getElementById('booking-ticket-price').textContent = event.price.toFixed(2);

// Reset number of tickets and total cost

document.getElementById('num-tickets').value = 1;

updateTotalCost();

// Pre-fill attendee name/email if logged in

if (loggedInUser) {

document.getElementById('attendee-name').value = loggedInUser.name || '';

document.getElementById('attendee-email').value = loggedInUser.email || '';

} else {

document.getElementById('attendee-name').value = '';

document.getElementById('attendee-email').value = '';

}

// Reset payment method selection

document.querySelectorAll('.payment-option').forEach(opt => opt.classList.remove('active'));

document.querySelectorAll('.payment-details').forEach(det => det.classList.remove('active'));

document.getElementById('booking-message').textContent = ''; // Clear previous messages

// Switch to booking section and show details

showSection('booking'); // Pass no link element, relies on internal logic

document.getElementById('booking-details').classList.remove('hidden');

document.getElementById('no-booking-selected').classList.add('hidden');

}

/\*\*

\* Placeholder function for processing the entire booking form submission.

\* In this simplified version, specific payment buttons trigger actions directly.

\* @param {Event} e - The form submission event.

\*/

function processBooking(e) {

e.preventDefault();

// This function might not be strictly needed if payment buttons handle the final action.

// You could add validation here before allowing payment button clicks.

console.log("Booking form submitted (placeholder - action triggered by payment buttons).");

const messageEl = document.getElementById('booking-message');

messageEl.textContent = 'Please select a payment method below and complete the payment step.';

messageEl.className = 'mt-2 text-info';

// alert("Placeholder: Please select a payment method and click the corresponding 'Pay' or 'Confirm' button.");

}

/\*\*

\* Placeholder for processing card payment.

\*/

function processCardPayment() {

const eventId = document.getElementById('booking-event-id').value;

const eventName = document.getElementById('booking-event-name').textContent;

const numTickets = document.getElementById('num-tickets').value;

const attendeeName = document.getElementById('attendee-name').value.trim();

const attendeeEmail = document.getElementById('attendee-email').value.trim();

const cardNumber = document.getElementById('card-number').value; // For simulation only

const expiry = document.getElementById('expiry-date').value; // For simulation only

const cvv = document.getElementById('cvv').value; // For simulation only

const messageEl = document.getElementById('booking-message');

if (!attendeeName || !attendeeEmail || !cardNumber || !expiry || !cvv) {

messageEl.textContent = 'Please fill in your name, email, and all card details.';

messageEl.className = 'mt-2 text-danger';

// alert("Placeholder: Please fill in your name, email, and card details.");

return;

}

console.log("Processing card payment (placeholder)...");

// Simulate success

alert(`Placeholder: Card payment successful for ${numTickets} ticket(s) to ${eventName}!`);

addAttendee(attendeeName, attendeeEmail, eventId, eventName, numTickets);

messageEl.textContent = 'Booking confirmed via Card! Check the Attendees section.';

messageEl.className = 'mt-2 text-success';

// Optionally clear form or redirect after a delay

setTimeout(() => {

document.getElementById('booking-form').reset();

document.getElementById('booking-details').classList.add('hidden');

document.getElementById('no-booking-selected').classList.remove('hidden');

showSection('events'); // Go back to events list

}, 2000);

}

/\*\*

\* Placeholder for processing PayPal payment.

\*/

function processPayPalPayment() {

const eventId = document.getElementById('booking-event-id').value;

const eventName = document.getElementById('booking-event-name').textContent;

const numTickets = document.getElementById('num-tickets').value;

const attendeeName = document.getElementById('attendee-name').value.trim();

const attendeeEmail = document.getElementById('attendee-email').value.trim();

const messageEl = document.getElementById('booking-message');

if (!attendeeName || !attendeeEmail) {

messageEl.textContent = 'Please fill in your name and email.';

messageEl.className = 'mt-2 text-danger';

// alert("Placeholder: Please fill in your name and email.");

return;

}

console.log("Processing PayPal payment (placeholder)...");

alert(`Placeholder: Redirecting to PayPal... Payment successful for ${numTickets} ticket(s) to ${eventName}!`);

addAttendee(attendeeName, attendeeEmail, eventId, eventName, numTickets);

messageEl.textContent = 'Booking confirmed via PayPal! Check the Attendees section.';

messageEl.className = 'mt-2 text-success';

// Optionally clear form or redirect after a delay

setTimeout(() => {

document.getElementById('booking-form').reset();

document.getElementById('booking-details').classList.add('hidden');

document.getElementById('no-booking-selected').classList.remove('hidden');

showSection('events'); // Go back to events list

}, 2000);

}

/\*\*

\* Placeholder for processing free event or pay later booking.

\*/

function processFreeOrPayLater() {

const eventId = document.getElementById('booking-event-id').value;

const eventName = document.getElementById('booking-event-name').textContent;

const numTickets = document.getElementById('num-tickets').value;

const attendeeName = document.getElementById('attendee-name').value.trim();

const attendeeEmail = document.getElementById('attendee-email').value.trim();

const messageEl = document.getElementById('booking-message');

if (!attendeeName || !attendeeEmail) {

messageEl.textContent = 'Please fill in your name and email.';

messageEl.className = 'mt-2 text-danger';

// alert("Placeholder: Please fill in your name and email.");

return;

}

console.log("Processing Free/Pay Later booking (placeholder)...");

alert(`Placeholder: Booking confirmed for ${numTickets} ticket(s) to ${eventName}!`);

addAttendee(attendeeName, attendeeEmail, eventId, eventName, numTickets);

messageEl.textContent = 'Booking confirmed! Check the Attendees section.';

messageEl.className = 'mt-2 text-success';

// Optionally clear form or redirect after a delay

setTimeout(() => {

document.getElementById('booking-form').reset();

document.getElementById('booking-details').classList.add('hidden');

document.getElementById('no-booking-selected').classList.remove('hidden');

showSection('events'); // Go back to events list

}, 2000);

}

/\*\*

\* Adds a new attendee record (Placeholder - uses localStorage).

\*/

function addAttendee(name, email, eventId, eventName, tickets) {

const newAttendee = {

id: `att${Date.now()}`,

name: name,

email: email,

eventId: eventId,

eventName: eventName,

tickets: parseInt(tickets),

checkedIn: false

};

attendees.push(newAttendee);

localStorage.setItem('attendees', JSON.stringify(attendees));

console.log("Added new attendee (placeholder):", newAttendee);

updateDashboardStats(); // Update dashboard count

// Reload attendee list if currently viewing it

if (currentSection === 'attendees') {

filterAttendees();

}

}

/\*\*

\* Placeholder for checking in/out an attendee.

\* @param {string} attendeeId - The ID of the attendee.

\* @param {boolean} checkInState - True to check in, false to undo.

\*/

function checkInAttendee(attendeeId, checkInState) {

const index = attendees.findIndex(att => att.id === attendeeId);

if (index > -1) {

attendees[index].checkedIn = checkInState;

localStorage.setItem('attendees', JSON.stringify(attendees));

console.log(`Placeholder: Attendee ${attendeeId} check-in status set to ${checkInState}.`);

// alert(`Placeholder: Attendee ${checkInState ? 'checked in' : 'check-out recorded'}.`);

// Re-render the current view if in the attendees section

if (currentSection === 'attendees') {

filterAttendees(); // Re-apply filters to refresh the view

}

} else {

console.error(`Attendee with ID ${attendeeId} not found for check-in.`);

alert('Error: Attendee not found.');

}

}

/\*\*

\* Placeholder for removing an attendee.

\* @param {string} attendeeId - The ID of the attendee to remove.

\*/

function removeAttendee(attendeeId) {

if (confirm('Are you sure you want to remove this attendee? This action cannot be undone.')) {

const initialLength = attendees.length;

attendees = attendees.filter(att => att.id !== attendeeId);

if (attendees.length < initialLength) {

localStorage.setItem('attendees', JSON.stringify(attendees));

console.log(`Placeholder: Removed attendee ${attendeeId}.`);

alert('Placeholder: Attendee removed.');

if (currentSection === 'attendees') {

filterAttendees(); // Refresh list

}

updateDashboardStats(); // Update dashboard count

} else {

console.error(`Attendee with ID ${attendeeId} not found for removal.`);

alert('Error: Attendee not found.');

}

}

}

/\*\*

\* Adds or updates a sponsor (Placeholder - uses localStorage).

\* @param {Event} e - The form submission event.

\*/

function addOrUpdateSponsor(e) {

e.preventDefault();

const sponsorId = document.getElementById('sponsor-id').value;

const sponsorData = {

id: sponsorId || `spn${Date.now()}`,

name: document.getElementById('sponsor-name').value.trim(),

level: document.getElementById('sponsor-level').value,

contact: document.getElementById('sponsor-contact').value.trim(),

logoUrl: document.getElementById('sponsor-logo-url').value.trim()

};

if (!sponsorData.name) {

alert("Please enter the sponsor name.");

return;

}

if (sponsorId) {

// Update

const index = sponsors.findIndex(sp => sp.id === sponsorId);

if (index > -1) {

sponsors[index] = sponsorData;

console.log("Updating sponsor (placeholder):", sponsorData);

} else {

console.error("Sponsor ID provided for update, but sponsor not found:", sponsorId);

alert("Error: Could not find the sponsor to update.");

return;

}

} else {

// Add

sponsors.push(sponsorData);

console.log("Adding new sponsor (placeholder):", sponsorData);

}

localStorage.setItem('sponsors', JSON.stringify(sponsors));

alert(`Placeholder: Sponsor "${sponsorData.name}" ${sponsorId ? 'updated' : 'added'} successfully!`);

hideSponsorForm();

loadSponsors(); // Refresh list

updateDashboardStats(); // Update dashboard count

}

/\*\*

\* Prepares the form for editing a specific sponsor.

\* @param {string} sponsorId - The ID of the sponsor to edit.

\*/

function editSponsor(sponsorId) {

const sponsor = sponsors.find(sp => sp.id === sponsorId);

if (sponsor) {

showSponsorForm(sponsor);

} else {

alert('Error: Sponsor not found.');

console.error(`Sponsor with ID ${sponsorId} not found for editing.`);

}

}

/\*\*

\* Placeholder for removing a sponsor.

\* @param {string} sponsorId - The ID of the sponsor to remove.

\*/

function removeSponsor(sponsorId) {

if (confirm('Are you sure you want to remove this sponsor?')) {

const initialLength = sponsors.length;

sponsors = sponsors.filter(sp => sp.id !== sponsorId);

if (sponsors.length < initialLength) {

localStorage.setItem('sponsors', JSON.stringify(sponsors));

console.log(`Placeholder: Removed sponsor ${sponsorId}.`);

alert('Placeholder: Sponsor removed.');

if (currentSection === 'sponsors') {

loadSponsors(); // Refresh list

}

updateDashboardStats(); // Update dashboard count

} else {

console.error(`Sponsor with ID ${sponsorId} not found for removal.`);

alert('Error: Sponsor not found.');

}

}

}

/\*\*

\* Placeholder for generating analytics reports.

\*/

function generateAnalytics() {

console.log("Generating analytics report (placeholder)...");

const outputDiv = document.getElementById('analytics-output');

// Simulate report generation

const totalEvents = events.length;

const totalAttendees = attendees.length;

const totalCheckIns = attendees.filter(a => a.checkedIn).length;

const totalSponsorsCount = sponsors.length;

// Simulate revenue (very basic)

const totalRevenue = attendees.reduce((sum, att) => {

const event = events.find(e => e.id === att.eventId);

return sum + (event ? event.price \* att.tickets : 0);

}, 0);

outputDiv.innerHTML = `

<h4>Analytics Report (Simulated - ${new Date().toLocaleDateString()})</h4>

<p><strong>Total Events Managed:</strong> ${totalEvents}</p>

<p><strong>Total Attendees Registered:</strong> ${totalAttendees}</p>

<p><strong>Total Event Check-ins:</strong> ${totalCheckIns}</p>

<p><strong>Total Sponsors:</strong> ${totalSponsorsCount}</p>

<p><strong>Simulated Total Revenue:</strong> $${totalRevenue.toFixed(2)}</p>

<br>

<p><i>Note: This is simulated data based on current localStorage entries. More detailed charts and breakdowns would require a dedicated analytics library and potentially backend processing.</i></p>

`;

alert("Placeholder: Analytics report generated!");

}

/\*\*

\* Placeholder for user login. Uses localStorage.

\* @param {Event} e - The form submission event.

\*/

function loginUser(e) {

e.preventDefault();

const email = document.getElementById('login-email').value.trim();

const password = document.getElementById('login-password').value; // Not actually checked here

const messageEl = document.getElementById('login-message');

// Basic validation

if (!email || !password) {

messageEl.textContent = 'Please enter both email and password.';

return;

}

console.log(`Attempting login for ${email} (placeholder)...`);

// --- Simulation ---

// In a real app, you'd send this to a server for validation.

// Here, we'll check against users stored in localStorage (if any).

const users = JSON.parse(localStorage.getItem('users')) || [];

const foundUser = users.find(u => u.email === email); // Simple check by email

// Simulate successful login if user exists or for any input if no users stored yet

if (foundUser || users.length === 0) {

loggedInUser = foundUser || { email: email, name: email.split('@')[0] }; // Use stored name or generate one

localStorage.setItem('loggedInUser', JSON.stringify(loggedInUser)); // Store login state

messageEl.textContent = '';

alert(`Placeholder: Welcome back, ${loggedInUser.name}!`);

updateAuthState();

showSection('dashboard'); // Redirect to dashboard after login

} else {

messageEl.textContent = 'Invalid email or password (simulation).';

console.log("Login failed (placeholder): User not found in localStorage.");

}

}

/\*\*

\* Placeholder for user registration. Uses localStorage.

\* @param {Event} e - The form submission event.

\*/

function registerUser(e) {

e.preventDefault();

const name = document.getElementById('register-name').value.trim();

const email = document.getElementById('register-email').value.trim();

const password = document.getElementById('register-password').value; // Not actually stored securely

const messageEl = document.getElementById('register-message');

// Basic validation

if (!name || !email || !password) {

messageEl.textContent = 'Please fill in all fields.';

messageEl.className = 'text-danger text-center mt-2';

return;

}

// Basic email format check

if (!/\S+@\S+\.\S+/.test(email)) {

messageEl.textContent = 'Please enter a valid email address.';

messageEl.className = 'text-danger text-center mt-2';

return;

}

console.log(`Attempting registration for ${email} (placeholder)...`);

// --- Simulation ---

// Check if user already exists in localStorage

let users = JSON.parse(localStorage.getItem('users')) || [];

if (users.some(u => u.email === email)) {

messageEl.textContent = 'Email already registered. Try logging in.';

messageEl.className = 'text-danger text-center mt-2';

return;

}

// Add new user (store only name and email for this simulation)

users.push({ name: name, email: email });

localStorage.setItem('users', JSON.stringify(users));

messageEl.textContent = 'Registration successful! Please login.';

messageEl.className = 'text-success text-center mt-2';

alert('Placeholder: Registration successful! You can now log in.');

document.getElementById('register-form').reset(); // Clear registration form

showAuthForm('login'); // Switch to login form

updateDashboardStats(); // Update user count if shown on dashboard

}

/\*\*

\* Placeholder for user logout.

\*/

function logoutUser() {

console.log("Logging out user (placeholder)...");

loggedInUser = null;

localStorage.removeItem('loggedInUser');

alert("Placeholder: You have been logged out.");

updateAuthState();

// Reset potentially user-specific data views

wishlist = [];

localStorage.removeItem('wishlist'); // Clear wishlist on logout

showSection('auth'); // Show login form after logout

showAuthForm('login');

}

/\*\*

\* Updates the visibility of Login/Register/Logout buttons based on login state.

\*/

function updateAuthState() {

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

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

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

if (loggedInUser) {

loginBtn.classList.add('hidden');

registerBtn.classList.add('hidden');

logoutBtn.classList.remove('hidden');

// Update sidebar welcome message or user display if you add one

} else {

loginBtn.classList.remove('hidden');

registerBtn.classList.remove('hidden');

logoutBtn.classList.add('hidden');

}

}

/\*\*

\* Toggles an event in the user's wishlist (Placeholder - uses localStorage).

\* @param {string} eventId - The ID of the event to toggle.

\* @param {HTMLElement} buttonElement - The wishlist button element.

\*/

function toggleWishlist(eventId, buttonElement) {

if (!loggedInUser) {

alert("Please log in to manage your wishlist.");

showSection('auth');

showAuthForm('login');

return;

}

// Ensure wishlist is loaded from storage for the current user session

wishlist = JSON.parse(localStorage.getItem('wishlist')) || [];

const index = wishlist.indexOf(eventId);

if (index > -1) {

// Remove from wishlist

wishlist.splice(index, 1);

buttonElement.classList.remove('active');

buttonElement.innerHTML = '♡'; // Empty heart

console.log(`Removed event ${eventId} from wishlist.`);

} else {

// Add to wishlist

wishlist.push(eventId);

buttonElement.classList.add('active');

buttonElement.innerHTML = '❤️'; // Filled heart

console.log(`Added event ${eventId} to wishlist.`);

}

localStorage.setItem('wishlist', JSON.stringify(wishlist));

// Refresh wishlist section if currently viewing it

if (currentSection === 'wishlist') {

loadWishlist();

}

}

/\*\*

\* Placeholder function for removing an item directly from the wishlist view.

\* @param {string} eventId - The ID of the event to remove.

\*/

function removeFromWishlist(eventId) {

// Ensure wishlist is loaded from storage

wishlist = JSON.parse(localStorage.getItem('wishlist')) || [];

const index = wishlist.indexOf(eventId);

if (index > -1) {

wishlist.splice(index, 1);

localStorage.setItem('wishlist', JSON.stringify(wishlist));

console.log(`Removed event ${eventId} from wishlist via wishlist section.`);

alert('Event removed from wishlist.');

loadWishlist(); // Refresh the wishlist view

// Also update the heart icon on the main event list if it's loaded

const eventCardButton = document.querySelector(`.event-card[data-event-id="${eventId}"] .btn-wishlist`);

if (eventCardButton) {

eventCardButton.classList.remove('active');

eventCardButton.innerHTML = '♡';

}

} else {

console.warn(`Attempted to remove event ${eventId} from wishlist view, but it wasn't found in the list.`);

}

}

/\*\*

\* Placeholder function for setting a reminder.

\* @param {string} eventId - The ID of the event.

\*/

function setReminder(eventId) {

const event = events.find(e => e.id === eventId);

if (event) {

console.log(`Setting reminder for event ${eventId} (placeholder)...`);

alert(`Placeholder: Reminder set for "${event.name}" on ${event.date}! (In a real app, this might use browser notifications or calendar integration).`);

} else {

alert('Error: Could not find event details to set reminder.');

}

}

// --- Chatbot Functions ---

/\*\*

\* Gets a basic response from the chatbot based on keywords.

\* @param {string} message - The user's message.

\* @returns {string} The bot's response.

\*/

function getBotResponse(message) {

const lowerMessage = message.toLowerCase();

let response = "I'm sorry, I didn't understand that. Can you please rephrase? You can ask about events, booking, login, or sponsors.";

// Simple keyword matching

if (lowerMessage.includes('hello') || lowerMessage.includes('hi') || lowerMessage.includes('hey')) {

response = 'Hello there! How can I assist you with the event portal?';

lastContext = { type: null, data: null };

} else if (lowerMessage.includes('event') && !lowerMessage.includes('book')) {

response = 'You can view upcoming events in the "Events" section. Use the filters to search by name, category, or date. You can also create a new event there if you have permission.';

lastContext = { type: 'event\_info', data: null };

} else if (lowerMessage.includes('book') || lowerMessage.includes('ticket') || lowerMessage.includes('register for event')) {

if (lastContext.type === 'event\_info' && lastContext.data?.eventId) {

response = `To book tickets for ${lastContext.data.eventName}, go to the "Events" section, find the event, and click "Book Now".`;

} else {

response = 'To book tickets, please go to the "Events" section, find the event you are interested in, and click the "Book Now" button. You\'ll need to be logged in. The booking process continues in the "Booking" section.';

}

lastContext = { type: 'booking\_info', data: null };

} else if (lowerMessage.includes('login') || lowerMessage.includes('register') || lowerMessage.includes('account') || lowerMessage.includes('logout')) {

response = 'You can log in, register for an account, or log out using the buttons in the sidebar. If you are on the Login/Register page, you can switch between forms using the links provided.';

lastContext = { type: 'auth\_info', data: null };

} else if (lowerMessage.includes('sponsor')) {

response = 'Information about event sponsors can be viewed and managed in the "Sponsors" section.';

lastContext = { type: 'sponsor\_info', data: null };

} else if (lowerMessage.includes('attendee')) {

response = 'Manage event attendees and check-ins in the "Attendees" section. You can filter by event or search by name/email.';

lastContext = { type: null, data: null };

} else if (lowerMessage.includes('wishlist') || lowerMessage.includes('reminder') || lowerMessage.includes('save event')) {

response = 'Add events to your wishlist via the ♡ icon on event cards (Events section). View your wishlist & set reminders in the "Wishlist" section (login required).';

lastContext = { type: 'wishlist\_info', data: null };

} else if (lowerMessage.includes('analytic') || lowerMessage.includes('report')) {

response = 'Generate simulated reports on attendance, revenue, etc., in the "Analytics" section.';

lastContext = { type: null, data: null };

} else if (lowerMessage.includes('thank')) {

response = "You're welcome! Let me know if you need anything else.";

lastContext = { type: null, data: null };

} else if (lowerMessage.includes('bye') || lowerMessage.includes('goodbye')) {

response = "Goodbye! Have a great day.";

lastContext = { type: null, data: null };

}

return response;

}

/\*\*

\* Sends the user's message and gets/displays the bot's response.

\*/

function sendChatMessage() {

const input = document.getElementById('chatbot-input');

const body = document.getElementById('chatbot-body');

if (!input || !body) return; // Ensure elements exist

const message = input.value.trim();

if (!message) return; // Don't send empty messages

appendChatMessage('You', message);

input.value = ''; // Clear input field

const botResponse = getBotResponse(message);

// Simulate bot thinking time

setTimeout(() => {

appendChatMessage('Bot', botResponse);

}, 300 + Math.random() \* 400);

}

/\*\*

\* Appends a message to the chatbot body.

\* @param {string} sender - 'You' or 'Bot'.

\* @param {string} message - The message text.

\*/

function appendChatMessage(sender, message) {

const body = document.getElementById('chatbot-body');

if (!body) return;

const messageElement = document.createElement('p');

// Sanitize message slightly to prevent basic HTML injection

const cleanMessage = message.replace(/</g, "&lt;").replace(/>/g, "&gt;");

messageElement.innerHTML = `<strong>${sender}:</strong> ${cleanMessage}`;

body.appendChild(messageElement);

// Scroll to the bottom

body.scrollTop = body.scrollHeight;

}

/\*\*

\* Toggles the minimized state of the chatbot.

\* @param {Event} [ev] - Optional click event (to stop propagation).

\*/

function minimizeChatbot(ev) {

ev?.stopPropagation(); // Prevent header click from triggering when clicking button

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

const minimizeButton = chatbot?.querySelector('.chatbot-header button');

if (!chatbot || !minimizeButton) return;

chatbot.classList.toggle('minimized');

// Change icon based on state

if (chatbot.classList.contains('minimized')) {

minimizeButton.innerHTML = `<svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"><path d="M21 15a2 2 0 0 1-2 2H7l-4 4V5a2 2 0 0 1 2-2h14a2 2 0 0 1 2 2z"></path></svg>`; // Chat icon

} else {

minimizeButton.textContent = '\_'; // Minimize icon

}

}

// --- Initialization ---

/\*\*

\* Runs when the page loads to set up the initial state.

\*/

window.onload = function() {

console.log("Initializing Event Portal...");

// Check login state on load

loggedInUser = JSON.parse(localStorage.getItem('loggedInUser'));

updateAuthState();

// Load initial data for the default section (Dashboard) and others needed for context/stats

// Load events first as other data might reference it

loadEvents();

loadAttendees();

loadSponsors();

loadWishlist(); // Load wishlist state (might be empty if not logged in)

// Show the initial section (Dashboard or Auth if not logged in)

if (loggedInUser) {

showSection('dashboard', document.querySelector('.sidebar nav ul li a[onclick\*="dashboard"]')); // Show dashboard by default if logged in

updateDashboardStats(); // Make sure stats are up-to-date

} else {

showSection('auth', document.getElementById('loginBtn')); // Show auth section if not logged in

showAuthForm('login');

updateDashboardStats(); // Show initial stats (likely 0)

}

// Add listener for Enter key in chatbot input

const chatbotInput = document.getElementById('chatbot-input');

if (chatbotInput) {

chatbotInput.addEventListener('keydown', function(event) {

if (event.key === 'Enter') {

sendChatMessage();

}

});

} else {

console.error("Chatbot input element not found.");

}

console.log("Event Portal Initialized.");

};

</script>

</body>

</html>