<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<meta name="description" content="CLUES-TES: Plan journeys in London with real-time TfL and Lime bike data, smart notifications, and accessibility features.">

<meta name="keywords" content="transit planning, London, TfL API, Lime bikes, weather, pollution, TES score, accessible routes">

<meta property="og:title" content="CLUES-TES Transit Planner">

<meta property="og:description" content="Intelligent transit planning with real-time TfL and bike integrations.">

<meta property="og:type" content="website">

<title>CLUES-TES | Enhanced London Transit Experience Score</title>

<link rel="manifest" href="data:application/json;base64,eyJuYW1lIjoiQ0xVRVMtVEVTIiwic2hvcnRfbmFtZSI6IkNMVUVTLVRFUyIsInN0YXJ0X3VybCI6Ii8iLCJkaXNwbGF5Ijoic3RhbmRhbG9uZSIsImJhY2tncm91bmRfY29sb3IiOiIjZmZmZmZmIiwidGhlbWVfY29sb3IiOiIjM2I4MmY2In0=">

<meta name="theme-color" content="#3b82f6">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">

<link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700;800;900&display=swap" rel="stylesheet">

<script src="https://cdn.jsdelivr.net/npm/chart.js@4.4.0/dist/chart.min.js"></script>

<script>

// Ensure Chart.js is loaded

if (typeof Chart === 'undefined') {

console.error('Chart.js failed to load from CDN, trying backup...');

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

script.src = 'https://cdnjs.cloudflare.com/ajax/libs/Chart.js/4.4.0/chart.min.js';

script.onload = () => console.log('Chart.js loaded from backup CDN');

document.head.appendChild(script);

} else {

console.log('✅ Chart.js loaded successfully');

}

</script>

<style>

:root {

--primary-50: #eff6ff; --primary-100: #dbeafe; --primary-500: #3b82f6; --primary-600: #2563eb;

--primary-700: #1d4ed8; --primary-900: #1e3a8a; --secondary-500: #f97316; --secondary-600: #ea580c;

--success-500: #10b981; --warning-500: #f59e0b; --danger-500: #ef4444;

--gray-50: #f8fafc; --gray-100: #f1f5f9; --gray-200: #e2e8f0; --gray-300: #cbd5e1; --gray-400: #94a3b8;

--gray-500: #64748b; --gray-600: #475569; --gray-700: #334155; --gray-800: #1e293b;

--gray-900: #0f172a;

--light-yellow: #fefce8; --light-gold: #fef3c7; --steel-gray: #64748b;

--sunshine-orange: #fed7af; --air-excellent: #10b981; --air-good: #84cc16; --air-moderate: #f59e0b;

--air-poor: #f97316; --air-hazardous: #ef4444;

--gradient-primary: linear-gradient(135deg, var(--primary-600) 0%, var(--primary-500) 50%, var(--secondary-500) 100%);

--gradient-surface: linear-gradient(135deg, rgba(255,255,255,0.9) 0%, rgba(255,255,255,0.7) 100%);

--shadow-sm: 0 1px 2px 0 rgb(0 0 0 / 0.05);

--shadow-md: 0 4px 6px -1px rgb(0 0 0 / 0.1), 0 2px 4px -2px rgb(0 0 0 / 0.1);

--shadow-lg: 0 10px 15px -3px rgb(0 0 0 / 0.1), 0 4px 6px -4px rgb(0 0 0 / 0.1);

--shadow-xl: 0 20px 25px -5px rgb(0 0 0 / 0.1), 0 8px 10px -6px rgb(0 0 0 / 0.1);

--radius-sm: 0.375rem; --radius-md: 0.5rem; --radius-lg: 0.75rem; --radius-xl: 1rem;

--transition: all 0.2s cubic-bezier(0.4, 0, 0.2, 1);

--transition-bounce: all 0.3s cubic-bezier(0.68, -0.55, 0.265, 1.55);

}

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

body {

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

background: var(--gradient-primary);

min-height: 100vh;

color: var(--gray-900);

line-height: 1.6;

overflow-x: hidden;

}

body.dark-mode {

--gradient-primary: linear-gradient(135deg, #1e3a8a 0%, #334155 50%, #475569 100%);

--gradient-surface: linear-gradient(135deg, rgba(30,41,59,0.9) 0%, rgba(30,41,59,0.7) 100%);

--gray-900: #f1f5f9;

--gray-700: #cbd5e1;

--gray-600: #e2e8f0;

--gray-500: #f1f5f9;

--gray-200: #1e293b;

--gray-100: #334155;

--primary-500: #60a5fa;

--secondary-500: #fb923c;

--success-500: #4ade80;

--warning-500: #fbbf24;

--danger-500: #f87171;

}

.skeleton { background: linear-gradient(90deg, var(--gray-200) 25%, var(--gray-100) 50%, var(--gray-200) 75%); background-size: 200% 100%; animation: pulse 1.5s ease-in-out infinite; border-radius: var(--radius-md); }

@keyframes pulse { 0%, 100% { background-position: 200% 0; } 50% { background-position: -200% 0; } }

.loading-overlay { position: fixed; top: 0; left: 0; right: 0; bottom: 0; background: rgba(255, 255, 255, 0.9); backdrop-filter: blur(4px); display: flex; align-items: center; justify-content: center; z-index: 10000; opacity: 0; pointer-events: none; transition: var(--transition); }

.loading-overlay.show { opacity: 1; pointer-events: all; }

.loading-spinner { width: 48px; height: 48px; border: 4px solid var(--gray-200); border-top: 4px solid var(--primary-500); border-radius: 50%; animation: spin 1s linear infinite; }

@keyframes spin { 0% { transform: rotate(0deg); } 100% { transform: rotate(360deg); } }

.container { max-width: 1400px; margin: 0 auto; padding: 1.5rem; }

.glass-panel { background: var(--gradient-surface); backdrop-filter: blur(20px); border-radius: var(--radius-xl); border: 1px solid rgba(255, 255, 255, 0.2); box-shadow: var(--shadow-xl); padding: 2rem; margin-bottom: 1.5rem; transition: var(--transition); }

.glass-panel:hover { transform: translateY(-2px); box-shadow: 0 25px 50px -12px rgb(0 0 0 / 0.25); }

.header { position: relative; margin-bottom: 2rem; overflow: hidden; }

.header::before { content: ''; position: absolute; top: 0; left: 0; right: 0; height: 4px; background: var(--gradient-primary); border-radius: var(--radius-xl) var(--radius-xl) 0 0; }

.header-content { display: flex; align-items: center; justify-content: space-between; flex-wrap: wrap; gap: 1rem; }

.brand { display: flex; align-items: center; gap: 1rem; }

.logo { width: 64px; height: 64px; background: var(--gradient-primary); border-radius: var(--radius-xl); display: flex; align-items: center; justify-content: center; font-size: 1.5rem; color: white; box-shadow: var(--shadow-lg); }

.brand-info h1 { font-size: 2.5rem; font-weight: 800; background: var(--gradient-primary); -webkit-background-clip: text; -webkit-text-fill-color: transparent; background-clip: text; margin-bottom: 0.25rem; }

.brand-info .subtitle { color: var(--gray-600); font-weight: 600; font-size: 1.125rem; }

.header-actions { display: flex; gap: 0.75rem; flex-wrap: wrap; }

.action-btn { background: rgba(255, 255, 255, 0.1); border: 1px solid rgba(255, 255, 255, 0.2); color: var(--gray-700); padding: 0.5rem 1rem; border-radius: var(--radius-lg); font-weight: 500; cursor: pointer; transition: var(--transition); display: flex; align-items: center; gap: 0.5rem; }

.action-btn:hover { background: rgba(255, 255, 255, 0.2); transform: translateY(-1px); }

.main-grid { display: grid; grid-template-columns: 1fr 400px; gap: 2rem; }

.section-title { font-size: 1.25rem; font-weight: 700; color: var(--gray-900); margin-bottom: 1rem; display: flex; align-items: center; gap: 0.5rem; }

.dynamic-feedback { background: var(--primary-50); border: 2px solid var(--primary-200); border-radius: var(--radius-lg); padding: 1rem; margin: 1rem 0; text-align: center; font-weight: 500; display: none; }

.location-input-group { position: relative; }

.location-input-wrapper { display: flex; align-items: center; gap: 0.5rem; }

.location-input { flex-grow: 1; }

.location-action-btn { background: var(--gray-100); border: 2px solid var(--gray-200); color: var(--gray-600); width: 48px; height: 48px; border-radius: var(--radius-lg); font-size: 1.2rem; cursor: pointer; transition: var(--transition); }

.location-action-btn:hover { background: var(--gray-200); color: var(--primary-500); }

.nomad-hotspots { margin: 1.5rem 0; }

.hotspot-grid { display: grid; grid-template-columns: repeat(auto-fit, minmax(120px, 1fr)); gap: 0.75rem; }

.hotspot-btn { background: var(--primary-50); border: 2px solid var(--primary-200); border-radius: var(--radius-lg); padding: 0.75rem; text-align: center; cursor: pointer; transition: var(--transition); font-weight: 600; font-size: 0.8rem; }

.hotspot-btn:hover { background: var(--primary-100); border-color: var(--primary-500); transform: translateY(-2px); }

.route-option .route-actions { display: flex; gap: 0.75rem; margin-top: 1rem; }

.route-action-btn { flex-grow: 1; background: var(--gray-100); border: 2px solid var(--gray-200); color: var(--gray-700); padding: 0.5rem 1rem; border-radius: var(--radius-lg); font-weight: 600; cursor: pointer; transition: var(--transition); display: flex; align-items: center; justify-content: center; gap: 0.5rem; }

.route-action-btn:hover { background: var(--gray-200); color: var(--primary-500); }

.city-grid { display: grid; grid-template-columns: 1fr; gap: 1rem; }

.city-card { background: var(--gray-50); border: 2px solid var(--gray-200); border-radius: var(--radius-lg); padding: 1.25rem; text-align: center; cursor: pointer; transition: var(--transition-bounce); position: relative; overflow: hidden; }

.city-card::before { content: ''; position: absolute; top: 0; left: -100%; width: 100%; height: 100%; background: linear-gradient(90deg, transparent, rgba(255,255,255,0.4), transparent); transition: left 0.5s; }

.city-card:hover::before { left: 100%; }

.city-card:hover, .city-card.active { border-color: var(--primary-500); background: var(--primary-50); transform: translateY(-4px) scale(1.02); box-shadow: var(--shadow-lg); }

.city-card.active { border-color: var(--primary-600); background: var(--primary-100); }

.city-flag { font-size: 2.5rem; margin-bottom: 0.75rem; display: block; transition: var(--transition); }

.city-name { font-weight: 700; color: var(--gray-900); margin-bottom: 0.5rem; font-size: 0.9rem; }

.city-tes { background: var(--gradient-primary); color: white; padding: 0.25rem 0.75rem; border-radius: var(--radius-lg); font-size: 0.8rem; font-weight: 600; display: inline-block; }

.weather-display { background: var(--light-yellow); border: 2px solid var(--light-gold); border-radius: var(--radius-lg); padding: 1rem; margin: 1rem 0; text-align: center; display: flex; align-items: center; justify-content: space-between; gap: 1rem; }

.weather-info { display: flex; align-items: center; gap: 0.75rem; }

.weather-icon { font-size: 2rem; }

.weather-temp { font-size: 1.5rem; font-weight: 700; color: var(--gray-900); }

.weather-desc { font-size: 0.875rem; color: var(--gray-600); }

.pollution-monitor { background: var(--gray-50); border: 2px solid var(--gray-200); border-radius: var(--radius-lg); padding: 1rem; margin: 1rem 0; display: flex; align-items: center; justify-content: space-between; gap: 1rem; }

.pollution-level { display: flex; align-items: center; gap: 0.75rem; }

.pollution-icon { width: 48px; height: 48px; border-radius: 50%; display: flex; align-items: center; justify-content: center; font-size: 1.25rem; color: white; }

.pollution-excellent { background: var(--air-excellent); }

.pollution-good { background: var(--air-good); }

.pollution-moderate { background: var(--air-moderate); }

.pollution-poor { background: var(--air-poor); }

.pollution-hazardous { background: var(--air-hazardous); }

.pollution-text { font-weight: 600; color: var(--gray-900); }

.pollution-aqi { font-size: 0.875rem; color: var(--gray-600); }

.journey-planner { background: var(--gray-50); border: 2px solid var(--primary-200); border-radius: var(--radius-xl); padding: 1.5rem; margin: 1.5rem 0; }

.location-inputs { display: flex; flex-direction: column; gap: 1rem; margin-bottom: 1.5rem; }

.location-label { display: flex; align-items: center; gap: 0.5rem; font-weight: 600; color: var(--gray-900); margin-bottom: 0.5rem; font-size: 0.875rem; }

.location-input { width: 100%; padding: 1rem 1.25rem; border: 2px solid var(--gray-200); border-radius: var(--radius-lg); font-size: 1rem; transition: var(--transition); background: white; font-family: inherit; }

.location-input:focus { outline: none; border-color: var(--primary-500); box-shadow: 0 0 0 3px rgba(59, 130, 246, 0.1); }

.swap-locations { display: flex; align-items: center; justify-content: center; width: 48px; height: 48px; background: var(--primary-500); color: white; border-radius: 50%; cursor: pointer; transition: var(--transition-bounce); margin: 0.5rem auto; border: none; font-size: 1.25rem; }

.swap-locations:hover { background: var(--primary-600); transform: rotate(180deg) scale(1.1); box-shadow: var(--shadow-lg); }

.priority-sliders { display: grid; gap: 1rem; margin: 1.5rem 0; }

.priority-slider { display: flex; flex-direction: column; gap: 0.5rem; }

.priority-slider span { font-weight: 600; display: flex; align-items: center; gap: 0.5rem; }

.slider { width: 100%; -webkit-appearance: none; height: 8px; border-radius: var(--radius-md); background: var(--gray-200); outline: none; transition: var(--transition); }

.slider::-webkit-slider-thumb { -webkit-appearance: none; width: 20px; height: 20px; border-radius: 50%; background: var(--primary-500); cursor: pointer; box-shadow: var(--shadow-sm); }

.control-grid { display: grid; gap: 1rem; margin-bottom: 1.5rem; }

.preferences-grid { grid-template-columns: repeat(2, 1fr); }

.methods-grid { grid-template-columns: repeat(3, 1fr); }

.amenities-grid { grid-template-columns: repeat(4, 1fr); }

.control-btn { background: white; border: 2px solid var(--gray-200); border-radius: var(--radius-lg); padding: 1rem 0.75rem; text-align: center; cursor: pointer; transition: var(--transition-bounce); font-size: 0.875rem; position: relative; overflow: hidden; }

.control-btn:hover { border-color: var(--primary-500); background: var(--primary-50); transform: translateY(-2px); box-shadow: var(--shadow-md); }

.control-btn.active { background: var(--primary-500); border-color: var(--primary-600); color: white; transform: translateY(-2px) scale(1.02); box-shadow: var(--shadow-lg); }

.method-btn { border: 2px solid var(--gray-200); border-radius: var(--radius-lg); padding: 1rem 0.75rem; text-align: center; cursor: pointer; transition: var(--transition-bounce); font-size: 0.875rem; position: relative; overflow: hidden; color: white; }

.method-btn[data-method="metro"] { background: linear-gradient(135deg, #1e3a8a, #2563eb); }

.method-btn[data-method="bus"] { background: linear-gradient(135deg, #f97316, #fb923c); }

.method-btn[data-method="train"] { background: linear-gradient(135deg, #475569, #64748b); }

.method-btn[data-method="bike"] { background: linear-gradient(135deg, #10b981, #34d399); }

.method-btn[data-method="walk"] { background: linear-gradient(135deg, #f59e0b, #fbbf24); }

.method-btn[data-method="rideshare"] { background: linear-gradient(135deg, #334155, #475569); }

.method-btn:hover, .method-btn.active { transform: translateY(-2px) scale(1.02); box-shadow: var(--shadow-md); }

.method-btn.active { border-width: 3px; border-color: white; transform: scale(1.05); box-shadow: 0 0 0 3px rgba(59, 130, 246, 0.3); }

.amenity-btn { background: white; border: 2px solid var(--gray-200); border-radius: var(--radius-lg); padding: 1rem 0.5rem; text-align: center; cursor: pointer; transition: var(--transition-bounce); font-size: 0.8rem; position: relative; overflow: hidden; }

.amenity-btn:hover, .amenity-btn.active { transform: translateY(-2px) scale(1.02); box-shadow: var(--shadow-md); }

.amenity-btn.active { border-width: 3px; color: var(--gray-900); font-weight: 600; box-shadow: 0 0 0 3px rgba(59, 130, 246, 0.3); background: var(--primary-100); }

.control-icon { font-size: 1.5rem; margin-bottom: 0.5rem; display: block; transition: var(--transition); }

.control-label { font-weight: 600; font-size: 0.8rem; }

.search-btn { background: var(--gradient-primary); color: white; padding: 1.25rem 2rem; font-size: 1.125rem; border: none; border-radius: var(--radius-xl); font-family: inherit; font-weight: 700; cursor: pointer; transition: var(--transition-bounce); display: flex; align-items: center; justify-content: center; gap: 0.75rem; width: 100%; margin-top: 1.5rem; position: relative; overflow: hidden; }

.search-btn::before { content: ''; position: absolute; top: 0; left: -100%; width: 100%; height: 100%; background: linear-gradient(90deg, transparent, rgba(255,255,255,0.2), transparent); transition: left 0.6s; }

.search-btn:hover::before { left: 100%; }

.search-btn:hover { transform: translateY(-3px); box-shadow: var(--shadow-xl); }

.search-btn:disabled { opacity: 0.7; cursor: not-allowed; transform: none; }

.map-download-btn { background: var(--success-500); color: white; padding: 0.75rem 1.5rem; font-size: 1rem; border: none; border-radius: var(--radius-lg); font-family: inherit; font-weight: 600; cursor: pointer; transition: var(--transition); display: flex; align-items: center; justify-content: center; gap: 0.5rem; width: 100%; margin-top: 1rem; opacity: 0; pointer-events: none; }

.map-download-btn.show { opacity: 1; pointer-events: all; animation: slideInUp 0.3s ease-out; }

.map-download-btn:hover { background: var(--success-600); transform: translateY(-2px); box-shadow: var(--shadow-md); }

.sidebar .glass-panel { padding: 1.5rem; }

.panel-header { display: flex; align-items: center; gap: 0.75rem; margin-bottom: 1.5rem; padding-bottom: 0.75rem; border-bottom: 2px solid var(--gray-100); }

.panel-icon { width: 48px; height: 48px; background: var(--gradient-primary); color: white; border-radius: var(--radius-lg); display: flex; align-items: center; justify-content: center; font-size: 1.25rem; }

.panel-title { font-size: 1.125rem; font-weight: 700; color: var(--gray-900); }

.score-circle { width: 120px; height: 120px; border-radius: 50%; background: var(--gradient-primary); display: flex; align-items: center; justify-content: center; margin: 0 auto 1rem; color: white; font-size: 2.5rem; font-weight: 800; box-shadow: var(--shadow-lg); position: relative; }

.score-circle::before { content: ''; position: absolute; inset: -4px; background: var(--gradient-primary); border-radius: 50%; z-index: -1; animation: pulse-ring 2s infinite; }

@keyframes pulse-ring { 0% { transform: scale(1); opacity: 1; } 100% { transform: scale(1.2); opacity: 0; } }

.score-label { font-size: 1rem; color: var(--gray-700); font-weight: 600; text-align: center; }

.mobility-grid { display: grid; grid-template-columns: repeat(3, 1fr); gap: 0.75rem; }

.mobility-card { background: white; border: 2px solid var(--gray-100); border-radius: var(--radius-lg); padding: 1rem; text-align: center; transition: var(--transition); cursor: pointer; }

.mobility-card:hover { transform: translateY(-2px); box-shadow: var(--shadow-md); }

.mobility-score { font-size: 1.5rem; font-weight: 700; margin-bottom: 0.25rem; }

.mobility-label { font-size: 0.75rem; font-weight: 600; color: var(--gray-600); display: flex; align-items: center; justify-content: center; gap: 0.25rem; }

.walkability { border-color: var(--success-500); } .walkability .mobility-score { color: var(--success-500); }

.bike-score { border-color: var(--secondary-500); } .bike-score .mobility-score { color: var(--secondary-500); }

.transit-score { border-color: var(--primary-500); } .transit-score .mobility-score { color: var(--primary-500); }

.map-container { height: 400px; border-radius: var(--radius-xl); overflow: hidden; margin: 1.5rem 0; border: 2px solid var(--primary-200); position: relative; box-shadow: var(--shadow-lg); }

.map-controls { position: absolute; top: 1rem; left: 1rem; z-index: 1000; display: flex; gap: 0.5rem; flex-wrap: wrap; }

.map-control-btn { background: rgba(255, 255, 255, 0.95); border: none; padding: 0.5rem 0.75rem; border-radius: var(--radius-md); cursor: pointer; font-size: 0.8rem; font-weight: 600; transition: var(--transition); box-shadow: var(--shadow-sm); backdrop-filter: blur(8px); }

.map-control-btn:hover, .map-control-btn.active { background: var(--primary-500); color: white; transform: translateY(-1px); box-shadow: var(--shadow-md); }

.route-results { background: white; border: 2px solid var(--success-500); border-radius: var(--radius-xl); padding: 1.5rem; margin: 1.5rem 0; display: none; box-shadow: var(--shadow-lg); }

.route-results.show { display: block; animation: slideInUp 0.3s ease-out; }

@keyframes slideInUp { from { transform: translateY(20px); opacity: 0; } to { transform: translateY(0); opacity: 1; } }

.route-option { background: var(--gray-50); border: 2px solid var(--gray-200); border-radius: var(--radius-lg); padding: 1.25rem; margin-bottom: 1rem; transition: var(--transition); position: relative; cursor: pointer; }

.route-option:hover { border-color: var(--primary-500); box-shadow: var(--shadow-md); transform: translateY(-1px); }

.route-option.recommended { border-color: var(--success-500); background: rgba(16, 185, 129, 0.05); }

.route-option.recommended::before { content: '⭐ RECOMMENDED'; position: absolute; top: -8px; right: 1rem; background: var(--success-500); color: white; padding: 0.25rem 0.75rem; border-radius: var(--radius-lg); font-size: 0.7rem; font-weight: 700; }

.route-option.selected { border-color: var(--success-500) !important; background: rgba(16, 185, 129, 0.1) !important; transform: translateY(-2px) scale(1.02) !important; box-shadow: var(--shadow-lg) !important; }

.route-option.selected::after { content: '✓ SELECTED'; position: absolute; top: -8px; left: 1rem; background: var(--success-500); color: white; padding: 0.25rem 0.75rem; border-radius: var(--radius-lg); font-size: 0.7rem; font-weight: 700; }

.route-option.recommended.selected::before { right: 6rem; }

.chart-container { position: relative; background: white; border-radius: var(--radius-lg); padding: 1rem; margin: 1rem 0; box-shadow: var(--shadow-sm); height: 250px; }

.chart-fallback { display: none; text-align: center; padding: 1rem; background: var(--gray-50); border-radius: var(--radius-lg); color: var(--gray-600); }

.infra-grid { display: grid; grid-template-columns: repeat(2, 1fr); gap: 0.75rem; margin-top: 1rem; }

.infra-card { background: white; border: 2px solid var(--gray-100); border-radius: var(--radius-lg); padding: 1rem; text-align: center; cursor: pointer; }

.infra-card:hover { border-color: var(--primary-500); }

.infra-score { font-size: 1.25rem; font-weight: 700; }

.infra-label { font-size: 0.75rem; color: var(--gray-600); }

.status-indicator { display: inline-flex; align-items: center; gap: 0.5rem; padding: 0.5rem 0.75rem; border-radius: var(--radius-lg); font-size: 0.875rem; font-weight: 500; margin: 0.25rem 0; }

.status-indicator.loading { background: rgba(245, 158, 11, 0.1); color: var(--warning-500); border: 1px solid rgba(245, 158, 11, 0.2); }

.status-indicator.success { background: rgba(16, 185, 129, 0.1); color: var(--success-500); border: 1px solid rgba(16, 185, 129, 0.2); }

.status-indicator.error { background: rgba(239, 68, 68, 0.1); color: var(--danger-500); border: 1px solid rgba(239, 68, 68, 0.2); }

#terms-modal { opacity: 0; transform: translateY(20px); transition: opacity 0.5s ease, transform 0.5s ease; }

#terms-modal.show { opacity: 1; transform: translateY(0); }

@media (max-width: 1024px) {

.main-grid { grid-template-columns: 1fr; }

.header-content { flex-direction: column; text-align: center; gap: 1.5rem; }

.brand-info h1 { font-size: 2rem; }

}

@media (max-width: 768px) {

.container { padding: 1rem; }

.glass-panel { padding: 1.25rem; }

.preferences-grid, .methods-grid, .amenities-grid { grid-template-columns: repeat(2, 1fr); }

.city-grid { grid-template-columns: 1fr; }

.mobility-grid { grid-template-columns: repeat(2, 1fr); }

.map-container { height: 300px; }

}

@media (max-width: 480px) {

.brand-info h1 { font-size: 1.75rem; }

.control-grid { grid-template-columns: 1fr; }

}

.modal-overlay { position: fixed; top: 0; left: 0; right: 0; bottom: 0; background: rgba(0, 0, 0, 0.6); backdrop-filter: blur(10px); display: flex; align-items: center; justify-content: center; z-index: 20000; opacity: 0; pointer-events: none; transition: opacity 0.3s ease-in-out; }

.modal-overlay.show { opacity: 1; pointer-events: all; }

.modal-content { background: var(--gradient-surface); border-radius: var(--radius-xl); border: 1px solid rgba(255, 255, 255, 0.2); box-shadow: var(--shadow-xl); padding: 2.5rem; width: 90%; max-width: 600px; max-height: 80vh; overflow-y: auto; transform: scale(0.95); transition: transform 0.3s ease-in-out; }

.modal-overlay.show .modal-content { transform: scale(1); }

.modal-content h2 { font-size: 1.5rem; font-weight: 700; margin-bottom: 1rem; color: var(--gray-900); }

.modal-content p { margin-bottom: 1rem; line-height: 1.6; }

.modal-content ul { list-style-position: inside; padding-left: 1rem; margin-bottom: 1rem; }

.modal-content pre { white-space: pre-wrap; word-wrap: break-word; font-family: inherit; font-size: 0.875rem; line-height: 1.5; background: rgba(0, 0, 0, 0.05); padding: 1rem; border-radius: var(--radius-md); color: var(--gray-700); margin-bottom: 1.5rem; }

.modal-content .modal-button { background: var(--gradient-primary); color: white; padding: 0.75rem 1.5rem; font-size: 1rem; border: none; border-radius: var(--radius-lg); font-family: inherit; font-weight: 600; cursor: pointer; transition: var(--transition-bounce); width: 100%; }

.modal-content .modal-button:hover { transform: translateY(-2px); box-shadow: var(--shadow-lg); }

.notification-container { position: fixed; top: 1.5rem; right: 1.5rem; z-index: 30000; display: flex; flex-direction: column; gap: 0.75rem; }

.notification-toast { padding: 1rem 1.5rem; border-radius: var(--radius-lg); box-shadow: var(--shadow-lg); color: white; font-weight: 600; display: flex; align-items: center; gap: 0.75rem; opacity: 0; transform: translateX(20px); animation: slideInToast 0.3s ease-out forwards; max-width: 350px; }

@keyframes slideInToast { to { opacity: 1; transform: translateX(0); } }

.notification-toast.error { background: linear-gradient(135deg, var(--danger-500), #f87171); }

.notification-toast.success { background: linear-gradient(135deg, var(--success-500), #4ade80); }

.notification-toast.info { background: linear-gradient(135deg, var(--primary-500), #60a5fa); }

.notification-toast.warning { background: linear-gradient(135deg, var(--warning-500), #fbbf24); }

.terms-dropdown { background: var(--gradient-surface); backdrop-filter: blur(20px); border-radius: var(--radius-lg); border: 1px solid rgba(255, 255, 255, 0.2); box-shadow: var(--shadow-md); padding: 1rem; position: fixed; top: 10px; left: 50%; transform: translateX(-50%); z-index: 10001; width: 80%; max-width: 400px; display: none; }

.terms-dropdown.show { display: block; }

.terms-dropdown .close-btn { background: var(--primary-500); color: white; border: none; padding: 0.5rem 1rem; border-radius: var(--radius-md); cursor: pointer; margin-top: 1rem; }

.pass-tab { background: var(--gradient-primary); color: white; padding: 0.5rem 1rem; border-radius: var(--radius-lg); font-weight: 600; cursor: pointer; transition: var(--transition); }

.pass-tab:hover { transform: translateY(-2px); }

.header-actions { gap: 0.5rem; }

/\* Event tabs styling \*/

.event-tab-container { display: flex; gap: 0.5rem; margin: 1rem 0; flex-wrap: wrap; }

.event-tab { background: var(--gray-100); padding: 0.5rem 1rem; border-radius: var(--radius-lg); cursor: pointer; transition: var(--transition); font-size: 0.875rem; font-weight: 600; }

.event-tab:hover, .event-tab.active { background: var(--primary-500); color: white; transform: translateY(-1px); }

/\* --- Icon Size Enhancement --- \*/

.method-btn .control-icon, .amenity-btn .control-icon {

font-size: 2.2rem !important; /\* Significantly larger icons \*/

display: block;

margin: 0 auto 0.5rem; /\* Center icon and add space below \*/

transition: transform 0.2s ease-in-out;

}

.method-btn:hover .control-icon, .amenity-btn:hover .control-icon {

transform: scale(1.1); /\* Add feedback on hover \*/

}

.amenity-btn .control-icon {

color: var(--primary-500); /\* Give amenity icons a default color \*/

}

.amenity-btn.active .control-icon {

color: var(--gray-900); /\* Change icon color when active \*/

}

</style>

</head>

<body>

<div class="terms-dropdown" id="termsDropdown">

<p>By using CLUES-TES, you agree to our Terms of Service. For details, see full terms.</p>

<button class="close-btn" onclick="acceptTerms()">OK</button>

</div>

<div id="loadingOverlay" class="loading-overlay"><div class="loading-spinner"></div></div>

<!-- Terms Modal -->

<div id="termsModal" class="modal-overlay">

<div class="modal-content">

<h2>Terms of Service & Important Disclaimers</h2>

<pre id="termsText">

Terms of Service, Emergency Disclaimer, and Liability Waiver

Welcome to CLUES-TES. By using this application, you agree to the following terms and conditions.

1. \*\*For Guidance Only\*\*: This application is intended for general transit planning and informational purposes only. All data, including routes, times, weather, and air quality, is provided for guidance and is not guaranteed to be accurate or up-to-date. Always verify critical information with official sources (e.g., Transport for London, official weather services).

2. \*\*⚠️ EMERGENCY DISCLAIMER\*\*: This application is NOT for emergency use. In case of a medical or other emergency, call 999 or your local emergency number immediately. Do not rely on this application for emergency transport, navigation, or decision-making.

3. \*\*No Liability\*\*: The creators and providers of this application shall not be held liable for any damages, losses, delays, or injuries arising from the use or inability to use this application. This includes, but is not limited to, damages from inaccurate data, application errors, or route recommendations. You use this application entirely at your own risk.

4. \*\*Assumption of Risk\*\*: You assume all risks associated with your travel decisions. The application does not account for all possible variables, such as real-time events, safety conditions, construction, or personal health factors.

By clicking "I Understand and Agree," you acknowledge that you have read, understood, and accepted these terms.

</pre>

<button id="agreeButton" class="modal-button">I Understand and Agree</button>

</div>

</div>

<!-- Passes Modal (updated with links) -->

<div id="passesModal" class="modal-overlay">

<div class="modal-content">

<h2>London Travel Passes & Fares</h2>

<p>Navigating London's fare system is easy once you know the basics. Here are the most common ways to pay for your journey:</p>

<ul>

<li><strong>Contactless Payment:</strong> Simply tap your contactless card or mobile device on the yellow card readers at the start and end of your journey. This is often the cheapest option and has daily and weekly price caps. <a href="https://tfl.gov.uk/fares/how-to-pay-and-where-to-buy-tickets-and-oyster/pay-as-you-go/contactless-and-mobile-pay-as-you-go" target="\_blank">More details on TfL</a>.</li>

<li><strong>Oyster Card:</strong> A reusable smartcard you can top up with credit. It works like contactless and also benefits from price capping. You may need to pay a small deposit for the card. <a href="https://tfl.gov.uk/fares/how-to-pay-and-where-to-buy-tickets-and-oyster/oyster-cards-and-travelcards" target="\_blank">Buy or top up Oyster</a>.</li>

<li><strong>Travelcards:</strong> Paper tickets or digital passes loaded onto an Oyster card that give you unlimited travel within specific zones for a set period (e.g., one day, one week). <a href="https://tfl.gov.uk/fares/find-fares/travelcard" target="\_blank">Travelcard options</a>.</li>

<li><strong>Visitor/Tourist Passes:</strong> Special Oyster for visitors with discounts on attractions. <a href="https://visitorshop.tfl.gov.uk/" target="\_blank">Visitor Oyster card</a>.</li>

<li><strong>Retiree Discounts:</strong> 60+ London Oyster photocard for free travel. <a href="https://tfl.gov.uk/fares/free-and-discounted-travel/60-plus-london-oyster-photocard" target="\_blank">60+ Oyster</a>.</li>

<li><strong>Monthly/Weekly Riders:</strong> Season tickets for regular commuters. <a href="https://tfl.gov.uk/fares/find-fares/season-tickets" target="\_blank">Season tickets</a>.</li>

</ul>

<p>For most visitors and nomads, using a contactless card is the most convenient and cost-effective method.</p>

<button onclick="closeModal('passesModal')" class="modal-button">Close</button>

</div>

</div>

<div id="notificationContainer" class="notification-container"></div>

<div class="container">

<header class="header glass-panel">

<div class="header-content">

<div class="brand">

<div class="logo"><i class="fas fa-route" aria-hidden="true"></i></div>

<div class="brand-info">

<h1>CLUES-TES</h1>

<p class="subtitle">Enhanced London Transit Experience Score</p>

</div>

</div>

<div class="header-actions">

<button class="action-btn" onclick="openModal('passesModal')"><i class="fas fa-ticket-alt" aria-hidden="true"></i><span>Passes</span></button>

<button class="action-btn" onclick="toggleDarkMode()"><i class="fas fa-moon" aria-hidden="true"></i><span>Dark Mode</span></button>

<button class="action-btn" onclick="toggleA11y()"><i class="fas fa-universal-access" aria-hidden="true"></i><span>Accessibility</span></button>

<button class="action-btn" onclick="showHelp()"><i class="fas fa-question-circle" aria-hidden="true"></i><span>Help</span></button>

</div>

</div>

</header>

<main class="main-grid">

<section class="route-planner glass-panel">

<h2 class="section-title"><i class="fas fa-globe-europe" aria-hidden="true"></i><span>London Transit Planner</span></h2>

<div class="city-grid">

<div class="city-card active" data-city="london" tabindex="0" role="button" aria-pressed="true">

<div class="city-flag" aria-hidden="true">🇬🇧</div>

<div class="city-name">London</div>

<div class="city-tes" id="london-tes">TES: 85</div>

</div>

</div>

<div id="dynamicFeedback" class="dynamic-feedback"></div>

<div class="weather-display" id="weatherDisplay">

<div class="weather-info">

<div class="weather-icon" id="weatherIcon">🌤️</div>

<div><div class="weather-temp" id="weatherTemp">18°C</div><div class="weather-desc" id="weatherDesc">Partly Cloudy</div></div>

</div>

<div style="font-size: 0.875rem; color: var(--gray-600);">Perfect for cycling and walking</div>

</div>

<div class="pollution-monitor" id="pollutionMonitor">

<div class="pollution-level">

<div class="pollution-icon pollution-good" id="pollutionIcon"><i class="fas fa-leaf"></i></div>

<div><div class="pollution-text" id="pollutionText">Good Air Quality</div><div class="pollution-aqi" id="pollutionAQI">AQI: 42 • Safe for outdoor activities</div></div>

</div>

<div style="font-size: 0.875rem; color: var(--gray-600);"><i class="fas fa-info-circle"></i> Real-time monitoring</div>

</div>

<div id="event-tab-container" class="event-tab-container"></div>

<h3 class="section-title"><i class="fas fa-route" aria-hidden="true"></i><span>Plan Your Journey</span></h3>

<div class="journey-planner">

<div class="location-inputs">

<div class="location-input-group">

<label class="location-label" for="fromLocation"><i class="fas fa-circle" style="color: var(--success-500);" aria-hidden="true"></i><span>Starting Location</span></label>

<div class="location-input-wrapper">

<input type="text" id="fromLocation" class="location-input" placeholder="Enter starting location..." aria-describedby="fromLocationHelp" />

<button class="location-action-btn" onclick="useCurrentLocation()" aria-label="Use my current location">

<i class="fas fa-crosshairs"></i>

</button>

</div>

</div>

<button class="swap-locations" onclick="swapLocations()" aria-label="Swap starting and destination locations"><i class="fas fa-exchange-alt" aria-hidden="true"></i></button>

<div class="location-input-group">

<label class="location-label" for="toLocation"><i class="fas fa-map-marker-alt" style="color: var(--danger-500);" aria-hidden="true"></i><span>Destination</span></label>

<input type="text" id="toLocation" class="location-input" placeholder="Enter destination..." aria-describedby="toLocationHelp" />

</div>

</div>

</div>

<div class="nomad-hotspots">

<h3 class="section-title"><i class="fas fa-star" aria-hidden="true"></i><span>Nomad Hotspots</span></h3>

<div class="hotspot-grid">

<button class="hotspot-btn" data-location="Heathrow Airport, London, UK">Heathrow</button>

<button class="hotspot-btn" data-location="King's Cross Station, London, UK">King's Cross</button>

<button class="hotspot-btn" data-location="Shoreditch, London, UK">Shoreditch</button>

<button class="hotspot-btn" data-location="Camden Town, London, UK">Camden</button>

</div>

</div>

<h3 class="section-title"><i class="fas fa-sliders-h" aria-hidden="true"></i><span>Set Priorities (1-3)</span></h3>

<div class="priority-sliders">

<div class="priority-slider"><span><i class="fas fa-star"></i><span>Ride Quality</span></span><input type="range" min="1" max="3" value="3" class="slider" data-priority="ride-quality"></div>

<div class="priority-slider"><span><i class="fas fa-tachometer-alt"></i><span>Speed</span></span><input type="range" min="1" max="3" value="2" class="slider" data-priority="speed"></div>

<div class="priority-slider"><span><i class="fas fa-euro-sign"></i><span>Cost</span></span><input type="range" min="1" max="3" value="2" class="slider" data-priority="cost"></div>

<div class="priority-slider"><span><i class="fas fa-wheelchair"></i><span>Accessibility</span></span><input type="range" min="1" max="3" value="1" class="slider" data-priority="accessibility"></div>

<div class="priority-slider"><span><i class="fas fa-leaf"></i><span>Environmental Impact</span></span><input type="range" min="1" max="3" value="2" class="slider" data-priority="environmental"></div>

</div>

<h3 class="section-title"><i class="fas fa-subway" aria-hidden="true"></i><span>Preferred Transport</span></h3>

<div class="control-grid methods-grid">

<div class="method-btn active" data-method="metro" tabindex="0" role="button" aria-pressed="true"><i class="control-icon fas fa-subway" aria-hidden="true"></i><div class="control-label">Metro/Tube</div></div>

<div class="method-btn" data-method="bus" tabindex="0" role="button" aria-pressed="false"><i class="control-icon fas fa-bus" aria-hidden="true"></i><div class="control-label">Bus</div></div>

<div class="method-btn" data-method="train" tabindex="0" role="button" aria-pressed="false"><i class="control-icon fas fa-train" aria-hidden="true"></i><div class="control-label">Train</div></div>

<div class="method-btn" data-method="bike" tabindex="0" role="button" aria-pressed="false"><i class="control-icon fas fa-bicycle" aria-hidden="true"></i><div class="control-label">Bike Share</div></div>

<div class="method-btn" data-method="walk" tabindex="0" role="button" aria-pressed="false"><i class="control-icon fas fa-walking" aria-hidden="true"></i><div class="control-label">Walking</div></div>

<div class="method-btn" data-method="rideshare" tabindex="0" role="button" aria-pressed="false"><i class="control-icon fas fa-car" aria-hidden="true"></i><div class="control-label">Rideshare</div></div>

</div>

<h3 class="section-title"><i class="fas fa-star" aria-hidden="true"></i><span>Important Amenities</span></h3>

<div class="control-grid amenities-grid">

<div class="amenity-btn" data-amenity="wifi" tabindex="0" role="button" aria-pressed="false"><i class="control-icon fas fa-wifi" aria-hidden="true"></i><div class="control-label">Free WiFi</div></div>

<div class="amenity-btn" data-amenity="ac" tabindex="0" role="button" aria-pressed="false"><i class="control-icon fas fa-snowflake" aria-hidden="true"></i><div class="control-label">Air Conditioning</div></div>

<div class="amenity-btn" data-amenity="bathrooms" tabindex="0" role="button" aria-pressed="false"><i class="control-icon fas fa-restroom" aria-hidden="true"></i><div class="control-label">Restrooms</div></div>

<div class="amenity-btn" data-amenity="elevators" tabindex="0" role="button" aria-pressed="false"><i class="control-icon fas fa-elevator" aria-hidden="true"></i><div class="control-label">Elevators</div></div>

<div class="amenity-btn" data-amenity="food" tabindex="0" role="button" aria-pressed="false"><i class="control-icon fas fa-utensils" aria-hidden="true"></i><div class="control-label">Food Options</div></div>

<div class="amenity-btn" data-amenity="shopping" tabindex="0" role="button" aria-pressed="false"><i class="control-icon fas fa-shopping-bag" aria-hidden="true"></i><div class="control-label">Shopping</div></div>

<div class="amenity-btn" data-amenity="charging" tabindex="0" role="button" aria-pressed="false"><i class="control-icon fas fa-battery-three-quarters" aria-hidden="true"></i><div class="control-label">Phone Charging</div></div>

<div class="amenity-btn" data-amenity="security" tabindex="0" role="button" aria-pressed="false"><i class="control-icon fas fa-shield-alt" aria-hidden="true"></i><div class="control-label">Security</div></div>

</div>

<h3 class="section-title"><i class="fas fa-map" aria-hidden="true"></i><span>Interactive Transit Map</span></h3>

<div class="map-container" role="img" aria-label="Interactive transit map">

<div class="map-controls">

<button class="map-control-btn active" data-layer="transit" aria-pressed="true"><i class="fas fa-subway" aria-hidden="true"></i><span>Transit</span></button>

<button class="map-control-btn" data-layer="bike" aria-pressed="false"><i class="fas fa-bicycle" aria-hidden="true"></i><span>Bikes</span></button>

<button class="map-control-btn" data-layer="traffic" aria-pressed="false"><i class="fas fa-car" aria-hidden="true"></i><span>Traffic</span></button>

<button class="map-control-btn" data-layer="accessibility" aria-pressed="false"><i class="fas fa-wheelchair" aria-hidden="true"></i><span>Accessible</span></button>

</div>

<div id="map" style="height: 100%; width: 100%;"></div>

</div>

<button class="search-btn" id="searchButton" onclick="findRoutes()"><i class="fas fa-search" aria-hidden="true"></i><span>Find Optimal Routes</span></button>

<div class="route-results" id="routeResults" role="region" aria-label="Route search results">

<h3 class="section-title"><i class="fas fa-route" aria-hidden="true"></i><span>Recommended Routes</span></h3>

<div id="routeList"></div>

<div class="download-buttons">

<button class="map-download-btn" onclick="downloadRouteMap(app.currentRoute?.id)">Download Route Map</button>

<button class="map-download-btn" onclick="downloadAreaMap()">Download Area Map</button>

</div>

</div>

</section>

<aside class="sidebar">

<div class="glass-panel">

<div class="panel-header"><div class="panel-icon"><i class="fas fa-brain" aria-hidden="true"></i></div><h3 class="panel-title">Live Transit Intelligence</h3></div>

<div class="score-circle" id="currentTESScore" role="img" aria-label="Current TES Score">85</div>

<div class="score-label">CLUES-TES Score</div>

<div style="margin-top: 1rem; font-size: 0.875rem; color: var(--gray-600); text-align: center;"><div id="apiConnections">🚀 All systems operational</div></div>

</div>

<div class="glass-panel">

<div class="panel-header"><div class="panel-icon"><i class="fas fa-chart-pie" aria-hidden="true"></i></div><h3 class="panel-title">TES Breakdown</h3></div>

<div class="chart-container">

<canvas id="tesRadarChart" width="300" height="200" aria-label="TES score breakdown radar chart"></canvas>

<div class="chart-fallback" id="chartFallback">Chart loading...</div>

</div>

</div>

<div class="glass-panel">

<div class="panel-header"><div class="panel-icon"><i class="fas fa-chart-line" aria-hidden="true"></i></div><h3 class="panel-title">Mobility & Environment</h3></div>

<div class="mobility-grid">

<div class="mobility-card walkability" role="button" tabindex="0"><div class="mobility-score" id="walkabilityScore">87</div><div class="mobility-label"><i class="fas fa-walking" aria-hidden="true"></i><span>Walkability</span></div></div>

<div class="mobility-card bike-score" role="button" tabindex="0"><div class="mobility-score" id="bikeScore">73</div><div class="mobility-label"><i class="fas fa-bicycle" aria-hidden="true"></i><span>Bike Score</span></div></div>

<div class="mobility-card transit-score" role="button" tabindex="0"><div class="mobility-score" id="transitScore">92</div><div class="mobility-label"><i class="fas fa-subway" aria-hidden="true"></i><span>Transit Score</span></div></div>

<div class="mobility-card walkability" role="button" tabindex="0"><div class="mobility-score" id="airQuality">Good</div><div class="mobility-label"><i class="fas fa-wind" aria-hidden="true"></i><span>Air Quality</span></div></div>

<div class="mobility-card bike-score" role="button" tabindex="0"><div class="mobility-score" id="noiseLevel">45dB</div><div class="mobility-label"><i class="fas fa-volume-down" aria-hidden="true"></i><span>Noise Level</span></div></div>

<div class="mobility-card transit-score" role="button" tabindex="0"><div class="mobility-score" id="safetyScore">89</div><div class="mobility-label"><i class="fas fa-shield-alt" aria-hidden="true"></i><span>Safety Score</span></div></div>

</div>

</div>

<div class="glass-panel">

<div class="panel-header"><div class="panel-icon"><i class="fas fa-road" aria-hidden="true"></i></div><h3 class="panel-title">Infrastructure Stats</h3></div>

<div class="infra-grid" id="infraGrid">

<div class="infra-card" data-type="bike"><div class="infra-score">450km</div><div class="infra-label">Bike Lanes</div></div>

<div class="infra-card" data-type="pedestrian"><div class="infra-score">15,000</div><div class="infra-label">Crossings</div></div>

<div class="infra-card" data-type="footpaths"><div class="infra-score">2,500km</div><div class="infra-label">Footpaths</div></div>

<div class="infra-card" data-type="transit"><div class="infra-score">400+</div><div class="infra-label">Stations</div></div>

</div>

</div>

</aside>

</main>

</div>

<script>

const app = {

map: null,

directionsService: null,

geocoder: null,

directionsRenderer: null,

transitLayer: null,

trafficLayer: null,

bikeMarkers: [],

stationMarkers: [],

tempMarkers: [],

chart: null,

preferences: {

priorities: {

'ride-quality': 3,

speed: 2,

cost: 2,

accessibility: 1,

environmental: 2

},

methods: ['metro'],

amenities: []

},

cityData: {

london: {

tes: 85,

radar: [85, 75, 90, 80, 70, 85],

coords: { lat: 51.5074, lng: -0.1278 }

}

},

stations: [],

routesCache: [],

currentRoute: null

};

window.API\_CONFIG = {

GOOGLE\_MAPS: 'AIzaSyB8ouAet84RCCM4HDLIwvYKYXC\_-HUE-Rk',

OPENWEATHER: '95c4f48f78213345c0036fb589b3c780',

TFL\_API\_KEY: '6e85b101ab8548f492ac7bce667419ac',

LIME\_GBFS\_URL: 'https://gbfs.lime.bike/london.json',

BACKEND\_URL: 'http://localhost:3000',

TIER\_GBFS\_URL: 'https://platform.tier-services.io/v1/gbfs/london',

SANTANDER\_URL: 'https://api.tfl.gov.uk/BikePoint',

STATIC\_MAPS\_KEY: 'AIzaSyDdHZqyUOSPs2HjvJtBWU8rf0aeT-T7LuE',

TICKETMASTER\_KEY: 'AgG5ZCHO0gb1f7WuyPR4YkLAXCsIT71X',

TICKETMASTER\_SECRET: '75Cpk2BIifs23AVv'

};

async function showNearbyBikes(userLat, userLon, map) {

const res = await fetch("https://api.citybik.es/v2/networks/santander-cycles");

const data = await res.json();

const stations = data.network.stations;

function distance(lat1, lon1, lat2, lon2) {

const R = 6371;

const dLat = (lat2 - lat1) \* Math.PI/180;

const dLon = (lon2 - lon1) \* Math.PI/180;

const a = Math.sin(dLat/2)\*\*2 +

Math.cos(lat1 \* Math.PI/180) \* Math.cos(lat2 \* Math.PI/180) \*

Math.sin(dLon/2)\*\*2;

const c = 2 \* Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

return R \* c;

}

stations.forEach(station => {

const dist = distance(userLat, userLon, station.latitude, station.longitude);

if (dist < 1.5) {

new google.maps.Marker({

position: { lat: station.latitude, lng: station.longitude },

map: map,

label: `${station.free\_bikes} 🚲`,

title: station.name

});

}

});

}

function showNotification(message, type = 'info', duration = 3000) {

const container = document.getElementById('notificationContainer');

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

toast.className = `notification-toast ${type}`;

toast.innerHTML = `<i class="fas fa-${type === 'success' ? 'check' : type === 'error' ? 'times' : type === 'warning' ? 'exclamation-triangle' : 'info-circle'}"></i> ${message}`;

container.appendChild(toast);

setTimeout(() => toast.remove(), duration);

}

function openModal(modalId) {

document.getElementById(modalId).classList.add('show');

}

function closeModal(modalId) {

document.getElementById(modalId).classList.remove('show');

}

function toggleDarkMode() {

document.body.classList.toggle('dark-mode');

localStorage.setItem('darkMode', document.body.classList.contains('dark-mode'));

}

function toggleA11y() {

document.body.classList.toggle('high-contrast');

showNotification('High contrast mode ' + (document.body.classList.contains('high-contrast') ? 'enabled' : 'disabled'), 'info');

localStorage.setItem('a11yMode', document.body.classList.contains('high-contrast'));

}

function showHelp() {

showNotification('Help: Visit https://clues-tes.com/help for assistance.', 'info', 7000);

}

function sanitizeInput(input) {

return input.replace(/[<>&'"/]/g, char => ({

'<': '&lt;',

'>': '&gt;',

'&': '&amp;',

"'": '&#39;',

'"': '&quot;',

'/': '&#x2F;'

}[char]));

}

// Initialize on DOM load

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

const modal = document.getElementById('termsModal');

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

if (sessionStorage.getItem('termsAgreed') !== 'true') {

modal.classList.add('show');

}

agreeButton.addEventListener('click', () => {

modal.classList.remove('show');

sessionStorage.setItem('termsAgreed', 'true');

});

showTermsDropdown();

loadEventTabs();

loadPreferences();

// Event listeners for sliders

document.querySelectorAll('.slider').forEach(slider => {

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

console.log(`Priority ${slider.dataset.priority} set to ${slider.value}`);

updatePreferences();

});

});

// Event listeners for method buttons

document.querySelectorAll('.method-btn').forEach(btn => {

btn.addEventListener('click', e => {

e.preventDefault();

btn.classList.toggle('active');

btn.setAttribute('aria-pressed', btn.classList.contains('active'));

console.log(`Method ${btn.dataset.method} ${btn.classList.contains('active') ? 'selected' : 'deselected'}`);

updatePreferences();

});

});

// Event listeners for amenity buttons

document.querySelectorAll('.amenity-btn').forEach(btn => {

btn.addEventListener('click', e => {

e.preventDefault();

btn.classList.toggle('active');

btn.setAttribute('aria-pressed', btn.classList.contains('active'));

console.log(`Amenity ${btn.dataset.amenity} ${btn.classList.contains('active') ? 'selected' : 'deselected'}`);

updatePreferences();

});

});

// Event listeners for map control buttons

document.querySelectorAll('.map-control-btn').forEach(btn => {

btn.addEventListener('click', e => {

e.preventDefault();

document.querySelectorAll('.map-control-btn').forEach(b => b.classList.remove('active'));

btn.classList.add('active');

console.log(`Map layer switched to ${btn.dataset.layer}`);

switchMapLayer(btn.dataset.layer);

// Add nearby bikes functionality when bike layer is selected

if (btn.dataset.layer === 'bike') {

navigator.geolocation.getCurrentPosition(pos => {

const userLat = pos.coords.latitude;

const userLon = pos.coords.longitude;

showNearbyBikes(userLat, userLon, app.map);

});

}

});

});

// Event listeners for hotspot buttons

document.querySelectorAll('.hotspot-btn').forEach(btn => {

btn.addEventListener('click', e => {

e.preventDefault();

document.getElementById('toLocation').value = btn.dataset.location;

console.log(`Destination set to: ${btn.dataset.location}`);

showNotification('Destination set!', 'success');

});

});

setTimeout(checkBackendSimple, 3000);

checkUrlParams();

console.log('✅ CLUES-TES ready for nomads!');

});

function showTermsDropdown() {

if (sessionStorage.getItem('termsAgreed') !== 'true') {

document.getElementById('termsDropdown').classList.add('show');

}

}

function acceptTerms() {

document.getElementById('termsDropdown').classList.remove('show');

sessionStorage.setItem('termsAgreed', 'true');

}

// Initialize Google Maps with better error handling

function initializeMap() {

return new Promise((resolve, reject) => {

// Check if Google Maps is already loaded

if (window.google && window.google.maps) {

console.log('Google Maps already loaded');

resolve();

return;

}

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

script.src = `https://maps.googleapis.com/maps/api/js?key=${window.API\_CONFIG.GOOGLE\_MAPS}&libraries=places,geometry&callback=initGoogleMap&loading=async`;

script.async = true;

script.defer = true;

script.onerror = () => {

console.error('Google Maps script failed to load');

reject(new Error('Google Maps failed to load'));

};

window.initGoogleMap = () => {

try {

app.map = new google.maps.Map(document.getElementById('map'), {

zoom: 12,

center: { lat: 51.5074, lng: -0.1278 },

mapTypeControl: false,

streetViewControl: false,

styles: []

});

app.directionsService = new google.maps.DirectionsService();

app.geocoder = new google.maps.Geocoder();

app.directionsRenderer = new google.maps.DirectionsRenderer({

suppressMarkers: false,

polylineOptions: { strokeColor: '#1d4ed8', strokeWeight: 6, strokeOpacity: 0.9 }

});

app.directionsRenderer.setMap(app.map);

app.transitLayer = new google.maps.TransitLayer();

app.transitLayer.setMap(app.map);

app.trafficLayer = new google.maps.TrafficLayer();

initializeAutocomplete();

console.log('✅ Google Maps initialized successfully');

resolve();

} catch (error) {

console.error('Google Maps initialization failed:', error);

reject(error);

}

};

document.head.appendChild(script);

// Timeout after 15 seconds

setTimeout(() => {

if (!window.google) {

reject(new Error('Google Maps timeout'));

}

}, 15000);

});

}

// Fallback to OpenStreetMap via Leaflet

function initializeLeafletMap() {

// Add Leaflet CSS

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

link.rel = 'stylesheet';

link.href = 'https://unpkg.com/leaflet@1.7.1/dist/leaflet.css';

document.head.appendChild(link);

// Add Leaflet JS

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

script.src = 'https://unpkg.com/leaflet@1.7.1/dist/leaflet.js';

script.onload = () => {

app.map = L.map('map').setView([51.5074, -0.1278], 12);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

maxZoom: 19,

attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'

}).addTo(app.map);

console.log('✅ Leaflet/OSM initialized as fallback');

showNotification('Switched to OpenStreetMap due to Google Maps failure.', 'warning');

};

document.head.appendChild(script);

}

// Main map loading function with comprehensive fallbacks

async function loadMap() {

try {

console.log('🗺️ Initializing map system...');

await initializeMap();

console.log('✅ Map system initialized');

} catch (error) {

console.warn('Google Maps failed, initializing fallback mode:', error);

initializeFallbackMode();

}

// Always try to load data, even if map fails

loadAllData();

}

// Fallback mode for when Google Maps fails

function initializeFallbackMode() {

const mapContainer = document.getElementById('map');

if (mapContainer) {

mapContainer.innerHTML = `

<div style="display: flex; align-items: center; justify-content: center; height: 100%; background: var(--gray-100); border-radius: var(--radius-lg); text-align: center; padding: 2rem;">

<div>

<i class="fas fa-map-marked-alt" style="font-size: 3rem; color: var(--gray-400); margin-bottom: 1rem;"></i>

<h3 style="color: var(--gray-700); margin-bottom: 0.5rem;">Map Offline</h3>

<p style="color: var(--gray-600); font-size: 0.875rem;">Route planning available with manual address entry</p>

<button onclick="enableOfflineRouting()" style="margin-top: 1rem; padding: 0.5rem 1rem; background: var(--primary-500); color: white; border: none; border-radius: var(--radius-md); cursor: pointer;">Enable Text-Based Routing</button>

</div>

</div>

`;

}

// Set helpful placeholders

const fromInput = document.getElementById('fromLocation');

const toInput = document.getElementById('toLocation');

if (fromInput) fromInput.placeholder = "Enter starting location (e.g., King's Cross Station, London)";

if (toInput) toInput.placeholder = "Enter destination (e.g., Heathrow Airport, London)";

showNotification('Map unavailable - using offline mode with basic routing', 'warning', 8000);

}

// Enable basic routing without map

function enableOfflineRouting() {

showNotification('Text-based routing enabled! Enter locations to get basic route info.', 'info');

// Mock route function for when Google Maps is unavailable

window.findRoutesOffline = function() {

const from = document.getElementById('fromLocation').value;

const to = document.getElementById('toLocation').value;

if (!from || !to) {

showNotification('Please enter both starting location and destination.', 'error');

return;

}

// Show mock route data

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

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

routeList.innerHTML = `

<div class="route-option recommended">

<div style="display: flex; justify-content: space-between; align-items: center; margin-bottom: 0.75rem;">

<div style="font-weight: 700; font-size: 1.1rem;">Suggested Route</div>

<div style="background: var(--gradient-primary); color: white; padding: 0.25rem 0.75rem; border-radius: var(--radius-lg); font-weight: 600;">TES: 85</div>

</div>

<div style="padding: 1rem; background: var(--gray-50); border-radius: var(--radius-md); margin-bottom: 1rem;">

<p><strong>From:</strong> ${from}</p>

<p><strong>To:</strong> ${to}</p>

<p><strong>Recommended:</strong> Use TfL Journey Planner or Google Maps app for detailed directions</p>

<p><strong>General Advice:</strong> London has excellent public transport - consider Tube, bus, or walking combinations</p>

</div>

<div style="font-size: 0.875rem; color: var(--gray-600);">

<i class="fas fa-info-circle"></i> For real-time directions, visit <a href="https://tfl.gov.uk/plan-a-journey" target="\_blank">TfL Journey Planner</a>

</div>

</div>

`;

routeResults.classList.add('show');

showNotification('Basic route info provided! Use TfL Journey Planner for detailed directions.', 'success');

};

// Update the search button to use offline routing

const searchBtn = document.getElementById('searchButton');

if (searchBtn) {

searchBtn.onclick = window.findRoutesOffline;

}

}

// Initialize map on DOM load

window.addEventListener('load', loadMap);

function initializeAutocomplete() {

const center = { lat: 51.5074, lng: -0.1278 };

const defaultBounds = {

north: center.lat + 0.1,

south: center.lat - 0.1,

east: center.lng + 0.1,

west: center.lng - 0.1,

};

const inputFrom = document.getElementById("fromLocation");

const inputTo = document.getElementById("toLocation");

if (window.google && google.maps.places) {

try {

// Try new PlaceAutocompleteElement first

if (google.maps.places.PlaceAutocompleteElement) {

const autocompleteFrom = new google.maps.places.PlaceAutocompleteElement({

componentRestrictions: { country: "uk" },

fields: ["address\_components", "geometry", "icon", "name"],

locationBias: defaultBounds

});

autocompleteFrom.id = "autocomplete-from";

const autocompleteTo = new google.maps.places.PlaceAutocompleteElement({

componentRestrictions: { country: "uk" },

fields: ["address\_components", "geometry", "icon", "name"],

locationBias: defaultBounds

});

autocompleteTo.id = "autocomplete-to";

// Insert after the input fields

inputFrom.parentNode.insertBefore(autocompleteFrom, inputFrom.nextSibling);

inputTo.parentNode.insertBefore(autocompleteTo, inputTo.nextSibling);

autocompleteFrom.addEventListener('gmp-placeselect', (event) => {

inputFrom.value = event.place.formattedAddress || event.place.displayName;

});

autocompleteTo.addEventListener('gmp-placeselect', (event) => {

inputTo.value = event.place.formattedAddress || event.place.displayName;

});

} else {

// Fallback to deprecated Autocomplete

const options = {

bounds: defaultBounds,

componentRestrictions: { country: "uk" },

fields: ["address\_components", "geometry", "icon", "name"],

strictBounds: false,

};

new google.maps.places.Autocomplete(inputFrom, options);

new google.maps.places.Autocomplete(inputTo, options);

}

} catch (e) {

console.warn('Autocomplete initialization failed:', e);

showNotification('Location autocomplete unavailable. You can still enter addresses manually.', 'warning');

}

}

}

function updatePreferences() {

app.preferences.priorities = {

'ride-quality': parseInt(document.querySelector('[data-priority="ride-quality"]').value),

speed: parseInt(document.querySelector('[data-priority="speed"]').value),

cost: parseInt(document.querySelector('[data-priority="cost"]').value),

accessibility: parseInt(document.querySelector('[data-priority="accessibility"]').value),

environmental: parseInt(document.querySelector('[data-priority="environmental"]').value)

};

app.preferences.methods = Array.from(document.querySelectorAll('.method-btn.active')).map(btn => btn.dataset.method);

app.preferences.amenities = Array.from(document.querySelectorAll('.amenity-btn.active')).map(btn => btn.dataset.amenity);

console.log('Preferences updated:', app.preferences);

savePreferences();

}

function loadPreferences() {

const savedDarkMode = localStorage.getItem('darkMode');

const savedA11yMode = localStorage.getItem('a11yMode');

const savedPrefs = localStorage.getItem('cluesTesPrefs');

if (savedDarkMode === 'true') document.body.classList.add('dark-mode');

if (savedA11yMode === 'true') document.body.classList.add('high-contrast');

if (savedPrefs) {

app.preferences = JSON.parse(savedPrefs);

console.log("Preferences loaded.");

// Apply preferences to UI

Object.entries(app.preferences.priorities).forEach(([key, value]) => {

const slider = document.querySelector(`.slider[data-priority="${key}"]`);

if (slider) slider.value = value;

});

document.querySelectorAll('.method-btn').forEach(btn => {

btn.classList.toggle('active', app.preferences.methods.includes(btn.dataset.method));

});

document.querySelectorAll('.amenity-btn').forEach(btn => {

btn.classList.toggle('active', app.preferences.amenities.includes(btn.dataset.amenity));

});

}

}

function savePreferences() {

localStorage.setItem('cluesTesPrefs', JSON.stringify(app.preferences));

console.log("Preferences saved.");

}

function useCurrentLocation() {

if (navigator.geolocation) {

showNotification('Getting your location...', 'info');

navigator.geolocation.getCurrentPosition(position => {

if (app.geocoder) {

app.geocoder.geocode({ location: { lat: position.coords.latitude, lng: position.coords.longitude } }, (results, status) => {

if (status === 'OK' && results[0]) {

document.getElementById('fromLocation').value = results[0].formatted\_address;

showNotification('Current location set!', 'success');

} else {

showNotification('Geolocation failed. Please enter a location.', 'error');

}

});

} else {

// Fallback for non-Google Maps

document.getElementById('fromLocation').value = `${position.coords.latitude}, ${position.coords.longitude}`;

showNotification('Current location set!', 'success');

}

}, () => showNotification('Geolocation permission denied.', 'error'));

} else {

showNotification('Geolocation is not supported by this browser.', 'error');

}

}

function swapLocations() {

let from = document.getElementById('fromLocation').value;

let to = document.getElementById('toLocation').value;

from = sanitizeInput(from);

to = sanitizeInput(to);

document.getElementById('fromLocation').value = to;

document.getElementById('toLocation').value = from;

showNotification('Locations swapped!', 'info');

}

function checkUrlParams() {

const urlParams = new URLSearchParams(window.location.search);

const from = urlParams.get('from');

const to = urlParams.get('to');

if (from) document.getElementById('fromLocation').value = sanitizeInput(from);

if (to) document.getElementById('toLocation').value = sanitizeInput(to);

}

function checkBackendSimple() {

fetch(`${window.API\_CONFIG.BACKEND\_URL}/api/health`)

.then(res => res.ok ? res.json() : Promise.reject('Backend unavailable'))

.then(data => {

document.getElementById('apiConnections').textContent = `🚀 Backend: ${data.status || 'Connected'}`;

})

.catch(() => {

document.getElementById('apiConnections').textContent = '🚧 Backend: Offline (using fallback data)';

showNotification('Backend offline, using cached data.', 'warning');

});

}

// Load all data functions

async function loadAllData() {

console.log('📡 Loading live data...');

showLoading(true);

try {

// Check if we have Google Maps loaded

if (!window.google || !window.google.maps) {

console.warn('Google Maps not available, skipping map-dependent features');

showNotification('Running in offline mode - some features limited', 'warning');

}

const dataPromises = [];

// Only add weather/pollution if we're not in a restricted environment

try {

dataPromises.push(updateWeatherDisplay());

} catch (e) {

console.warn('Weather API not accessible');

}

try {

dataPromises.push(updatePollutionDisplay());

} catch (e) {

console.warn('Pollution API not accessible');

}

try {

dataPromises.push(fetchBikeData());

} catch (e) {

console.warn('Bike APIs not accessible');

}

try {

dataPromises.push(fetchAmenitiesAndCalculateTES());

} catch (e) {

console.warn('Station data not accessible');

}

try {

dataPromises.push(loadTFLTrafficData());

} catch (e) {

console.warn('TfL API not accessible');

}

// Wait for all promises, but don't fail if some fail

const results = await Promise.allSettled(dataPromises);

const failed = results.filter(r => r.status === 'rejected').length;

const succeeded = results.filter(r => r.status === 'fulfilled').length;

console.log(`Data loading complete: ${succeeded} succeeded, ${failed} failed`);

updateSystemStatus();

showLoading(false);

if (succeeded > 0) {

showNotification(`Loaded ${succeeded} data sources successfully!`, 'success');

} else {

showNotification('Running in offline mode with cached data', 'warning');

}

} catch (error) {

console.error('❌ Error loading data:', error);

showNotification('Running in offline mode - limited functionality', 'warning');

showLoading(false);

// Still try to initialize basic functionality

try {

fetchAmenitiesAndCalculateTES();

} catch (e) {

console.warn('Fallback initialization failed');

}

}

}

async function updateWeatherDisplay() {

try {

const response = await fetch(`https://api.openweathermap.org/data/2.5/weather?lat=51.5074&lon=-0.1278&appid=${window.API\_CONFIG.OPENWEATHER}&units=metric`);

const data = await response.json();

const tempC = Math.round(data.main.temp);

const tempF = (data.main.temp \* 9/5) + 32;

document.getElementById('weatherTemp').textContent = `${tempC}°C`;

document.getElementById('weatherDesc').textContent = data.weather[0].description;

document.getElementById('weatherIcon').textContent = getWeatherIcon(data.weather[0].main);

checkTemperature(tempF);

} catch (e) {

console.warn('Weather fetch failed:', e);

showNotification('Weather data unavailable.', 'error');

}

}

function getWeatherIcon(main) {

const icons = { Clear: '☀️', Clouds: '☁️', Rain: '🌧️', Snow: '❄️', Thunderstorm: '⛈️', Drizzle: '🌦️' };

return icons[main] || '🌤️';

}

function checkTemperature(tempF) {

const feedbackEl = document.getElementById('dynamicFeedback');

if (tempF >= 80) {

showHeatWarning();

rerouteToCoolTransit();

} else if (tempF >= 60 && tempF < 75) {

feedbackEl.innerHTML = '<i class="fas fa-sun"></i> Great day for walking or cycling!';

feedbackEl.style.display = 'block';

} else if (tempF < 40) {

feedbackEl.innerHTML = '<i class="fas fa-snowflake"></i> Cold weather - consider indoor transport options.';

feedbackEl.style.display = 'block';

}

}

function showHeatWarning() {

showNotification('High temperature detected! Consider air-conditioned transport.', 'warning', 6000);

}

function rerouteToCoolTransit() {

app.preferences.amenities = [...new Set([...app.preferences.amenities, 'ac'])];

const acBtn = document.querySelector('.amenity-btn[data-amenity="ac"]');

if (acBtn && !acBtn.classList.contains('active')) {

acBtn.classList.add('active');

acBtn.setAttribute('aria-pressed', 'true');

}

const feedbackEl = document.getElementById('dynamicFeedback');

feedbackEl.innerHTML = '<i class="fas fa-thermometer-full"></i> High temperature! Prioritizing air-conditioned transport.';

feedbackEl.style.display = 'block';

}

async function updatePollutionDisplay() {

try {

const response = await fetch(`https://api.openweathermap.org/data/2.5/air\_pollution?lat=51.5074&lon=-0.1278&appid=${window.API\_CONFIG.OPENWEATHER}`);

const data = await response.json();

const aqi = data.list[0].main.aqi;

const levels = ['Excellent', 'Good', 'Moderate', 'Poor', 'Hazardous'];

const status = levels[aqi - 1] || 'Unknown';

document.getElementById('pollutionText').textContent = `${status} Air Quality`;

document.getElementById('pollutionAQI').textContent = `AQI: ${data.list[0].main.aqi} • ${getAQIAdvice(aqi)}`;

document.getElementById('pollutionIcon').className = `pollution-icon pollution-${status.toLowerCase().replace(' ', '-')}`;

if (aqi >= 4) {

const feedbackEl = document.getElementById('dynamicFeedback');

feedbackEl.innerHTML = '<i class="fas fa-wind"></i> Air quality is poor. Consider indoor transport like the Tube.';

feedbackEl.style.display = 'block';

}

} catch (e) {

console.warn('Pollution fetch failed:', e);

showNotification('Air quality data unavailable.', 'error');

}

}

function getAQIAdvice(aqi) {

const advice = {

1: 'Safe for outdoor activities',

2: 'Generally safe, sensitive groups beware',

3: 'Unhealthy for sensitive groups',

4: 'Unhealthy, reduce outdoor time',

5: 'Very unhealthy, avoid outdoor activities'

};

return advice[aqi] || 'Check local advisories';

}

// Fetch bike data with backend proxy and fallbacks (Lime -> Tier -> Santander -> mock)

async function fetchBikeData() {

try {

// Try backend proxy first

const response = await fetch(`${window.API\_CONFIG.BACKEND\_URL}/api/bikes`);

if (response.ok) {

const data = await response.json();

if (data.bikes && data.bikes.length > 0) {

addBikeMarkers(data.bikes);

console.log(`✅ Loaded ${data.bikes.length} bikes from backend`);

return;

}

}

} catch (e) {

console.warn('Backend bike fetch failed:', e);

}

// Try direct Lime GBFS

try {

const response = await fetch(window.API\_CONFIG.LIME\_GBFS\_URL);

if (response.ok) {

const data = await response.json();

if (data.data && data.data.bikes) {

addBikeMarkers(data.data.bikes);

console.log(`✅ Loaded ${data.data.bikes.length} Lime bikes directly`);

return;

}

}

} catch (e) {

console.warn('Direct Lime fetch failed:', e);

}

// Try Tier GBFS

try {

const response = await fetch(window.API\_CONFIG.TIER\_GBFS\_URL);

if (response.ok) {

const data = await response.json();

if (data.data && data.data.bikes) {

addBikeMarkers(data.data.bikes.map(bike => ({

lat: bike.lat,

lon: bike.lon,

battery\_level: bike.battery\_level,

type: 'tier'

})));

console.log(`✅ Loaded ${data.data.bikes.length} Tier bikes`);

return;

}

}

} catch (e) {

console.warn('Tier bike fetch failed:', e);

}

// Try Santander bikes from TfL

try {

const response = await fetch(`${window.API\_CONFIG.SANTANDER\_URL}?app\_key=${window.API\_CONFIG.TFL\_API\_KEY}`);

if (response.ok) {

const data = await response.json();

const bikes = data.slice(0, 50).map(point => ({

lat: point.lat,

lon: point.lon,

battery\_level: point.nbBikes > 0 ? 100 : 0,

type: 'santander'

}));

addBikeMarkers(bikes);

console.log(`✅ Loaded ${bikes.length} Santander bike points`);

return;

}

} catch (e) {

console.warn('Santander bike fetch failed:', e);

}

// Fallback to mock data

console.warn('All bike APIs failed, using mock data');

const mockBikes = generateMockBikes();

addBikeMarkers(mockBikes);

}

function generateMockBikes() {

const bikes = [];

const center = { lat: 51.5074, lng: -0.1278 };

for (let i = 0; i < 50; i++) {

const angle = Math.random() \* 2 \* Math.PI;

const distance = Math.random() \* 0.05;

bikes.push({

lat: center.lat + (distance \* Math.cos(angle)),

lon: center.lng + (distance \* Math.sin(angle)),

battery\_level: Math.floor(Math.random() \* 100) + 1,

type: 'mock'

});

}

return bikes;

}

function addBikeMarkers(bikes) {

// Clear existing bike markers

app.bikeMarkers.forEach(marker => {

if (app.map instanceof google.maps.Map) {

marker.setMap(null);

} else if (app.map instanceof L.Map) {

app.map.removeLayer(marker);

}

});

app.bikeMarkers = [];

bikes.forEach(bike => {

if (app.map instanceof google.maps.Map) {

const marker = new google.maps.Marker({

position: { lat: bike.lat, lng: bike.lon },

map: null, // Initially hidden

title: `${bike.type || 'Bike'} - Battery: ${bike.battery\_level}%`,

icon: {

path: google.maps.SymbolPath.CIRCLE,

scale: 6,

fillColor: bike.battery\_level > 50 ? '#32cd32' : '#ffa500',

fillOpacity: 0.8,

strokeWeight: 2,

strokeColor: '#ffffff'

}

});

app.bikeMarkers.push(marker);

} else if (app.map instanceof L.Map) {

const marker = L.circleMarker([bike.lat, bike.lon], {

radius: 6,

fillColor: bike.battery\_level > 50 ? '#32cd32' : '#ffa500',

fillOpacity: 0.8,

weight: 2,

color: '#ffffff'

}).bindPopup(`${bike.type || 'Bike'} - Battery: ${bike.battery\_level}%`);

app.bikeMarkers.push(marker);

}

});

}

async function loadEventTabs() {

const tabContainer = document.getElementById('event-tab-container');

if (!tabContainer) return;

tabContainer.innerHTML = '';

const tabs = ['🎵 Music & Events', '💻 Co-Working Spaces', '🎪 Festivals Nearby', '🛒 Grocery Reminders'];

for (const tab of tabs) {

const t = document.createElement('div');

t.className = 'event-tab';

t.textContent = tab;

t.addEventListener('click', (e) => loadTabContent(tab, e));

tabContainer.appendChild(t);

}

}

function clearTempMarkers() {

app.tempMarkers.forEach(marker => {

if (app.map instanceof google.maps.Map) {

marker.setMap(null);

} else if (app.map instanceof L.Map) {

app.map.removeLayer(marker);

}

});

app.tempMarkers = [];

}

function addTemporaryPlaceMarker(place, iconUrl) {

if (!place.geometry || !place.geometry.location) return;

const marker = new google.maps.Marker({

map: app.map,

position: place.geometry.location,

title: place.name,

icon: {

url: iconUrl,

scaledSize: new google.maps.Size(32, 32)

}

});

app.tempMarkers.push(marker);

}

async function loadTabContent(tabName, event) {

document.querySelectorAll('.event-tab').forEach(t => t.classList.remove('active'));

event.target.classList.add('active');

clearTempMarkers();

// Use the destination of the current route, or the center of London as a fallback

const location = app.currentRoute ? app.currentRoute.route.legs[0].end\_location : new google.maps.LatLng(app.cityData.london.coords.lat, app.cityData.london.coords.lng);

if (!google.maps.places) {

showNotification('Places service is not available.', 'error');

return;

}

const placesService = new google.maps.places.PlacesService(app.map);

let request;

let iconUrl;

let useTicketmaster = false;

switch (tabName) {

case '💻 Co-Working Spaces':

request = { location, radius: 2000, keyword: 'coworking space' };

iconUrl = 'https://maps.google.com/mapfiles/kml/shapes/workoffice.png';

break;

case '🛒 Grocery Reminders':

request = { location, radius: 1500, type: ['supermarket', 'grocery\_or\_supermarket'] };

iconUrl = 'https://maps.google.com/mapfiles/kml/shapes/grocerystore.png';

break;

case '🎵 Music & Events':

useTicketmaster = true;

iconUrl = 'https://maps.google.com/mapfiles/kml/shapes/arts.png';

break;

case '🎪 Festivals Nearby':

useTicketmaster = true;

iconUrl = 'https://maps.google.com/mapfiles/kml/shapes/hiker.png';

break;

default:

return;

}

if (useTicketmaster) {

try {

const date = new Date().toISOString().slice(0, 10);

const classification = tabName === '🎵 Music & Events' ? '&classificationName=Music' : '&keyword=festival&classificationName=Arts & Theatre|Music';

const response = await fetch(`https://app.ticketmaster.com/discovery/v2/events.json?apikey=${window.API\_CONFIG.TICKETMASTER\_KEY}&latlong=${location.lat()},${location.lng()}&radius=10&startDateTime=${date}T00:00:00Z${classification}`);

const data = await response.json();

if (data.\_embedded && data.\_embedded.events) {

showNotification(`Found ${data.\_embedded.events.length} events nearby!`, 'info');

data.\_embedded.events.forEach(event => {

const venue = event.\_embedded.venues[0];

if (venue && venue.location) {

const marker = new google.maps.Marker({

map: app.map,

position: { lat: parseFloat(venue.location.latitude), lng: parseFloat(venue.location.longitude) },

title: event.name,

icon: { url: iconUrl, scaledSize: new google.maps.Size(32, 32) }

});

marker.addListener('click', () => {

new google.maps.InfoWindow({

content: `<h3>${event.name}</h3><p>${event.dates.start.localDate}</p><a href="${event.url}" target="\_blank">More info</a>`

}).open(app.map, marker);

});

app.tempMarkers.push(marker);

}

});

return;

}

} catch (e) {

console.warn('Ticketmaster fetch failed:', e);

showNotification('Ticketmaster unavailable, using fallback search.', 'warning');

}

}

// Fallback to Google Places

placesService.nearbySearch(request, (results, status) => {

if (status === google.maps.places.PlacesServiceStatus.OK && results) {

showNotification(`Found ${results.length} results for "${tabName}"!`, 'success');

results.forEach(place => addTemporaryPlaceMarker(place, iconUrl));

} else {

showNotification(`Could not find results for "${tabName}".`, 'warning');

}

});

}

async function fetchAmenitiesAndCalculateTES() {

const cached = localStorage.getItem('cachedStations');

const cacheTimestamp = localStorage.getItem('cachedStationsTimestamp');

const now = Date.now();

if (cached && cacheTimestamp && (now - cacheTimestamp < 3600000)) {

app.stations = JSON.parse(cached);

calculateTES(app.stations);

return;

}

try {

const res = await fetch(`${window.API\_CONFIG.BACKEND\_URL}/api/stations`);

if (!res.ok) throw new Error('Backend fetch failed');

const data = await res.json();

app.stations = data;

localStorage.setItem('cachedStations', JSON.stringify(data));

localStorage.setItem('cachedStationsTimestamp', now);

calculateTES(data);

} catch (e) {

console.warn('Station data fetch failed:', e);

// Use default stations

app.stations = [

{ name: "King's Cross", score: 90, amenities: ['wifi', 'elevators', 'ac', 'food', 'shopping'], lat: 51.532, lng: -0.123 },

{ name: 'Victoria', score: 85, amenities: ['wifi', 'bathrooms', 'food', 'shopping'], lat: 51.496, lng: -0.144 },

{ name: 'Liverpool Street', score: 88, amenities: ['wifi', 'elevators', 'bathrooms', 'food'], lat: 51.518, lng: -0.082 },

{ name: 'Paddington', score: 82, amenities: ['wifi', 'bathrooms', 'food'], lat: 51.516, lng: -0.177 }

];

calculateTES(app.stations);

}

}

function calculateTES(data) {

// Add delay to ensure DOM is ready

setTimeout(() => {

const canvas = document.getElementById('tesRadarChart');

const fallback = document.getElementById('chartFallback');

if (!canvas) {

console.warn('TES chart canvas not found');

return;

}

// Check if Chart.js is available

if (typeof Chart === 'undefined') {

console.warn('Chart.js not available, showing fallback');

if (fallback) {

fallback.style.display = 'block';

fallback.innerHTML = `

<div style="text-align: center; padding: 2rem;">

<h3>TES Score Breakdown</h3>

<div style="display: grid; grid-template-columns: repeat(2, 1fr); gap: 1rem; margin-top: 1rem;">

<div><strong>Speed:</strong> ${app.cityData.london.radar[0]}/100</div>

<div><strong>Comfort:</strong> ${app.cityData.london.radar[1]}/100</div>

<div><strong>Cost:</strong> ${app.cityData.london.radar[2]}/100</div>

<div><strong>Accessibility:</strong> ${app.cityData.london.radar[3]}/100</div>

<div><strong>Environmental:</strong> ${app.cityData.london.radar[4]}/100</div>

<div><strong>Safety:</strong> ${app.cityData.london.radar[5]}/100</div>

</div>

</div>

`;

canvas.style.display = 'none';

}

return;

}

const ctx = canvas.getContext('2d');

if (!ctx) {

console.warn('TES chart context not available');

return;

}

if (app.chart) {

app.chart.destroy();

}

try {

app.chart = new Chart(ctx, {

type: 'radar',

data: {

labels: ['Speed', 'Comfort', 'Cost', 'Accessibility', 'Environmental', 'Safety'],

datasets: [{

label: 'London TES Score',

data: app.cityData.london.radar,

backgroundColor: 'rgba(59, 130, 246, 0.2)',

borderColor: 'rgba(59, 130, 246, 1)',

borderWidth: 2,

pointBackgroundColor: 'rgba(59, 130, 246, 1)',

pointBorderColor: '#fff',

pointHoverBackgroundColor: '#fff',

pointHoverBorderColor: 'rgba(59, 130, 246, 1)'

}]

},

options: {

responsive: true,

maintainAspectRatio: false,

scales: {

r: {

beginAtZero: true,

max: 100,

ticks: { display: false },

grid: { color: 'rgba(0, 0, 0, 0.1)' },

angleLines: { color: 'rgba(0, 0, 0, 0.1)' }

}

},

plugins: { legend: { display: false } }

}

});

console.log('✅ TES radar chart initialized successfully');

if (fallback) fallback.style.display = 'none';

canvas.style.display = 'block';

} catch (error) {

console.error('TES chart creation failed:', error);

if (fallback) {

fallback.style.display = 'block';

fallback.innerHTML = `

<div style="text-align: center; padding: 2rem;">

<h3>TES Score Breakdown</h3>

<div style="display: grid; grid-template-columns: repeat(2, 1fr); gap: 1rem; margin-top: 1rem;">

<div><strong>Speed:</strong> ${app.cityData.london.radar[0]}/100</div>

<div><strong>Comfort:</strong> ${app.cityData.london.radar[1]}/100</div>

<div><strong>Cost:</strong> ${app.cityData.london.radar[2]}/100</div>

<div><strong>Accessibility:</strong> ${app.cityData.london.radar[3]}/100</div>

<div><strong>Environmental:</strong> ${app.cityData.london.radar[4]}/100</div>

<div><strong>Safety:</strong> ${app.cityData.london.radar[5]}/100</div>

</div>

</div>

`;

}

canvas.style.display = 'none';

}

}, 500); // Increased delay for better reliability

document.getElementById('currentTESScore').textContent = app.cityData.london.tes;

}

async function loadTFLTrafficData() {

try {

const response = await fetch(`${window.API\_CONFIG.BACKEND\_URL}/api/disruptions`); // Assume backend proxies TfL disruptions

if (!response.ok) throw new Error('TfL API failed');

const data = await response.json();

// Check for disruptions

const hasDisruptions = data.some(line =>

line.lineStatuses.some(status => status.statusSeverity !== 10)

);

if (hasDisruptions) {

showNotification('⚠️ Service disruptions detected on some lines', 'warning');

}

console.log('✅ TfL traffic data loaded');

} catch (e) {

console.warn('TfL traffic data fetch failed:', e);

}

}

async function fetchCrimeData(coords) {

try {

const date = new Date();

date.setMonth(date.getMonth() - 1); // Last month for most recent

const dateStr = date.toISOString().slice(0, 7);

const response = await fetch(`https://data.police.uk/api/crimes-street/all-crime?lat=${coords.lat}&lng=${coords.lng || coords.lon}&date=${dateStr}`);

if (!response.ok) throw new Error('Crime data fetch failed');

const data = await response.json();

return data; // Return full data for patterns/categories

} catch (e) {

console.warn('Crime data fetch failed:', e);

return [];

}

}

async function checkRecentCrimeWarnings(route) {

const endCoords = route.route.legs[0].end\_location;

const crimes = await fetchCrimeData(endCoords);

let warning = false;

let message = '';

crimes.forEach(crime => {

if (crime.category.includes('robbery') || crime.category.includes('violent-crime')) {

// Check if "recent" - since API is lagged, assume last month as "recent" for warning

warning = true;

message += `<li>${crime.category} at ${crime.location.street.name}</li>`;

}

});

if (warning) {

showNotification(`⚠️ Recent crimes near destination: <ul>${message}</ul> Consider alternative route or stay alert.`, 'warning', 10000);

} else if (crimes.length > 5) { // Pattern threshold

showNotification('Crime patterns detected in area. Be cautious.', 'info');

}

// Proxy real-time via X search for breaking news

try {

const endAddress = route.route.legs[0].end\_address.split(',')[0]; // Approximate station

const xResponse = await fetch(`${window.API\_CONFIG.BACKEND\_URL}/api/x-search?query=crime near ${endAddress} London today&since=2025-08-01&limit=5`); // Assume backend proxies X tool

const xData = await xResponse.json();

if (xData.posts && xData.posts.length > 0) {

showNotification('🚨 Possible real-time crime alert near destination from social media. Check local news.', 'error', 10000);

}

} catch (e) {

console.warn('Real-time crime proxy failed:', e);

}

}

function getNormalizedPriorities(preferences) {

const total = Object.values(preferences.priorities).reduce((sum, val) => sum + val, 0);

return Object.fromEntries(

Object.entries(preferences.priorities).map(([key, val]) => [key, val / total])

);

}

async function calculateTESScore(route, preferences) {

const leg = route.legs[0];

const normalized = getNormalizedPriorities(preferences);

let score = 70;

// Duration score (inverse to time, weighted by speed)

const durationMinutes = leg.duration.value / 60;

score += Math.max(0, (60 - durationMinutes) / 60 \* 20) \* normalized.speed;

// Transfer penalty (weighted inverse by ride-quality)

const transfers = Math.max(0, leg.steps.filter(s => s.travel\_mode === 'TRANSIT').length - 1);

score -= transfers \* 5 \* (1 - normalized['ride-quality']);

// Accessibility bonus

score += normalized.accessibility \* 10; // Scaled up for importance

// Environmental bonus (if transit/walk/bike)

if (leg.steps.some(s => s.travel\_mode === 'TRANSIT' || s.travel\_mode === 'WALKING' || s.travel\_mode === 'BICYCLING')) {

score += normalized.environmental \* 15;

}

// Crime data impact (real from Police UK, weighted by safety implied)

const crimes = await fetchCrimeData(leg.end\_location);

const crimePenalty = Math.min(crimes.length \* 0.5, 10);

score -= crimePenalty \* (1 - normalized.accessibility); // Accessibility as proxy for safety weight

// Station amenities (bonus scaled by ride-quality and matches)

let stationScoreTotal = 0;

let stationCount = 0;

leg.steps.forEach(step => {

if (step.travel\_mode === 'TRANSIT' && step.transit) {

const stationName = step.transit.departure\_stop.name.toLowerCase();

const station = app.stations.find(s => s.name.toLowerCase().includes(stationName));

if (station) {

stationScoreTotal += station.score;

stationCount++;

// Bonus for matching amenities, scaled by normalized ride-quality

const matchingAmenities = station.amenities.filter(a => preferences.amenities.includes(a));

score += matchingAmenities.length \* 2 \* normalized['ride-quality'];

}

}

});

if (stationCount > 0) {

score += (stationScoreTotal / stationCount) \* normalized['ride-quality'] \* 10;

}

return Math.min(100, Math.max(0, Math.round(score)));

}

async function findRoutes() {

let from = sanitizeInput(document.getElementById('fromLocation').value);

let to = sanitizeInput(document.getElementById('toLocation').value);

if (!from || !to) {

showNotification('Please enter both starting location and destination.', 'error');

return;

}

console.log(`🔍 Finding routes from "${from}" to "${to}"`);

// Check if Google Maps is available

if (!window.google || !window.google.maps || !app.directionsService) {

console.warn('Google Maps not available, using offline routing');

if (window.findRoutesOffline) {

window.findRoutesOffline();

} else {

enableOfflineRouting();

window.findRoutesOffline();

}

return;

}

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

routeResults.classList.add('show');

app.routesCache = [];

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

routeList.innerHTML = '<div class="skeleton" style="height: 100px;"></div>';

const button = document.getElementById('searchButton');

button.disabled = true;

button.innerHTML = '<i class="fas fa-spinner fa-spin"></i> Calculating...';

try {

const transitModes = getGoogleTransitModes(app.preferences.methods);

console.log('Using transit modes:', transitModes);

const request = {

origin: from,

destination: to,

travelMode: google.maps.TravelMode.TRANSIT,

transitOptions: {

modes: transitModes,

routingPreference: google.maps.TransitRoutePreference.FEWER\_TRANSFERS

},

provideRouteAlternatives: true,

unitSystem: google.maps.UnitSystem.METRIC

};

app.directionsService.route(request, async (result, status) => {

console.log('Directions API response status:', status);

if (status === google.maps.DirectionsStatus.OK) {

console.log(`Found ${result.routes.length} route(s)`);

let routes = await Promise.all(result.routes.map(async (route, index) => {

const tesScore = await calculateTESScore(route, app.preferences);

return {

id: index,

route: route,

score: tesScore

};

}));

// Ensure at least 3 routes (retry with varied prefs if <3)

if (routes.length < 3) {

console.log('Getting additional routes with different preferences...');

// Variant request, e.g., more transfers for alternatives

request.transitOptions.routingPreference = google.maps.TransitRoutePreference.LESS\_WALKING;

const variantResult = await new Promise((res) => app.directionsService.route(request, (r, s) => res(s === 'OK' ? r : null)));

if (variantResult) {

const additional = await Promise.all(variantResult.routes.slice(0, 3 - routes.length).map(async (route, index) => {

const tesScore = await calculateTESScore(route, app.preferences);

return {

id: routes.length + index,

route: route,

score: tesScore

};

}));

routes = [...routes, ...additional];

}

}

routes.sort((a, b) => b.score - a.score);

app.routesCache = routes;

displayRoutes(routes);

if (routes.length > 0) {

showNotification(`Found ${routes.length} optimal routes!`, 'success');

checkDisruptionsForRoutes(routes);

}

} else {

console.error('Directions API failed with status:', status);

let errorMessage = 'No routes found. ';

switch (status) {

case google.maps.DirectionsStatus.NOT\_FOUND:

errorMessage += 'One or both locations could not be found.';

break;

case google.maps.DirectionsStatus.ZERO\_RESULTS:

errorMessage += 'No route could be found between these locations.';

break;

case google.maps.DirectionsStatus.OVER\_QUERY\_LIMIT:

errorMessage += 'Query limit exceeded. Please try again later.';

break;

case google.maps.DirectionsStatus.REQUEST\_DENIED:

errorMessage += 'Request was denied. Check API configuration.';

break;

default:

errorMessage += 'Try different locations or transport modes.';

}

showNotification(errorMessage, 'error');

routeList.innerHTML = '<p>No routes available. Try adjusting your preferences or locations.</p>';

}

button.disabled = false;

button.innerHTML = '<i class="fas fa-search"></i>Find Optimal Routes';

});

} catch (error) {

console.error('Route finding failed:', error);

showNotification('Route planning failed: ' + error.message, 'error');

routeList.innerHTML = '<p>Route planning encountered an error. Please try again.</p>';

button.disabled = false;

button.innerHTML = '<i class="fas fa-search"></i>Find Optimal Routes';

}

}

function getGoogleTransitModes(methods) {

const modeMap = {

'metro': 'SUBWAY',

'bus': 'BUS',

'train': 'TRAIN',

'bike': 'BICYCLE', // Note: Google uses BICYCLING

'walk': 'WALKING'

};

return methods.map(method => modeMap[method] ? google.maps.TravelMode[modeMap[method]] : null).filter(Boolean);

}

async function checkDisruptionsForRoutes(routes) {

const transitLines = new Set();

routes.forEach(routeData => {

routeData.route.legs[0].steps.forEach(step => {

if (step.travel\_mode === 'TRANSIT' && step.transit) {

transitLines.add(step.transit.line.name.toLowerCase());

}

});

});

if (transitLines.size === 0) return;

try {

const response = await fetch(`${window.API\_CONFIG.BACKEND\_URL}/api/disruptions`);

if (!response.ok) throw new Error('TfL Disruptions API failed');

const disruptions = await response.json();

const affectedLines = disruptions.filter(line =>

transitLines.has(line.name.toLowerCase()) &&

line.lineStatuses.some(status => status.statusSeverity !== 10)

);

affectedLines.forEach(line => {

const status = line.lineStatuses.find(s => s.statusSeverity !== 10);

if (status) {

showNotification(`Disruption on ${line.name} Line: ${status.reason}`, 'warning', 8000);

}

});

} catch (error) {

console.error("Failed to check for disruptions:", error);

}

}

function displayRoutes(routes) {

const container = document.getElementById('routeList');

container.innerHTML = '';

routes.forEach((routeData, index) => {

const route = routeData.route;

const leg = route.legs[0];

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

element.className = `route-option ${index === 0 ? 'recommended' : ''}`;

element.innerHTML = `

<div style="display: flex; justify-content: space-between; align-items: center; margin-bottom: 0.75rem;">

<div style="font-weight: 700; font-size: 1.1rem; display: flex; align-items: center; gap: 0.5rem;">

<i class="fas fa-route" style="color: var(--primary-600);"></i> Route ${index + 1}

</div>

<div style="background: var(--gradient-primary); color: white; padding: 0.25rem 0.75rem; border-radius: var(--radius-lg); font-weight: 600;">TES: ${routeData.score}</div>

</div>

<div style="display: grid; grid-template-columns: repeat(auto-fit, minmax(70px, 1fr)); gap: 0.75rem;">

<div style="text-align: center; padding: 0.5rem; background: white; border-radius: var(--radius-md);"><div style="font-weight: 700; color: var(--primary-600);">${leg.duration.text}</div><div style="font-size: 0.8rem; color: var(--gray-600);">Time</div></div>

<div style="text-align: center; padding: 0.5rem; background: white; border-radius: var(--radius-md);"><div style="font-weight: 700; font-size: 1.2rem; color: var(--success-500);">${leg.distance.text}</div><div style="font-size: 0.8rem; color: var(--gray-600);">Distance</div></div>

<div style="text-align: center; padding: 0.5rem; background: white; border-radius: var(--radius-md);"><div style="font-weight: 700; color: var(--secondary-500);">${Math.max(0, leg.steps.filter(s => s.travel\_mode === 'TRANSIT').length - 1)}</div><div style="font-size: 0.8rem; color: var(--gray-600);">Transfers</div></div>

</div>

<div style="margin-top: 0.75rem; font-size: 0.875rem; color: var(--gray-600);"><i class="fas fa-money-bill-wave"></i> Estimated fare: £2.80 (Oyster/Contactless)</div>

<div class="route-actions">

<button class="route-action-btn" onclick='shareRoute(${routeData.id})'><i class="fas fa-share-alt"></i> Share</button>

<button class="route-action-btn" onclick='downloadRouteMap(${routeData.id})'><i class="fas fa-download"></i> Download</button>

</div>

`;

element.setAttribute('role', 'button');

element.setAttribute('tabindex', '0');

element.addEventListener('click', () => selectRoute(routeData, element));

container.appendChild(element);

});

// Ensure route results section is visible

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

routeResults.classList.add('show');

// Force show download buttons

document.querySelectorAll('.map-download-btn').forEach(btn => {

btn.classList.add('show');

console.log('Download button shown');

});

if (routes.length > 0) {

selectRoute(routes[0], container.firstChild);

console.log(`✅ Displayed ${routes.length} routes with download buttons enabled`);

}

}

function selectRoute(routeData, element) {

document.querySelectorAll('.route-option').forEach(el => el.classList.remove('selected'));

element.classList.add('selected');

app.directionsRenderer.setDirections({ routes: [routeData.route] });

app.currentRoute = routeData;

console.log(`🎯 Selected route with TES score: ${routeData.score}`);

// Update city TES with selected route

app.cityData.london.tes = routeData.score;

document.getElementById('currentTESScore').textContent = routeData.score;

document.getElementById('london-tes').textContent = `TES: ${routeData.score}`;

// Check for crime warnings on selection

checkRecentCrimeWarnings(routeData);

}

function switchMapLayer(layer) {

if (app.map instanceof google.maps.Map) {

app.transitLayer.setMap(layer === 'transit' ? app.map : null);

app.trafficLayer.setMap(layer === 'traffic' ? app.map : null);

toggleMarkers(app.bikeMarkers, layer === 'bike');

toggleMarkers(app.stationMarkers, layer === 'accessibility');

} else if (app.map instanceof L.Map) {

// Leaflet layer handling

if (layer === 'bike') {

app.bikeMarkers.forEach(marker => {

if (!app.map.hasLayer(marker)) {

marker.addTo(app.map);

}

});

} else {

app.bikeMarkers.forEach(marker => {

if (app.map.hasLayer(marker)) {

app.map.removeLayer(marker);

}

});

}

}

}

function toggleMarkers(markers, show) {

markers.forEach(marker => {

if (app.map instanceof google.maps.Map) {

marker.setMap(show ? app.map : null);

}

});

}

async function downloadRouteMap(routeId) {

const route = app.routesCache.find(r => r.id === routeId);

if (!route) {

showNotification('Please select a route first.', 'error');

return;

}

showNotification('Route map download in progress...', 'info');

try {

const path = encodeURIComponent(route.route.overview\_polyline.points);

const url = `https://maps.googleapis.com/maps/api/staticmap?size=400x400&path=enc:${path}&key=${window.API\_CONFIG.STATIC\_MAPS\_KEY}`;

const response = await fetch(url);

const blob = await response.blob();

const link = document.createElement('a');

link.href = URL.createObjectURL(blob);

link.download = 'route\_map.png';

link.click();

URL.revokeObjectURL(link.href);

showNotification('Route map downloaded!', 'success');

} catch (e) {

showNotification('Download failed.', 'error');

}

}

async function downloadAreaMap() {

showNotification('Area map download in progress...', 'info');

try {

const center = app.map.getCenter();

const zoom = app.map.getZoom();

const url = `https://maps.googleapis.com/maps/api/staticmap?center=${center.lat()},${center.lng()}&zoom=${zoom}&size=400x400&key=${window.API\_CONFIG.STATIC\_MAPS\_KEY}`;

const response = await fetch(url);

const blob = await response.blob();

const link = document.createElement('a');

link.href = URL.createObjectURL(blob);

link.download = 'area\_map.png';

link.click();

URL.revokeObjectURL(link.href);

showNotification('Area map downloaded!', 'success');

} catch (e) {

showNotification('Download failed.', 'error');

}

}

function shareRoute(routeId) {

const route = app.routesCache.find(r => r.id === routeId);

if (!route) {

showNotification('Please select a route first.', 'error');

return;

}

const from = encodeURIComponent(route.route.legs[0].start\_address);

const to = encodeURIComponent(route.route.legs[0].end\_address);

const shareUrl = `${window.location.origin}${window.location.pathname}?from=${from}&to=${to}`;

if (navigator.share) {

navigator.share({

title: 'My CLUES-TES Route',

text: `Check out my route with TES score ${route.score}!`,

url: shareUrl

}).then(() => showNotification('Route shared!', 'success'))

.catch(() => showNotification('Share cancelled.', 'info'));

} else {

navigator.clipboard.writeText(shareUrl)

.then(() => showNotification('Route link copied to clipboard!', 'success'))

.catch(() => showNotification('Could not copy link.', 'error'));

}

}

function updateSystemStatus() {

const statuses = document.querySelectorAll('.status-indicator');

// This would check actual API status in production

statuses.forEach(status => {

status.classList.remove('loading');

status.classList.add('success');

});

}

function showLoading(show) {

document.getElementById('loadingOverlay').classList.toggle('show', show);

}

// OSM Overpass API for amenities (fallback if backend fails)

async function fetchOSMAmenities(coords) {

try {

const query = `[out:json];(node(around:1500,${coords.lat},${coords.lng || coords.lon})[amenity];);out body;`;

const response = await fetch(`https://overpass-api.de/api/interpreter?data=${encodeURIComponent(query)}`);

const data = await response.json();

if (data.elements.length > 0) {

console.log(`Found ${data.elements.length} nearby amenities from OSM`);

// Add markers for relevant amenities

data.elements.forEach(amenity => {

if (['cafe', 'restaurant', 'toilets', 'atm'].includes(amenity.tags.amenity)) {

if (app.map instanceof google.maps.Map) {

new google.maps.Marker({

position: { lat: amenity.lat, lng: amenity.lon },

map: app.map,

title: amenity.tags.name || amenity.tags.amenity,

icon: {

path: google.maps.SymbolPath.CIRCLE,

scale: 4,

fillColor: '#3b82f6',

fillOpacity: 0.6,

strokeWeight: 1,

strokeColor: '#ffffff'

}

});

}

}

});

}

} catch (e) {

console.warn('OSM Overpass query failed:', e);

}

}

if (typeof module !== 'undefined' && module.exports) {

module.exports = {

loadWeatherData: updateWeatherDisplay,

findRoutes: findRoutes,

fetchBikeData: fetchBikeData,

calculateTESScore: calculateTESScore

};

}

console.log('📱 CLUES-TES Enhanced App loaded successfully');

// Make enableOfflineRouting globally available

window.enableOfflineRouting = enableOfflineRouting;

</script>

</body>

</html>