```
checked;
}
```

```
  function setupMobileUI() {
    // Show/hide mobile filter panel
    document.getElementById('show-filters').addEventListener('click', function() {
      document.getElementById('mobile-filter-panel').classList.remove('hidden');
    });

    document.getElementById('close-filters').addEventListener('click', closeMobileFilters);

    // Show/hide mobile events list
    document.getElementById('show-list').addEventListener('click', function() {
      document.getElementById('mobile-events-list').classList.remove('hidden');
    });

    document.getElementById('close-list').addEventListener('click', function() {
      document.getElementById('mobile-events-list').classList.add('hidden');
    });

    // Close when clicking outside content
    document.getElementById('mobile-filter-panel').addEventListener('click', function(e) {
      if (e.target === this) {
        closeMobileFilters();
      }
    });
  }

  function closeMobileFilters() {
    document.getElementById('mobile-filter-panel').classList.add('hidden');
  }

  function getCategoryIcon(category) {
    switch (category) {
      case 'music': return 'fa-music';
      case 'art': return 'fa-palette';
      case 'food': return 'fa-utensils';
      case 'nightlife': return 'fa-glass-cheers';
      case 'sports': return 'fa-running';
      case 'community': return 'fa-users';
      default: return 'fa-calendar-alt';
    }
  }
```

</script> </body> </html> ```

3.2. Add Location Services Module (Duration: 2 days)

```
javascript
```

```javascript
// Create file: scripts/services/location-service.js
class LocationService {
  constructor() {
    this.geocodeCache = {};
    this.userLocation = null;
  }

  async getUserLocation() {
    // Return cached location if available
    if (this.userLocation) {
      return this.userLocation;
    }

    return new Promise((resolve, reject) => {
      if (!navigator.geolocation) {
        reject(new Error('Geolocation is not supported by your browser'));
        return;
      }

      navigator.geolocation.getCurrentPosition(
        position => {
          const location = {
            latitude: position.coords.latitude,
            longitude: position.coords.longitude,
            accuracy: position.coords.accuracy
          };

          // Cache the location
          this.userLocation = location;

          resolve(location);
        },
        error => {
          console.error('Geolocation error:', error);
          reject(error);
        },
        {
          enableHighAccuracy: true,
          timeout: 5000,
          maximumAge: 0
        }
      );
    });
  }

  async geocodeAddress(address) {
    // Check cache first
    if (this.geocodeCache[address]) {
      return this.geocodeCache[address];
    }

    try {
      // Use Nominatim for geocoding (free and open-source)
      const response = await fetch(`https://nominatim.openstreetmap.org/search?format=json&q=${
      const data = await response.json();

      if (data.length === 0) {
        throw new Error('Location not found');
      }

      const result = {
        latitude: parseFloat(data[0].lat),
        longitude: parseFloat(data[0].lon),
        displayName: data[0].display_name
      };

      // Cache the result
      this.geocodeCache[address] = result;

      return result;
    } catch (error) {
      console.error('Geocoding error:', error);
      throw error;
    }
  }

  async reverseGeocode(latitude, longitude) {
    const cacheKey = `${latitude},${longitude}`;

    // Check cache first
    if (this.geocodeCache[cacheKey]) {
      return this.geocodeCache[cacheKey];
    }

    try {
      // Use Nominatim for reverse geocoding
      const response = await fetch(`https://nominatim.openstreetmap.org/reverse?format=json&lat
    const data = await response.json();

    if (!data || data.error) {
      throw new Error('Unable to reverse geocode coordinates');
    }

    const result = {
      address: data.display_name,
      city: data.address.city || data.address.town || data.address.village || '',
      state: data.address.state || '',
      country: data.address.country || '',
      postcode: data.address.postcode || ''
    };

    // Cache the result
    this.geocodeCache[cacheKey] = result;

    return result;
  } catch (error) {
    console.error('Reverse geocoding error:', error);
    throw error;
  }
}

calculateDistance(lat1, lon1, lat2, lon2) {
  // Haversine formula to calculate distance between two points on Earth
  const R = 6371; // Radius of the earth in km
  const dLat = this.deg2rad(lat2 - lat1);
  const dLon = this.deg2rad(lon2 - lon1);
  const a =
    Math.sin(dLat/2) * Math.sin(dLat/2) +
    Math.cos(this.deg2rad(lat1)) * Math.cos(this.deg2rad(lat2)) *
    Math.sin(dLon/2) * Math.sin(dLon/2);
  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
  const distance = R * c; // Distance in km

  return distance;
}

deg2rad(deg) {
  return deg * (Math.PI/180);
}

async findNearbyEvents(radius = 10) {
```

```
try {
  // Get user location
  const userLocation = await this.getUserLocation();

  // Get all events
  const db = firebase.firestore();
  const eventsSnapshot = await db.collection('events').get();

  // Filter events by distance
  const nearbyEvents = [];

  eventsSnapshot.forEach(doc => {
    const event = {
      id: doc.id,
      ...doc.data()
    };

    // Skip events without coordinates
    if (!event.coordinates) return;

    // Calculate distance
    const distance = this.calculateDistance(
      userLocation.latitude,
      userLocation.longitude,
      event.coordinates.lat,
      event.coordinates.lng
    );

    // Add distance to event
    event.distance = distance;

    // Add to nearby events if within radius
    if (distance <= radius) {
      nearbyEvents.push(event);
    }
  });

  // Sort by distance
  nearbyEvents.sort((a, b) => a.distance - b.distance);

  return nearbyEvents;
} catch (error) {
  console.error('Error finding nearby events:', error);
  throw error;
```

```
    }
  }

  getDirectionsUrl(destination, mode = 'driving') {
    // Generate a Google Maps directions URL
    let destinationParam = '';

    if (typeof destination === 'string') {
      // If destination is an address
      destinationParam = encodeURIComponent(destination);
    } else if (destination.lat && destination.lng) {
      // If destination is coordinates
      destinationParam = `${destination.lat},${destination.lng}`;
    } else {
      throw new Error('Invalid destination format');
    }

    // Get travel mode
    let travelMode = '';
    switch (mode.toLowerCase()) {
      case 'driving':
        travelMode = '&dirflg=d';
        break;
      case 'walking':
        travelMode = '&dirflg=w';
        break;
      case 'transit':
        travelMode = '&dirflg=r';
        break;
      case 'bicycling':
        travelMode = '&dirflg=b';
        break;
    }

    return `https://www.google.com/maps/dir/?api=1&destination=${destinationParam}${travelMode}
  }
}

// Create a singleton instance
const locationService = new LocationService();

export default locationService;
```

3.3. Implement "Events Near Me" Feature (Duration: 1 day)

```
javascript
```

```javascript
// Create file: scripts/components/nearby-events.js
import locationService from '../services/location-service.js';
import { formatEventDate } from '../utils/date-format.js';
import errorHandler from '../utils/error-handler.js';

class NearbyEventsComponent {
  constructor(containerId, options = {}) {
    this.containerId = containerId;
    this.container = document.getElementById(containerId);
    this.options = {
      radius: 10, // km
      limit: 6,
      showDistance: true,
      ...options
    };

    if (!this.container) {
      console.error(`Container with ID ${containerId} not found`);
      return;
    }

    this.initialize();
  }

  async initialize() {
    try {
      // Show loading state
      this.showLoading();

      // Load nearby events
      const events = await this.loadNearbyEvents();

      // Update UI
      this.updateUI(events);
    } catch (error) {
      console.error('Error initializing nearby events component:', error);

      if (error.code === 1) {
        // Permission denied
        this.showGeolocationError('Location permission denied. Please enable location services
      } else {
        this.showError('Failed to load nearby events');
      }
    }
  }

  async loadNearbyEvents() {
    try {
      const events = await locationService.findNearbyEvents(this.options.radius);

      // Limit number of events
      return events.slice(0, this.options.limit);
    } catch (error) {
      console.error('Error loading nearby events:', error);
      throw error;
    }
  }

  updateUI(events) {
    // Clear container
    this.container.innerHTML = '';

    // Check if we have events
    if (events.length === 0) {
      this.showNoEvents();
      return;
    }

    // Add heading
    const heading = document.createElement('h2');
    heading.className = 'text-xl font-bold mb-4';
    heading.textContent = 'Events Near You';
    this.container.appendChild(heading);

    // Create events grid
    const grid = document.createElement('div');
    grid.className = 'grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4';

    // Add events to grid
    events.forEach(event => {
      const eventElement = this.createEventElement(event);
      grid.appendChild(eventElement);
    });

    // Add grid to container
    this.container.appendChild(grid);
    // Add "View All" button if we have more events than the limit
    if (events.length >= this.options.limit) {
      const viewAllBtn = document.createElement('div');
      viewAllBtn.className = 'text-center mt-6';
      viewAllBtn.innerHTML = `
        <a href="map.html" class="btn-primary inline-block">
          View All Nearby Events
        </a>
      `;
      this.container.appendChild(viewAllBtn);
    }
  }

  createEventElement(event) {
    const element = document.createElement('div');
    element.className = 'bg-white rounded-lg shadow-sm overflow-hidden';

    // Format date
    let dateStr = 'Date TBD';
    if (event.dateTime) {
      const dateTime = new Date(event.dateTime.seconds * 1000);
      dateStr = formatEventDate(dateTime);
    }

    // Format distance
    let distanceStr = '';
    if (this.options.showDistance && event.distance) {
      distanceStr = `<span class="ml-2 text-xs bg-gray-100 rounded-full px-2 py-1">${event.dist
    }

    element.innerHTML = `
      <div class="h-36 bg-gray-200 relative">
        <img src="${event.imageUrl || '/api/placeholder/400/250'}" alt="${event.title}" class="
        <div class="absolute top-2 right-2 vibe-score text-xs">
          <i class="fas fa-fire mr-1"></i> ${event.score || '8.5'}
        </div>
      </div>
      <div class="p-3">
        <h3 class="font-bold text-sm mb-1">${event.title}</h3>
        <p class="text-xs text-gray-600 flex items-center mb-1">
          <i class="fas fa-calendar-alt mr-1"></i> ${dateStr}${distanceStr}
        </p>
        <p class="text-xs text-gray-600 truncate">
          <i class="fas fa-map-marker-alt mr-1"></i> ${event.location || 'Location TBD'}
      </p>
    </div>
  `;

  // Add click event
  element.addEventListener('click', () => {
    window.location.href = `event-detail.html?id=${event.id}`;
  });

  return element;
}

showLoading() {
  this.container.innerHTML = `
    <div class="text-center py-8">
      <div class="animate-spin rounded-full h-8 w-8 border-t-2 border-b-2 border-primary mx-a
      <p class="text-gray-600">Finding events near you...</p>
    </div>
  `;
}

showNoEvents() {
  this.container.innerHTML = `
    <div class="text-center py-8">
      <p class="text-gray-600 mb-4">No events found within ${this.options.radius} km of your
      <a href="map.html" class="btn-primary inline-block">
        Explore Events Map
      </a>
    </div>
  `;
}

showGeolocationError(message) {
  this.container.innerHTML = `
    <div class="text-center py-8">
      <p class="text-gray-600 mb-4">${message}</p>
      <a href="map.html" class="btn-primary inline-block">
        View Events Map
      </a>
    </div>
  `;
}

showError(message) {
    this.container.innerHTML = `
      <div class="text-center py-8">
        <p class="text-gray-600 mb-4">${message}</p>
        <button class="btn-primary inline-block" onclick="location.reload()">
          Try Again
        </button>
      </div>
    `;
  }
}

export default NearbyEventsComponent;
```

4. Search and Discovery (Duration: 5 days)

4.1. Create Search Page (Duration: 2 days)

html

```html
<!-- Create file: pages/app/search.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Search Events - The Play</title>
  <!-- Include your standard CSS and JS files here -->
  <link href="https://cdnjs.cloudflare.com/ajax/libs/tailwindcss/2.2.19/tailwind.min.css" rel="
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/
  <link rel="stylesheet" href="../../assets/styles/main.css">
  <link rel="stylesheet" href="../../assets/styles/variables.css">
  <link rel="stylesheet" href="../../assets/styles/layout.css">
  <link rel="stylesheet" href="../../assets/styles/components.css">

  <!-- Firebase Libraries -->
  <script src="https://www.gstatic.com/firebasejs/9.22.2/firebase-app-compat.js"></script>
  <script src="https://www.gstatic.com/firebasejs/9.22.2/firebase-auth-compat.js"></script>
  <script src="https://www.gstatic.com/firebasejs/9.22.2/firebase-firestore-compat.js"></script
  <script src="../../scripts/firebase/config.js"></script>
</head>
<body class="bg-gray-100">
  <!-- Header Container -->
  <div id="header-container"></div>

  <!-- Main Content -->
  <div class="container mx-auto px-4 py-8">
    <div class="max-w-4xl mx-auto">
      <!-- Search Form -->
      <div class="bg-white rounded-xl shadow-md p-6 mb-6">
        <h1 class="text-2xl font-bold mb-6">Search Events</h1>

        <form id="search-form">
          <div class="relative mb-6">
            <input type="text" id="search-input" placeholder="Search events, venues, or categor
            <i class="fas fa-search absolute left-4 top-4 text-gray-400"></i>
            <button type="submit" class="absolute right-4 top-2 btn-primary py-2 px-4">Search</
          </div>

          <div class="grid grid-cols-1 md:grid-cols-3 gap-4">
            <!-- Category Filter -->
            <div>
              <label for="search-category" class="block text-gray-700 text-sm font-medium mb-2"
          <select id="search-category" class="w-full px-4 py-3 rounded-lg border border-gra
            <option value="">All Categories</option>
            <option value="music">Music</option>
            <option value="art">Art & Culture</option>
            <option value="food">Food & Drink</option>
            <option value="nightlife">Nightlife</option>
            <option value="sports">Sports & Fitness</option>
            <option value="community">Community</option>
            <option value="other">Other</option>
          </select>
        </div>

        <!-- Date Filter -->
        <div>
          <label for="search-date" class="block text-gray-700 text-sm font-medium mb-2">Dat
          <select id="search-date" class="w-full px-4 py-3 rounded-lg border border-gray-30
            <option value="">Any Date</option>
            <option value="today">Today</option>
            <option value="tomorrow">Tomorrow</option>
            <option value="this-weekend">This Weekend</option>
            <option value="this-week">This Week</option>
            <option value="next-week">Next Week</option>
            <option value="this-month">This Month</option>
          </select>
        </div>

        <!-- Price Filter -->
        <div>
          <label for="search-price" class="block text-gray-700 text-sm font-medium mb-2">Pr
          <select id="search-price" class="w-full px-4 py-3 rounded-lg border border-gray-3
            <option value="">Any Price</option>
            <option value="free">Free</option>
            <option value="paid">Paid</option>
          </select>
        </div>
      </div>
    </form>
  </div>

  <!-- Search Results -->
  <div id="search-results">
    <!-- Initial state -->
    <div id="initial-state" class="text-center py-12">
      <i class="fas fa-search text-5xl text-gray-300 mb-4"></i>
```

```
        <p class="text-gray-600">Search for events above to see results</p>
      </div>

      <!-- Loading state (hidden by default) -->
      <div id="loading-state" class="text-center py-12 hidden">
        <div class="animate-spin rounded-full h-12 w-12 border-t-2 border-b-2 border-primary
        <p class="text-gray-600">Searching for events...</p>
      </div>

      <!-- No results state (hidden by default) -->
      <div id="no-results" class="text-center py-12 hidden">
        <i class="fas fa-search text-5xl text-gray-300 mb-4"></i>
        <p class="text-gray-600">No events found. Try adjusting your search.</p>
      </div>

      <!-- Results Grid (hidden by default) -->
      <div id="results-grid" class="hidden">
        <h2 class="text-xl font-bold mb-4">Search Results</h2>
        <div id="results-container" class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap
          <!-- Results will be added here -->
        </div>
      </div>
    </div>
  </div>
</div>

<!-- Scripts -->
<script type="module">
  import { loadComponent } from '../../scripts/utils/component-loader.js';
  import eventService from '../../scripts/services/event-service.js';
  import { formatEventDate } from '../../scripts/utils/date-format.js';
  import errorHandler from '../../scripts/utils/error-handler.js';

  document.addEventListener('DOMContentLoaded', function() {
    // Load header component
    loadComponent('header-container', '../../components/header.html');

    // Setup search form
    setupSearchForm();

    // Check if search was passed in URL
    const urlParams = new URLSearchParams(window.location.search);
    const queryParam = urlParams.get('q');
  if (queryParam) {
    // Set the search input value
    document.getElementById('search-input').value = queryParam;

    // Perform search
    performSearch(queryParam);
  }
});

function setupSearchForm() {
  const searchForm = document.getElementById('search-form');

  searchForm.addEventListener('submit', function(e) {
    e.preventDefault();

    const searchInput = document.getElementById('search-input');
    const searchQuery = searchInput.value.trim();

    if (!searchQuery) {
      errorHandler.showToast('Please enter a search term', 'warning');
      return;
    }

    performSearch();
  });
}

async function performSearch() {
  // Show loading state
  showLoadingState();

  // Get search parameters
  const searchQuery = document.getElementById('search-input').value.trim();
  const categoryFilter = document.getElementById('search-category').value;
  const dateFilter = document.getElementById('search-date').value;
  const priceFilter = document.getElementById('search-price').value;

  try {
    // Get all events (we'll implement server-side search in a future update)
    const allEvents = await eventService.getEvents({}, 100);

    // Client-side filtering for now
    const filteredEvents = filterEvents(allEvents, {
      query: searchQuery,
      category: categoryFilter,
      date: dateFilter,
      price: priceFilter
    });

    // Display results
    displaySearchResults(filteredEvents);

    // Update URL with search parameters
    updateSearchURL(searchQuery, categoryFilter, dateFilter, priceFilter);
  } catch (error) {
    console.error('Search error:', error);
    errorHandler.showToast('Error searching events', 'error');
    showNoResults();
  }
}

function filterEvents(events, filters) {
  return events.filter(event => {
    // Text search
    if (filters.query) {
      const query = filters.query.toLowerCase();
      const title = (event.title || '').toLowerCase();
      const description = (event.description || '').toLowerCase();
      const location = (event.location || '').toLowerCase();

      // If none of the fields match the query, exclude this event
      if (!title.includes(query) && !description.includes(query) && !location.includes(quer
        return false;
      }
    }

    // Category filter
    if (filters.category && event.category !== filters.category) {
      return false;
    }

    // Date filter
    if (filters.date) {
      const eventDate = event.dateTime ? new Date(event.dateTime.seconds * 1000) : null;

      if (!eventDate) {
        return false;
      }
```

```
const today = new Date();
today.setHours(0, 0, 0, 0);

const tomorrow = new Date(today);
tomorrow.setDate(tomorrow.getDate() + 1);

// Get day of week (0 = Sunday, 6 = Saturday)
const currentDay = today.getDay();

// Calculate days until weekend
const daysUntilSaturday = 6 - currentDay;

// Calculate weekend range
const saturday = new Date(today);
saturday.setDate(today.getDate() + daysUntilSaturday);

const sunday = new Date(saturday);
sunday.setDate(saturday.getDate() + 1);

// Calculate week range
const endOfWeek = new Date(today);
endOfWeek.setDate(today.getDate() + (7 - currentDay));

const startOfNextWeek = new Date(endOfWeek);
startOfNextWeek.setDate(endOfWeek.getDate() + 1);

const endOfNextWeek = new Date(startOfNextWeek);
endOfNextWeek.setDate(startOfNextWeek.getDate() + 6);

// Calculate month range
const endOfMonth = new Date(today.getFullYear(), today.getMonth() + 1, 0);

switch (filters.date) {
  case 'today':
    // Event date must be today
    if (eventDate < today || eventDate >= tomorrow) {
      return false;
    }
    break;

  case 'tomorrow':
    // Event date must be tomorrow
    const dayAfterTomorrow = new Date(tomorrow);
```

```
        dayAfterTomorrow.setDate(dayAfterTomorrow.getDate() + 1);

        if (eventDate < tomorrow || eventDate >= dayAfterTomorrow) {
          return false;
        }
        break;

      case 'this-weekend':
        // Event date must be this weekend (Saturday or Sunday)
        if (eventDate < saturday || eventDate >= new Date(sunday.setDate(sunday.getDate()
          return false;
        }
        break;

      case 'this-week':
        // Event date must be within the current week
        if (eventDate < today || eventDate > endOfWeek) {
          return false;
        }
        break;

      case 'next-week':
        // Event date must be within next week
        if (eventDate < startOfNextWeek || eventDate > endOfNextWeek) {
          return false;
        }
        break;

      case 'this-month':
        // Event date must be within the current month
        if (eventDate < today || eventDate > endOfMonth) {
          return false;
        }
        break;
    }
  }

  // Price filter
  if (filters.price) {
    const isFree = !event.price || event.price === 'Free' || event.price === '$0';

    if (filters.price === 'free' && !isFree) {
      return false;
    }
```

```
      if (filters.price === 'paid' && isFree) {
        return false;
      }
    }

    // All filters passed, include this event
    return true;
  });
}

function displaySearchResults(events) {
  const initialState = document.getElementById('initial-state');
  const loadingState = document.getElementById('loading-state');
  const noResults = document.getElementById('no-results');
  const resultsGrid = document.getElementById('results-grid');
  const resultsContainer = document.getElementById('results-container');

  // Hide all states
  initialState.classList.add('hidden');
  loadingState.classList.add('hidden');
  noResults.classList.add('hidden');
  resultsGrid.classList.add('hidden');

  // Clear results container
  resultsContainer.innerHTML = '';

  // Check if we have results
  if (events.length === 0) {
    noResults.classList.remove('hidden');
    return;
  }

  // Add events to results container
  events.forEach(event => {
    const eventElement = createEventElement(event);
    resultsContainer.appendChild(eventElement);
  });

  // Show results grid
  resultsGrid.classList.remove('hidden');
}

function createEventElement(event) {
```

```
const element = document.createElement('div');
element.className = 'bg-white rounded-lg shadow-sm overflow-hidden';

// Format date
let dateStr = 'Date TBD';
if (event.dateTime) {
  const dateTime = new Date(event.dateTime.seconds * 1000);
  dateStr = formatEventDate(dateTime);
}

element.innerHTML = `
  <div class="h-36 bg-gray-200 relative">
    <img src="${event.imageUrl || '/api/placeholder/400/250'}" alt="${event.title}" class
    <div class="absolute top-2 right-2 vibe-score text-xs">
      <i class="fas fa-fire mr-1"></i> ${event.score || '8.5'}
    </div>
  </div>
  <div class="p-3">
    <h3 class="font-bold text-sm mb-1">${event.title}</h3>
    <p class="text-xs text-gray-600 mb-1">
      <i class="fas fa-calendar-alt mr-1"></i> ${dateStr}
    </p>
    <p class="text-xs text-gray-600 truncate">
      <i class="fas fa-map-marker-alt mr-1"></i> ${event.location || 'Location TBD'}
    </p>
  </div>
```