COMPLETE IMPLEMENTATION GUIDE FOR RENTAL SEARCH SYSTEM

PART 2: CONTROLLERS & API ENDPOINTS

1. Set up SearchController:

- Implement search functionality with filters
- Handle location-based searching
- Manage price range filtering
- Process feature-based filtering
- Handle availability checks

2. Create API Endpoints:

- /api/vehicles/search (main search endpoint)
- /api/vehicles/{id} (single vehicle details)
- /api/vehicles/availability (availability checking)
- /api/locations/autocomplete (location search)

3. Implement Response Resources:

- Create VehicleResource for consistent responses
- Include necessary vehicle details
- Handle image URLs
- Include availability information

PART 3: VUE COMPONENTS & FRONTEND SETUP

1. Create Base Components:

- SearchHeader component
- VehicleCard component
- FilterContainer component
- StarRating component
- Modal component
- LoadingSpinner component

2. Set up State Management:

- Create filter store
- Implement vehicle store
- Handle search parameters
- Manage UI state

3. Implement Layout Components:

- Split view layout
- Responsive sidebar
- Results grid/list views
- Mobile adaptations

PART 4: MAP INTEGRATION & MARKER CLUSTERING

1. Set up Map Configuration:

- Initialize Leaflet map
- Configure map settings
- Set up default views
- Handle map events

2. Implement Marker Clustering:

- Set up marker cluster plugin
- Configure cluster appearance
- Handle cluster events
- Optimize cluster performance

3. Create Custom Markers:

- Design price tooltips
- Implement info windows
- Handle marker interactions
- Create custom cluster icons

PART 5: SEARCH FILTERS & PRICE RANGE COMPONENT

1. Implement Core Filters:

- Price range slider
- Feature checkboxes
- Vehicle type selector
- Rating filter

- Amenities filter

2. Create Filter Logic:

- Handle filter combinations
- Implement filter dependencies
- Manage filter state
- Handle URL parameters

3. Add Filter Features:

- Clear all functionality
- Save filter preferences
- Filter analytics
- Mobile filter drawer

PART 6: AVAILABILITY CALENDAR INTEGRATION

1. Set up Calendar Component:

- Create date picker
- Implement range selection
- Show unavailable dates
- Handle booking restrictions

2. Implement Availability Logic:

- Check date ranges
- Handle blocked dates
- Process booking rules
- Manage conflicts

3. Add Calendar Features:

- Visual date highlights
- Price calculations
- Minimum stay rules
- Season handling

Additional Considerations for Each Part:

1. Error Handling:

- Implement try-catch blocks
- Show user-friendly errors
- Handle network issues
- Log important errors

2. Performance:

- Implement caching
- Optimize database queries
- Use lazy loading
- Handle large datasets

3. Mobile Responsiveness:

- Adapt all components
- Handle touch events
- Optimize for small screens
- Test on various devices

4. Testing:

- Unit tests for components
- Integration tests
- API endpoint tests
- Browser testing

5. Documentation:

- API documentation
- Component documentation
- Setup instructions
- Deployment guide

Deployment Checklist:

1. Environment Setup:

- Configure environment variables
- Set up API keys
- Configure services
- Set up monitoring

2. Build Process:

- Optimize assets
- Minify code
- Configure CDN
- Set up caching

3. Security:

- Implement CSRF protection
- Set up API authentication
- Configure CORS
- Secure sensitive data

PART 1: DATABASE SETUP

1. First, create the database migrations:

```
// database/migrations/2024_01_03_create_vehicles_table.php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up()
    {
        Schema::create('vehicles', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->text('description');
            $table->decimal('price_per_day', 10, 2);
            $table->string('location');
            $table->decimal('latitude', 10, 8);
            $table->decimal('longitude', 11, 8);
            $table->json('images');
            $table->integer('rating')->default(0);
            $table->integer('reviews_count')->default(0);
            $table->json('features');
            $table->json('availability_calendar')->nullable();
            $table->decimal('min_price', 10, 2);
            $table->decimal('max_price', 10, 2);
            $table->string('vehicle_type');
            $table->boolean('is_available')->default(true);
            $table->timestamps();

            // Add spatial index
            $table->spatialIndex(['latitude', 'longitude']);
        });

        // Create table for vehicle unavailable dates
        Schema::create('vehicle_unavailable_dates', function (Blueprint $table) {
            $table->id();
            $table->foreignId('vehicle_id')->constrained()->onDelete('cascade');
            $table->date('start_date');
            $table->date('end_date');
            $table->string('reason')->nullable();
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('vehicle_unavailable_dates');
        Schema::dropIfExists('vehicles');
    }
};
```

2. Create the Vehicle Model:

```php
// app/Models/Vehicle.php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Vehicle extends Model
{
    protected $casts = [
        'images' => 'array',
        'features' => 'array',
        'availability_calendar' => 'array',
    ];

    protected $appends = [
        'thumbnail',
        'feature_list',
    ];

    // Relationships
    public function unavailableDates(): HasMany
    {
        return $this->hasMany(VehicleUnavailableDate::class);
    }

    // Accessors
    public function getThumbnailAttribute()
    {
        return $this->images[0] ?? null;
    }

    public function getFeatureListAttribute()
    {
        return collect($this->features)->map(function ($feature) {
            return [
                'name' => $feature,
                'icon' => $this->getFeatureIcon($feature)
            ];
        });
    }

    // Scopes
    public function scopeAvailableInDateRange($query, $startDate, $endDate)
    {
        return $query->whereDoesntHave('unavailableDates', function ($query) use ($startDate, $endDate) {
            $query->where(function ($q) use ($startDate, $endDate) {
                $q->whereBetween('start_date', [$startDate, $endDate])
                    ->orWhereBetween('end_date', [$startDate, $endDate])
                    ->orWhere(function ($q) use ($startDate, $endDate) {
                        $q->where('start_date', '<=', $startDate)
                            ->where('end_date', '>=', $endDate);
                    });
            });
        });
    }

    public function scopeInPriceRange($query, $minPrice, $maxPrice)
    {
        return $query->whereBetween('price_per_day', [$minPrice, $maxPrice]);
    }

    public function scopeWithFeatures($query, array $features)
    {
```

```
        return $query->where(function ($query) use ($features) {
            foreach ($features as $feature) {
                $query->whereJsonContains('features', $feature);
            }
        });
    }

    // Helper methods
    private function getFeatureIcon($feature)
    {
        // Map features to their corresponding icons
        $iconMap = [
            'AC' => 'air-conditioner',
            'GPS' => 'navigation',
            // Add more feature-to-icon mappings
        ];

        return $iconMap[$feature] ?? 'default-feature';
    }
}
```

3. Create the UnavailableDate Model:

```
// app/Models/VehicleUnavailableDate.php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class VehicleUnavailableDate extends Model
{
    protected $fillable = [
        'vehicle_id',
        'start_date',
        'end_date',
        'reason'
    ];

    protected $casts = [
        'start_date' => 'date',
        'end_date' => 'date'
    ];

    public function vehicle()
    {
        return $this->belongsTo(Vehicle::class);
    }
}
```

PART 2: CONTROLLERS & API ENDPOINTS

1. First, create the SearchController:

```php
// app/Http/Controllers/SearchController.php
<?php

namespace App\Http\Controllers;

use App\Models\Vehicle;
use Illuminate\Http\Request;
use Inertia\Inertia;

class SearchController extends Controller
{
    public function index(Request $request)
    {
        // Validate request
        $validated = $request->validate([
            'location' => 'nullable|string',
            'start_date' => 'nullable|date',
            'end_date' => 'nullable|date|after:start_date',
            'min_price' => 'nullable|numeric',
            'max_price' => 'nullable|numeric|gt:min_price',
            'features' => 'nullable|array',
            'vehicle_type' => 'nullable|string'
        ]);

        // Query vehicles
        $vehicles = Vehicle::query()
            ->when($request->location, function($query, $location) {
                return $query->where('location', 'like', "%{$location}%");
            })
            ->when($request->min_price && $request->max_price, function($query) use ($request) {
                return $query->inPriceRange($request->min_price, $request->max_price);
            })
            ->when($request->features, function($query, $features) {
                return $query->withFeatures($features);
            })
            ->when($request->start_date && $request->end_date, function($query) use ($request) {
                return $query->availableInDateRange($request->start_date, $request->end_date);
            })
            ->when($request->vehicle_type, function($query, $type) {
                return $query->where('vehicle_type', $type);
            })
            ->get();

        // Get price range for filters
        $priceRange = Vehicle::selectRaw('MIN(price_per_day) as min_price, MAX(price_per_day) as max_price')
            ->first();

        // Get all available features for filters
        $allFeatures = Vehicle::select('features')
            ->get()
            ->pluck('features')
            ->flatten()
            ->unique()
            ->values();

        return Inertia::render('Search/Index', [
            'vehicles' => $vehicles,
            'filters' => $request->all(),
            'priceRange' => $priceRange,
            'availableFeatures' => $allFeatures,
        ]);
    }
}
```

2. Create the VehicleController for single vehicle view:

```
// app/Http/Controllers/VehicleController.php
<?php

namespace App\Http\Controllers;

use App\Models\Vehicle;
use Illuminate\Http\Request;
use Inertia\Inertia;

class VehicleController extends Controller
{
    public function show(Vehicle $vehicle, Request $request)
    {
        // Load unavailable dates
        $vehicle->load('unavailableDates');

        // Get similar vehicles
        $similarVehicles = Vehicle::where('id', '!=', $vehicle->id)
            ->where('location', $vehicle->location)
            ->take(3)
            ->get();

        return Inertia::render('Vehicles/Show', [
            'vehicle' => $vehicle,
            'similarVehicles' => $similarVehicles,
            'searchParams' => $request->only(['start_date', 'end_date', 'location'])
        ]);
    }

    public function availability(Vehicle $vehicle, Request $request)
    {
        $unavailableDates = $vehicle->unavailableDates()
            ->where('end_date', '>=', now())
            ->get()
            ->map(function ($date) {
                return [
                    'start' => $date->start_date->format('Y-m-d'),
                    'end' => $date->end_date->format('Y-m-d')
                ];
            });

        return response()->json($unavailableDates);
    }
}
```

3. Create API routes:

```
// routes/web.php
use App\Http\Controllers\SearchController;
use App\Http\Controllers\VehicleController;

Route::get('/search', [SearchController::class, 'index'])->name('search');
Route::get('/vehicles/{vehicle}', [VehicleController::class, 'show'])->name('vehicles.show');
Route::get('/api/vehicles/{vehicle}/availability', [VehicleController::class, 'availability'])->name('vehicles.availability');
```

4. Create a Resource class for consistent API responses:

```php
// app/Http/Resources/VehicleResource.php
<?php

namespace App\Http\Resources;

use Illuminate\Http\Resources\Json\JsonResource;

class VehicleResource extends JsonResource
{
    public function toArray($request)
    {
        return [
            'id' => $this->id,
            'title' => $this->title,
            'description' => $this->description,
            'price_per_day' => $this->price_per_day,
            'location' => $this->location,
            'latitude' => $this->latitude,
            'longitude' => $this->longitude,
            'images' => $this->images,
            'thumbnail' => $this->thumbnail,
            'rating' => $this->rating,
            'reviews_count' => $this->reviews_count,
            'features' => $this->feature_list,
            'vehicle_type' => $this->vehicle_type,
            'is_available' => $this->is_available,
            'unavailable_dates' => $this->when($this->relationLoaded('unavailableDates'),
                $this->unavailableDates->map(function($date) {
                    return [
                        'start' => $date->start_date->format('Y-m-d'),
                        'end' => $date->end_date->format('Y-m-d')
                    ];
                })
            )
        ];
    }
}
```

5. Create a service class for geocoding:

```php
// app/Services/GeocodingService.php
<?php

namespace App\Services;

use Illuminate\Support\Facades\Http;

class GeocodingService
{
    public function geocodeAddress($address)
    {
        $response = Http::get('https://api.stadiamaps.com/geocoding/v1/search', [
            'api_key' => config('services.stadia.key'),
            'q' => $address
        ]);

        if ($response->successful()) {
            $data = $response->json();
            if (!empty($data['features'])) {
                $coordinates = $data['features'][0]['geometry']['coordinates'];
                return [
                    'latitude' => $coordinates[1],
                    'longitude' => $coordinates[0]
                ];
            }
        }

        return null;
    }
}
```

PART 3: VUE COMPONENTS & FRONTEND SETUP

1. First, let's create the main Search Page component:

```vue
// resources/js/Pages/Search/Index.vue
<template>
  <div class="flex flex-col h-screen">
    <!-- Header with search controls -->
    <SearchHeader
      v-model:location="searchFilters.location"
      v-model:startDate="searchFilters.start_date"
      v-model:endDate="searchFilters.end_date"
      @search="performSearch"
    />

    <!-- Main content -->
    <div class="flex flex-1 overflow-hidden">
      <!-- Sidebar with filters and listings -->
      <div class="w-1/3 flex flex-col border-r">
        <!-- Filters section -->
        <div class="p-4 border-b">
          <PriceRangeFilter
            v-model:min="searchFilters.min_price"
            v-model:max="searchFilters.max_price"
            :range="priceRange"
            @update:range="performSearch"
          />

          <FeatureFilter
            v-model:selected="searchFilters.features"
            :features="availableFeatures"
            @update:features="performSearch"
          />
        </div>
```

```
        <!-- Vehicle listings -->
        <div class="flex-1 overflow-y-auto">
          <div v-if="isLoading" class="p-4">
            <LoadingSpinner />
          </div>

          <div v-else class="p-4 space-y-4">
            <VehicleCard
              v-for="vehicle in vehicles"
              :key="vehicle.id"
              :vehicle="vehicle"
              :is-selected="selectedVehicle?.id === vehicle.id"
              @click="selectVehicle(vehicle)"
            />
          </div>
        </div>
      </div>

      <!-- Map section -->
      <div class="w-2/3 relative">
        <VehicleMap
          :vehicles="vehicles"
          :selected-vehicle="selectedVehicle"
          :center="mapCenter"
          :zoom="mapZoom"
          @marker-click="selectVehicle"
        />
      </div>
    </div>

    <!-- Vehicle preview modal -->
    <VehiclePreviewModal
      v-if="showPreview"
      :vehicle="selectedVehicle"
      :search-params="searchFilters"
      @close="showPreview = false"
    />
  </div>
</template>

<script setup>
import { ref, computed, watch } from 'vue'
import { router } from '@inertiajs/vue3'
import debounce from 'lodash/debounce'
import SearchHeader from './components/SearchHeader.vue'
import PriceRangeFilter from './components/PriceRangeFilter.vue'
import FeatureFilter from './components/FeatureFilter.vue'
import VehicleCard from './components/VehicleCard.vue'
import VehicleMap from './components/VehicleMap.vue'
import VehiclePreviewModal from './components/VehiclePreviewModal.vue'
import LoadingSpinner from '@/Components/LoadingSpinner.vue'

// Props
const props = defineProps({
  vehicles: {
    type: Array,
    required: true
  },
  filters: {
    type: Object,
    default: () => ({})
  },
  priceRange: {
    type: Object,
    required: true
```

```
  },
  availableFeatures: {
    type: Array,
    required: true
  }
})

// State
const isLoading = ref(false)
const selectedVehicle = ref(null)
const showPreview = ref(false)
const mapZoom = ref(12)
const mapCenter = ref([25.2048, 55.2708]) // Default to Dubai
const searchFilters = ref({
  location: props.filters.location || '',
  start_date: props.filters.start_date || '',
  end_date: props.filters.end_date || '',
  min_price: props.filters.min_price || props.priceRange.min_price,
  max_price: props.filters.max_price || props.priceRange.max_price,
  features: props.filters.features || [],
})

// Methods
const performSearch = debounce(async () => {
  isLoading.value = true

  router.get(route('search'), searchFilters.value, {
    preserveState: true,
    preserveScroll: true,
    onSuccess: () => {
      isLoading.value = false
    },
    onError: () => {
      isLoading.value = false
    }
  })
}, 500)

const selectVehicle = (vehicle) => {
  selectedVehicle.value = vehicle
  showPreview.value = true
  mapCenter.value = [vehicle.latitude, vehicle.longitude]
  mapZoom.value = 14
}

// Update URL when filters change
watch(searchFilters, () => {
  router.get(
    route('search'),
    { ...searchFilters.value },
    { preserveState: true, replace: true }
  )
}, { deep: true })

// Initialize map center based on search location
onMounted(async () => {
  if (searchFilters.value.location) {
    try {
      const response = await fetch(
        `https://api.stadiamaps.com/geocoding/v1/search?` +
        `api_key=${import.meta.env.VITE_STADIA_API_KEY}&` +
        `q=${encodeURIComponent(searchFilters.value.location)}`
      )
      const data = await response.json()
```

```
      if (data.features?.[0]) {
        const [lng, lat] = data.features[0].geometry.coordinates
        mapCenter.value = [lat, lng]
      }
    } catch (error) {
      console.error('Geocoding error:', error)
    }
  }
})
</script>
```

2. Create the SearchHeader component:

```
<!-- resources/js/Pages/Search/components/SearchHeader.vue -->
<template>
  <div class="bg-white shadow-sm p-4">
    <div class="container mx-auto flex items-center gap-4">
      <!-- Location Search -->
      <div class="flex-1">
        <input
          type="text"
          v-model="locationInput"
          placeholder="Enter location"
          class="w-full p-2 border rounded"
          @input="handleLocationSearch"
        />
        <!-- Location suggestions dropdown -->
        <div v-if="showSuggestions" class="relative">
          <div class="absolute top-0 left-0 w-full bg-white border rounded shadow-lg mt-1">
            <div
              v-for="suggestion in locationSuggestions"
              :key="suggestion.id"
              class="p-2 hover:bg-gray-100 cursor-pointer"
              @click="selectLocation(suggestion)"
            >
              {{ suggestion.name }}
            </div>
          </div>
        </div>
      </div>

      <!-- Date Range Picker -->
      <div class="flex gap-2">
        <DatePicker
          v-model="startDateInput"
          placeholder="Start Date"
          :min-date="new Date()"
          @update:model-value="emitDates"
        />
        <DatePicker
          v-model="endDateInput"
          placeholder="End Date"
          :min-date="startDateInput || new Date()"
          @update:model-value="emitDates"
        />
      </div>

      <!-- Search Button -->
      <button
        class="bg-blue-500 text-white px-6 py-2 rounded hover:bg-blue-600"
        @click="$emit('search')"
      >
        Search
      </button>
    </div>
  </div>
```

```
  </div>
</template>

<script setup>
import { ref, watch } from 'vue'
import debounce from 'lodash/debounce'
import DatePicker from '@/Components/DatePicker.vue'

// Props and emits
const props = defineProps({
  location: String,
  startDate: String,
  endDate: String
})

const emit = defineEmits([
  'update:location',
  'update:startDate',
  'update:endDate',
  'search'
])

// State
const locationInput = ref(props.location || '')
const startDateInput = ref(props.startDate || '')
const endDateInput = ref(props.endDate || '')
const locationSuggestions = ref([])
const showSuggestions = ref(false)

// Methods
const handleLocationSearch = debounce(async () => {
  if (!locationInput.value) return

  try {
    const response = await fetch(
      `https://api.stadiamaps.com/geocoding/v1/autocomplete?` +
      `api_key=${import.meta.env.VITE_STADIA_API_KEY}&` +
      `q=${encodeURIComponent(locationInput.value)}`
    )
    const data = await response.json()

    locationSuggestions.value = data.features.map(feature => ({
      id: feature.id,
      name: feature.properties.name,
      coordinates: feature.geometry.coordinates
    }))

    showSuggestions.value = true
  } catch (error) {
    console.error('Location search error:', error)
  }
}, 300)

const selectLocation = (suggestion) => {
  locationInput.value = suggestion.name
  emit('update:location', suggestion.name)
  showSuggestions.value = false
}

const emitDates = () => {
  emit('update:startDate', startDateInput.value)
  emit('update:endDate', endDateInput.value)
}

// Watch for changes in inputs
watch(locationInput, (newValue) => {
```

```
  emit('update:location', newValue)
})
</script>
```

3. Create the PriceRangeFilter component:

```
<!-- resources/js/Pages/Search/components/PriceRangeFilter.vue -->
<template>
  <div class="price-range-filter">
    <h3 class="font-bold mb-2">Price Range</h3>

    <div class="flex items-center gap-4 mb-4">
      <div class="flex-1">
        <label class="text-sm text-gray-600">Min Price</label>
        <input
          type="number"
          v-model.number="localMin"
          :min="range.min_price"
          :max="localMax"
          class="w-full p-2 border rounded"
        />
      </div>

      <div class="flex-1">
        <label class="text-sm text-gray-600">Max Price</label>
        <input
          type="number"
          v-model.number="localMax"
          :min="localMin"
          :max="range.max_price"
          class="w-full p-2 border rounded"
        />
      </div>
    </div>

    <!-- Range slider -->
    <div class="relative h-2 bg-gray-200 rounded-full">
      <div
        class="absolute h-full bg-blue-500 rounded-full"
        :style="{
          left: `${((localMin - range.min_price) / (range.max_price - range.min_price)) * 100}%`,
          right: `${100 - ((localMax - range.min_price) / (range.max_price - range.min_price)) * 100}%`
        }"
      ></div>
    </div>
  </div>
</template>

<script setup>
import { ref, watch } from 'vue'
import debounce from 'lodash/debounce'

const props = defineProps({
  min: {
    type: Number,
    required: true
  },
  max: {
    type: Number,
    required: true
  },
  range: {
    type: Object,
    required: true
  }
```

```
})

const emit = defineEmits(['update:min', 'update:max', 'update:range'])

const localMin = ref(props.min)
const localMax = ref(props.max)

const emitChanges = debounce(() => {
  emit('update:min', localMin.value)
  emit('update:max', localMax.value)
  emit('update:range')
}, 300)

watch([localMin, localMax], () => {
  emitChanges()
})
</script>
```

4. Let's create the VehicleMap component with marker clustering:

```
<!-- resources/js/Pages/Search/components/VehicleMap.vue -->
<template>
  <div class="h-full">
    <l-map
      v-model:zoom="zoom"
      :center="center"
      :use-global-leaflet="false"
      @ready="handleMapReady"
    >
      <!-- Base map layer -->
      <l-tile-layer
        url="https://tiles.stadiamaps.com/tiles/osm_bright/{z}/{x}/{y}{r}.png"
        :attribution="attribution"
      />

      <!-- Marker cluster group -->
      <l-marker-cluster :options="clusterOptions">
        <l-marker
          v-for="vehicle in vehicles"
          :key="vehicle.id"
          :lat-lng="[vehicle.latitude, vehicle.longitude]"
          @click="handleMarkerClick(vehicle)"
        >
          <!-- Price tooltip -->
          <l-tooltip :permanent="true" class="custom-price-tooltip">
            €{{ vehicle.price_per_day }}/day
          </l-tooltip>

          <!-- Detailed popup -->
          <l-popup :options="popupOptions">
            <div class="vehicle-popup">
              <img
                :src="vehicle.thumbnail"
                :alt="vehicle.title"
                class="w-full h-32 object-cover rounded-t"
              />
              <div class="p-3">
                <h3 class="font-bold text-lg">{{ vehicle.title }}</h3>
                <div class="flex items-center mt-1">
                  <StarRating :rating="vehicle.rating" />
                  <span class="text-sm text-gray-600 ml-2">
                    ({{ vehicle.reviews_count }} reviews)
                  </span>
                </div>
                <p class="text-lg font-bold mt-2">
                  €{{ vehicle.price_per_day }}/day
```

```
                  ₹{{ vehicle.price_per_day }}/day
                </p>
                <div class="flex flex-wrap gap-2 mt-2">
                  <span
                    v-for="feature in vehicle.features.slice(0, 3)"
                    :key="feature.name"
                    class="text-xs bg-gray-100 px-2 py-1 rounded"
                  >
                    {{ feature.name }}
                  </span>
                </div>
                <button
                  @click="$emit('marker-click', vehicle)"
                  class="w-full mt-3 bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-600 transition"
                >
                  View Details
                </button>
              </div>
            </div>
          </l-popup>
        </l-marker>
      </l-marker-cluster>
    </l-map>
  </div>
</template>

<script setup>
import { ref, watch, onMounted } from 'vue'
import {
  LMap,
  LTileLayer,
  LMarker,
  LTooltip,
  LPopup
} from '@vue-leaflet/vue-leaflet'
import 'leaflet.markercluster/dist/MarkerCluster.css'
import 'leaflet.markercluster/dist/MarkerCluster.Default.css'
import StarRating from '@/Components/StarRating.vue'

// Props
const props = defineProps({
  vehicles: {
    type: Array,
    required: true
  },
  selectedVehicle: {
    type: Object,
    default: null
  },
  center: {
    type: Array,
    required: true
  },
  zoom: {
    type: Number,
    default: 12
  }
})

// Emits
const emit = defineEmits(['marker-click'])

// State
const map = ref(null)
const attribution = ref('&copy; <a href="https://stadiamaps.com/">Stadia Maps</a>')
```

```
// Cluster options
const clusterOptions = {
  maxClusterRadius: 50,
  spiderfyOnMaxZoom: true,
  showCoverageOnHover: false,
  zoomToBoundsOnClick: true,
  disableClusteringAtZoom: 15,
  chunkedLoading: true,
  iconCreateFunction: function(cluster) {
    const count = cluster.getChildCount()
    let size = 'small'

    if (count > 50) size = 'large'
    else if (count > 10) size = 'medium'

    return L.divIcon({
      html: `<div class="cluster-icon cluster-${size}">${count}</div>`,
      className: 'custom-cluster-icon',
      iconSize: L.point(40, 40)
    })
  }
}

// Popup options
const popupOptions = {
  maxWidth: 300,
  className: 'vehicle-popup-container'
}

// Methods
const handleMapReady = (mapInstance) => {
  map.value = mapInstance
}

const handleMarkerClick = (vehicle) => {
  emit('marker-click', vehicle)
}

// Watch for selected vehicle changes
watch(() => props.selectedVehicle, (newVehicle) => {
  if (newVehicle && map.value) {
    map.value.setView(
      [newVehicle.latitude, newVehicle.longitude],
      15,
      { animate: true }
    )
  }
}, { deep: true })

// Initialize map when component mounts
onMounted(() => {
  // Fix Leaflet default icon paths
  delete L.Icon.Default.prototype._getIconUrl
  L.Icon.Default.mergeOptions({
    iconRetinaUrl: '/images/marker-icon-2x.png',
    iconUrl: '/images/marker-icon.png',
    shadowUrl: '/images/marker-shadow.png'
  })
})
</script>

<style>
.custom-price-tooltip {
  background: white;
  border: 2px solid #3b82f6;
```

```
  padding: 4px 8px;
  border-radius: 4px;
  font-weight: bold;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.custom-cluster-icon {
  background: white;
  border: 2px solid #3b82f6;
  display: flex;
  align-items: center;
  justify-content: center;
  border-radius: 50%;
  font-weight: bold;
  color: #3b82f6;
}

.cluster-small {
  width: 30px;
  height: 30px;
  font-size: 12px;
}

.cluster-medium {
  width: 35px;
  height: 35px;
  font-size: 14px;
}

.cluster-large {
  width: 40px;
  height: 40px;
  font-size: 16px;
}

.vehicle-popup-container .leaflet-popup-content-wrapper {
  padding: 0;
  overflow: hidden;
  border-radius: 8px;
}

.vehicle-popup-container .leaflet-popup-content {
  margin: 0;
}
</style>
```

5. Create the VehicleCard component:

```
<!-- resources/js/Pages/Search/components/VehicleCard.vue -->
<template>
  <div
    class="vehicle-card border rounded-lg overflow-hidden hover:shadow-lg transition-shadow cursor-pointer"
    :class="{ 'border-blue-500': isSelected }"
    @click="$emit('click')"
  >
    <div class="relative">
      <img
        :src="vehicle.thumbnail"
        :alt="vehicle.title"
        class="w-full h-48 object-cover"
      />
      <div class="absolute top-2 right-2">
        <button
          class="bg-white p-2 rounded-full shadow hover:bg-gray-100"
          @click.stop="toggleFavorite"
```

```
        >
          <HeartIcon :class="{ 'text-red-500': isFavorite }" />
        </button>
      </div>
    </div>

    <div class="p-4">
      <div class="flex justify-between items-start">
        <h3 class="font-bold text-lg">{{ vehicle.title }}</h3>
        <p class="text-xl font-bold">€{{ vehicle.price_per_day }}/day</p>
      </div>

      <div class="flex items-center mt-2">
        <StarRating :rating="vehicle.rating" />
        <span class="text-sm text-gray-600 ml-2">
          ({{ vehicle.reviews_count }} reviews)
        </span>
      </div>

      <div class="mt-3 flex flex-wrap gap-2">
        <span
          v-for="feature in vehicle.features.slice(0, 3)"
          :key="feature.name"
          class="text-xs bg-gray-100 px-2 py-1 rounded-full"
        >
          <component
            :is="feature.icon"
            class="w-4 h-4 inline-block mr-1"
          />
          {{ feature.name }}
        </span>
      </div>

      <div class="mt-3 text-sm text-gray-600">
        <LocationIcon class="w-4 h-4 inline-block mr-1" />
        {{ vehicle.location }}
      </div>
    </div>
  </div>
</template>

<script setup>
import { ref } from 'vue'
import { HeartIcon, LocationIcon } from '@heroicons/vue/outline'
import StarRating from '@/Components/StarRating.vue'

// Props
const props = defineProps({
  vehicle: {
    type: Object,
    required: true
  },
  isSelected: {
    type: Boolean,
    default: false
  }
})

// State
const isFavorite = ref(false)

// Methods
const toggleFavorite = (event) => {
  event.stopPropagation()
  isFavorite.value = !isFavorite.value
  // You can add favorite functionality here
```

```
}
</script>
```

Let's continue with the remaining components:

6. First, let's create the VehiclePreviewModal component:

```
<!-- resources/js/Pages/Search/components/VehiclePreviewModal.vue -->
<template>
  <Modal @close="$emit('close')">
    <div class="max-w-4xl">
      <!-- Image Gallery -->
      <div class="relative h-80">
        <img
          :src="currentImage"
          :alt="vehicle.title"
          class="w-full h-full object-cover rounded-t-lg"
        />
        <!-- Image Navigation -->
        <button
          v-if="vehicle.images.length > 1"
          class="absolute left-2 top-1/2 -translate-y-1/2 bg-white/80 p-2 rounded-full"
          @click="previousImage"
        >
          <ChevronLeftIcon class="w-6 h-6" />
        </button>
        <button
          v-if="vehicle.images.length > 1"
          class="absolute right-2 top-1/2 -translate-y-1/2 bg-white/80 p-2 rounded-full"
          @click="nextImage"
        >
          <ChevronRightIcon class="w-6 h-6" />
        </button>
      </div>

      <div class="p-6">
        <div class="flex justify-between items-start">
          <div>
            <h2 class="text-2xl font-bold">{{ vehicle.title }}</h2>
            <div class="flex items-center mt-2">
              <StarRating :rating="vehicle.rating" />
              <span class="text-sm text-gray-600 ml-2">
                ({{ vehicle.reviews_count }} reviews)
              </span>
            </div>
          </div>
          <div class="text-right">
            <p class="text-2xl font-bold">€{{ vehicle.price_per_day }}/day</p>
            <p class="text-sm text-gray-600">
              Total: €{{ totalPrice }} for {{ numberOfDays }} days
            </p>
          </div>
        </div>

        <!-- Features -->
        <div class="mt-6">
          <h3 class="font-semibold mb-2">Features</h3>
          <div class="grid grid-cols-2 gap-4">
            <div
              v-for="feature in vehicle.features"
              :key="feature.name"
              class="flex items-center"
            >
              <component :is="feature.icon" class="w-5 h-5 mr-2" />
              {{ feature.name }}
            </div>
```

```
          </div>
        </div>

        <!-- Availability Calendar -->
        <div class="mt-6">
          <h3 class="font-semibold mb-2">Availability</h3>
          <AvailabilityCalendar
            v-model:startDate="startDate"
            v-model:endDate="endDate"
            :unavailable-dates="unavailableDates"
            :min-date="new Date()"
          />
        </div>

        <!-- Actions -->
        <div class="mt-6 flex gap-4">
          <Link
            :href="route('vehicles.show', {
              vehicle: vehicle.id,
              start_date: startDate,
              end_date: endDate
            })"
            class="flex-1 bg-blue-500 text-white px-6 py-3 rounded-lg text-center hover:bg-blue-600 transition"
          >
            View Full Details
          </Link>
          <button
            class="px-6 py-3 border rounded-lg hover:bg-gray-50 transition"
            @click="$emit('close')"
          >
            Close
          </button>
        </div>
      </div>
    </div>
  </Modal>
</template>

<script setup>
import { ref, computed, onMounted } from 'vue'
import { Link } from '@inertiajs/vue3'
import { ChevronLeftIcon, ChevronRightIcon } from '@heroicons/vue/outline'
import Modal from '@/Components/Modal.vue'
import StarRating from '@/Components/StarRating.vue'
import AvailabilityCalendar from './AvailabilityCalendar.vue'

// Props
const props = defineProps({
  vehicle: {
    type: Object,
    required: true
  },
  searchParams: {
    type: Object,
    default: () => ({})
  }
})

// Emits
const emit = defineEmits(['close'])

// State
const currentImageIndex = ref(0)
const startDate = ref(props.searchParams.start_date || '')
const endDate = ref(props.searchParams.end_date || '')
```

```
const unavailableDates = ref([])

// Computed
const currentImage = computed(() => {
  return props.vehicle.images[currentImageIndex.value]
})

const numberOfDays = computed(() => {
  if (!startDate.value || !endDate.value) return 0
  const start = new Date(startDate.value)
  const end = new Date(endDate.value)
  return Math.ceil((end - start) / (1000 * 60 * 60 * 24))
})

const totalPrice = computed(() => {
  return (props.vehicle.price_per_day * numberOfDays.value).toFixed(2)
})

// Methods
const previousImage = () => {
  currentImageIndex.value = currentImageIndex.value === 0
    ? props.vehicle.images.length - 1
    : currentImageIndex.value - 1
}

const nextImage = () => {
  currentImageIndex.value = currentImageIndex.value === props.vehicle.images.length - 1
    ? 0
    : currentImageIndex.value + 1
}

// Lifecycle
onMounted(async () => {
  try {
    const response = await fetch(
      route('vehicles.availability', { vehicle: props.vehicle.id })
    )
    unavailableDates.value = await response.json()
  } catch (error) {
    console.error('Error fetching availability:', error)
  }
})
</script>
```

7. Let's create the FeatureFilter component:

```
<!-- resources/js/Pages/Search/components/FeatureFilter.vue -->
<template>
  <div class="feature-filter">
    <h3 class="font-bold mb-2">Features</h3>

    <!-- Search features -->
    <input
      type="text"
      v-model="searchQuery"
      placeholder="Search features..."
      class="w-full p-2 border rounded mb-3"
    />

    <!-- Feature list -->
    <div class="space-y-2 max-h-60 overflow-y-auto">
      <label
        v-for="feature in filteredFeatures"
        :key="feature"
        class="flex items-center hover:bg-gray-50 p-2 rounded cursor-pointer"
```

```
      >
        <input
          type="checkbox"
          :value="feature"
          v-model="selectedFeatures"
          class="rounded border-gray-300"
          @change="updateFeatures"
        />
        <span class="ml-2">{{ feature }}</span>
      </label>
    </div>

    <!-- Selected count -->
    <div v-if="selectedFeatures.length" class="mt-3 text-sm text-gray-600">
      {{ selectedFeatures.length }} feature(s) selected
    </div>
  </div>
</template>

<script setup>
import { ref, computed, watch } from 'vue'

// Props and emits
const props = defineProps({
  features: {
    type: Array,
    required: true
  },
  selected: {
    type: Array,
    default: () => []
  }
})

const emit = defineEmits(['update:selected', 'update:features'])

// State
const searchQuery = ref('')
const selectedFeatures = ref(props.selected)

// Computed
const filteredFeatures = computed(() => {
  return props.features.filter(feature =>
    feature.toLowerCase().includes(searchQuery.value.toLowerCase())
  )
})

// Methods
const updateFeatures = () => {
  emit('update:selected', selectedFeatures.value)
  emit('update:features')
}

// Watch for external changes
watch(() => props.selected, (newValue) => {
  selectedFeatures.value = newValue
})
</script>
```

8. Finally, let's create the AvailabilityCalendar component:

```
<!-- resources/js/Pages/Search/components/AvailabilityCalendar.vue -->
<template>
  <div class="availability-calendar">
    <div class="flex gap-4 mb-4">
      <div class="flex-1">
```

```
      <div class="flex-1">
        <label class="block text-sm text-gray-600 mb-1">Start Date</label>
        <DatePicker
          v-model="localStartDate"
          :min-date="minDate"
          :disabled-dates="disabledDates"
          @update:model-value="handleStartDateChange"
        />
      </div>
      <div class="flex-1">
        <label class="block text-sm text-gray-600 mb-1">End Date</label>
        <DatePicker
          v-model="localEndDate"
          :min-date="computedMinEndDate"
          :disabled-dates="disabledDates"
          @update:model-value="handleEndDateChange"
        />
      </div>
    </div>

    <!-- Calendar View -->
    <div class="calendar-grid">
      <!-- Calendar implementation -->
    </div>
  </div>
</template>

<script setup>
import { ref, computed, watch } from 'vue'
import DatePicker from '@/Components/DatePicker.vue'
import { format, addDays, isWithinInterval } from 'date-fns'

// Props
const props = defineProps({
  startDate: String,
  endDate: String,
  unavailableDates: {
    type: Array,
    default: () => []
  },
  minDate: {
    type: Date,
    default: () => new Date()
  }
})

// Emits
const emit = defineEmits(['update:startDate', 'update:endDate'])

// State
const localStartDate = ref(props.startDate)
const localEndDate = ref(props.endDate)

// Computed
const computedMinEndDate = computed(() => {
  return localStartDate.value ? addDays(new Date(localStartDate.value), 1) : props.minDate
})

const disabledDates = computed(() => {
  return props.unavailableDates.map(dateRange => ({
    start: new Date(dateRange.start),
    end: new Date(dateRange.end)
  }))
})

// Methods
```

```
const isDateDisabled = (date) => {
  return disabledDates.value.some(range =>
    isWithinInterval(date, { start: range.start, end: range.end })
  )
}

const handleStartDateChange = (date) => {
  localStartDate.value = date
  emit('update:startDate', date)

  // Clear end date if it's before new start date
  if (localEndDate.value && new Date(localEndDate.value) <= new Date(date)) {
    localEndDate.value = null
    emit('update:endDate', null)
  }
}

const handleEndDateChange = (date) => {
  localEndDate.value = date
  emit('update:endDate', date)
}

// Watch for external changes
watch(() => props.startDate, (newValue) => {
  localStartDate.value = newValue
})

watch(() => props.endDate, (newValue) => {
  localEndDate.value = newValue
})
</script>

<style scoped>
.calendar-grid {
  display: grid;
  grid-template-columns: repeat(7, 1fr);
  gap: 1px;
  background-color: #e5e7eb;
  border: 1px solid #e5e7eb;
  border-radius: 0.5rem;
  overflow: hidden;
}
</style>
```

I'll continue with the remaining components and then move to the next parts sequentially.

9. Let's create the DatePicker component:

```
<!-- resources/js/Components/DatePicker.vue -->
<template>
  <div class="date-picker relative">
    <div
      class="input-wrapper"
      @click="showCalendar = !showCalendar"
    >
      <input
        type="text"
        :value="formattedDate"
        readonly
        placeholder="Select date"
        class="w-full p-2 border rounded cursor-pointer"
      />
      <CalendarIcon class="absolute right-2 top-1/2 -translate-y-1/2 w-5 h-5 text-gray-400" />
    </div>

    <!-- Calendar Dropdown -->
```

```
    <div
      v-if="showCalendar"
      class="absolute z-50 mt-1 bg-white rounded-lg shadow-lg border p-4 w-72"
      v-click-outside="() => showCalendar = false"
    >
      <!-- Month Navigation -->
      <div class="flex items-center justify-between mb-4">
        <button
          class="p-1 hover:bg-gray-100 rounded"
          @click="previousMonth"
        >
          <ChevronLeftIcon class="w-5 h-5" />
        </button>
        <span class="font-semibold">
          {{ format(currentDate, 'MMMM yyyy') }}
        </span>
        <button
          class="p-1 hover:bg-gray-100 rounded"
          @click="nextMonth"
        >
          <ChevronRightIcon class="w-5 h-5" />
        </button>
      </div>

      <!-- Weekday Headers -->
      <div class="grid grid-cols-7 mb-2">
        <span
          v-for="day in weekDays"
          :key="day"
          class="text-center text-sm font-medium text-gray-600"
        >
          {{ day }}
        </span>
      </div>

      <!-- Calendar Grid -->
      <div class="grid grid-cols-7 gap-1">
        <button
          v-for="date in calendarDates"
          :key="date.toISOString()"
          :class="[
            'h-8 w-8 rounded-full flex items-center justify-center text-sm',
            isSelected(date) ? 'bg-blue-500 text-white' : 'hover:bg-gray-100',
            isDisabled(date) ? 'text-gray-300 cursor-not-allowed' : 'cursor-pointer',
            isToday(date) ? 'border border-blue-500' : ''
          ]"
          :disabled="isDisabled(date)"
          @click="selectDate(date)"
        >
          {{ date.getDate() }}
        </button>
      </div>
    </div>
  </div>
</template>

<script setup>
import { ref, computed } from 'vue'
import {
  format,
  startOfMonth,
  endOfMonth,
  eachDayOfInterval,
  startOfWeek,
  endOfWeek,
```

```
  isSameDay,
  addMonths,
  subMonths,
  isAfter,
  isBefore,
  isWithinInterval
} from 'date-fns'
import { CalendarIcon, ChevronLeftIcon, ChevronRightIcon } from '@heroicons/vue/outline'

// Props
const props = defineProps({
  modelValue: {
    type: [Date, String],
    default: null
  },
  minDate: {
    type: Date,
    default: null
  },
  maxDate: {
    type: Date,
    default: null
  },
  disabledDates: {
    type: Array,
    default: () => []
  }
})

// Emits
const emit = defineEmits(['update:modelValue'])

// State
const showCalendar = ref(false)
const currentDate = ref(props.modelValue ? new Date(props.modelValue) : new Date())

// Computed
const weekDays = computed(() => ['Su', 'Mo', 'Tu', 'We', 'Th', 'Fr', 'Sa'])

const calendarDates = computed(() => {
  const monthStart = startOfMonth(currentDate.value)
  const monthEnd = endOfMonth(monthStart)
  const start = startOfWeek(monthStart)
  const end = endOfWeek(monthEnd)

  return eachDayOfInterval({ start, end })
})

const formattedDate = computed(() => {
  if (!props.modelValue) return ''
  return format(new Date(props.modelValue), 'MMM dd, yyyy')
})

// Methods
const isSelected = (date) => {
  return props.modelValue && isSameDay(new Date(props.modelValue), date)
}

const isDisabled = (date) => {
  // Check min/max dates
  if (props.minDate && isBefore(date, props.minDate)) return true
  if (props.maxDate && isAfter(date, props.maxDate)) return true

  // Check disabled dates
  return props.disabledDates.some(range =>
    isWithinInterval(date, {
```

```
      start: new Date(range.start),
      end: new Date(range.end)
    })
  )
}

const isToday = (date) => {
  return isSameDay(date, new Date())
}

const selectDate = (date) => {
  if (isDisabled(date)) return
  emit('update:modelValue', format(date, 'yyyy-MM-dd'))
  showCalendar.value = false
}

const previousMonth = () => {
  currentDate.value = subMonths(currentDate.value, 1)
}

const nextMonth = () => {
  currentDate.value = addMonths(currentDate.value, 1)
}
</script>

<style scoped>
.date-picker {
  width: 100%;
}
</style>
```

10. Create the StarRating component:

```
<!-- resources/js/Components/StarRating.vue -->
<template>
  <div class="flex items-center">
    <div class="flex">
      <StarIcon
        v-for="i in 5"
        :key="i"
        :class="[
          'w-4 h-4',
          i <= rating ? 'text-yellow-400' : 'text-gray-300'
        ]"
      />
    </div>
  </div>
</template>

<script setup>
import { StarIcon } from '@heroicons/vue/solid'

defineProps({
  rating: {
    type: Number,
    required: true,
    validator: value => value >= 0 && value <= 5
  }
})
</script>
```

11. Create the Modal base component:

```
<!-- resources/js/Components/Modal.vue -->
<template>
  <TransitionRoot appear :show="true" as="template">
    <Dialog as="div" @close="$emit('close')" class="relative z-50">
      <TransitionChild
        as="template"
        enter="duration-300 ease-out"
        enter-from="opacity-0"
        enter-to="opacity-100"
        leave="duration-200 ease-in"
        leave-from="opacity-100"
        leave-to="opacity-0"
      >
        <div class="fixed inset-0 bg-black bg-opacity-25" />
      </TransitionChild>

      <div class="fixed inset-0 overflow-y-auto">
        <div class="flex min-h-full items-center justify-center p-4 text-center">
          <TransitionChild
            as="template"
            enter="duration-300 ease-out"
            enter-from="opacity-0 scale-95"
            enter-to="opacity-100 scale-100"
            leave="duration-200 ease-in"
            leave-from="opacity-100 scale-100"
            leave-to="opacity-0 scale-95"
          >
            <DialogPanel
              class="w-full max-w-xl transform overflow-hidden rounded-2xl bg-white text-left align-middle shadow-xl transition-all"
            >
              <button
                class="absolute right-4 top-4 text-gray-400 hover:text-gray-500"
                @click="$emit('close')"
              >
                <XIcon class="h-6 w-6" />
              </button>

              <slot></slot>
            </DialogPanel>
          </TransitionChild>
        </div>
      </div>
    </Dialog>
  </TransitionRoot>
</template>

<script setup>
import { Dialog, DialogPanel, TransitionChild, TransitionRoot } from '@headlessui/vue'
import { XIcon } from '@heroicons/vue/outline'

defineEmits(['close'])
</script>
```

12. Create the LoadingSpinner component:

```
<!-- resources/js/Components/LoadingSpinner.vue -->
<template>
  <div class="flex justify-center items-center">
    <svg
      class="animate-spin h-8 w-8 text-blue-500"
      xmlns="http://www.w3.org/2000/svg"
      fill="none"
      viewBox="0 0 24 24"
    >
      <circle
        class="opacity-25"
        cx="12"
        cy="12"
        r="10"
        stroke="currentColor"
        stroke-width="4"
      />
      <path
        class="opacity-75"
        fill="currentColor"
        d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"
      />
    </svg>
  </div>
</template>
```

13. Add a click-outside directive:

```
// resources/js/directives/click-outside.js
export const clickOutside = {
  beforeMount(el, binding) {
    el.clickOutsideEvent = (event) => {
      if (!(el === event.target || el.contains(event.target))) {
        binding.value(event)
      }
    }
    document.addEventListener('click', el.clickOutsideEvent)
  },
  unmounted(el) {
    document.removeEventListener('click', el.clickOutsideEvent)
  }
}
```

14. Register the directive in your app:

```
// resources/js/app.js
import { createApp, h } from 'vue'
import { createInertiaApp } from '@inertiajs/vue3'
import { clickOutside } from './directives/click-outside'

createInertiaApp({
  resolve: name => {
    const pages = import.meta.glob('./Pages/**/*.vue', { eager: true })
    return pages[`./Pages/${name}.vue`]
  },
  setup({ el, App, props, plugin }) {
    const app = createApp({ render: () => h(App, props) })
    app.use(plugin)
    app.directive('click-outside', clickOutside)
    app.mount(el)
  },
})
```

I'll continue with the remaining parts of PART 4: Map Integration & Marker Clustering.

1. First, let's enhance our map configuration by adding marker clustering:

```
// resources/js/config/map-config.js
export const clusterOptions = {
  maxClusterRadius: 50,
  spiderfyOnMaxZoom: true,
  showCoverageOnHover: false,
  zoomToBoundsOnClick: true,
  disableClusteringAtZoom: 15,
  chunkedLoading: true,
  iconCreateFunction: function(cluster) {
    const count = cluster.getChildCount();
    return L.divIcon({
      html: `<div class="cluster-marker">${count}</div>`,
      className: 'custom-cluster',
      iconSize: L.point(40, 40)
    });
  }
};

export const mapOptions = {
  zoomControl: false,
  scrollWheelZoom: true,
  maxZoom: 18,
  minZoom: 3
};
```

2. Update the VehicleMap component to include clustering:

```
<!-- resources/js/Components/VehicleMap.vue -->
<template>
  <div class="relative h-full">
    <l-map
      v-model:zoom="zoom"
      :center="center"
      :options="mapOptions"
      @ready="handleMapReady"
    >
      <l-tile-layer
        url="https://tiles.stadiamaps.com/tiles/osm_bright/{z}/{x}/{y}{r}.png"
        :attribution="attribution"
      />

      <!-- Custom zoom controls -->
      <div class="absolute right-5 top-5 z-[1000] flex flex-col gap-2">
        <button
          @click="handleZoom(1)"
          class="bg-white p-2 rounded-lg shadow hover:bg-gray-50"
        >
          <PlusIcon class="w-5 h-5" />
        </button>
        <button
          @click="handleZoom(-1)"
          class="bg-white p-2 rounded-lg shadow hover:bg-gray-50"
        >
          <MinusIcon class="w-5 h-5" />
        </button>
      </div>

      <!-- Marker Cluster Layer -->
      <l-marker-cluster :options="clusterOptions">
        <l-marker
          v-for="vehicle in vehicles"
          :key="vehicle.id"
          :lat-lng="[vehicle.latitude, vehicle.longitude]"
```

```
          @click="handleMarkerClick(vehicle)"
        >
          <!-- Price Tooltip -->
          <l-tooltip
            :permanent="true"
            :interactive="true"
            :options="{ direction: 'top', offset: [0, -10] }"
          >
            <div class="price-tooltip">
              €{{ vehicle.price_per_day }}
            </div>
          </l-tooltip>

          <!-- Info Popup -->
          <l-popup :options="popupOptions">
            <div class="vehicle-popup">
              <img
                :src="vehicle.thumbnail"
                :alt="vehicle.title"
                class="w-full h-32 object-cover rounded-t"
              />
              <div class="p-3">
                <h3 class="font-bold">{{ vehicle.title }}</h3>
                <div class="flex items-center mt-1">
                  <StarRating :rating="vehicle.rating" />
                  <span class="text-sm ml-2">({{ vehicle.reviews_count }})</span>
                </div>
                <p class="font-bold mt-2">€{{ vehicle.price_per_day }}/day</p>
                <button
                  @click="$emit('marker-click', vehicle)"
                  class="mt-2 bg-blue-500 text-white px-4 py-1 rounded text-sm w-full"
                >
                  View Details
                </button>
              </div>
            </div>
          </l-popup>
        </l-marker>
      </l-marker-cluster>
    </l-map>

    <!-- Loading overlay -->
    <div
      v-if="isLoading"
      class="absolute inset-0 bg-white/75 flex items-center justify-center"
    >
      <LoadingSpinner />
    </div>
  </div>
</template>

<script setup>
import { ref, onMounted, watch } from 'vue'
import {
  LMap,
  LTileLayer,
  LMarker,
  LTooltip,
  LPopup,
  LMarkerCluster
} from '@vue-leaflet/vue-leaflet'
import { PlusIcon, MinusIcon } from '@heroicons/vue/outline'
import { clusterOptions, mapOptions } from '@/config/map-config'
import StarRating from '@/Components/StarRating.vue'
import LoadingSpinner from '@/Components/LoadingSpinner.vue'
import 'leaflet.markercluster/dist/MarkerCluster.css'
```

```
import 'leaflet.markercluster/dist/MarkerCluster.Default.css'

// Props
const props = defineProps({
  vehicles: {
    type: Array,
    required: true
  },
  center: {
    type: Array,
    required: true
  },
  zoom: {
    type: Number,
    default: 12
  }
})

// Emits
const emit = defineEmits(['marker-click', 'update:zoom', 'update:center'])

// State
const mapInstance = ref(null)
const isLoading = ref(false)
const attribution = ref('&copy; <a href="https://stadiamaps.com/">Stadia Maps</a>')

// Popup options
const popupOptions = {
  maxWidth: 300,
  className: 'vehicle-popup',
  closeButton: false
}

// Methods
const handleMapReady = (map) => {
  mapInstance.value = map
  // Add custom controls if needed
  map.on('moveend', handleMapMove)
  map.on('zoomend', handleMapZoom)
}

const handleMarkerClick = (vehicle) => {
  emit('marker-click', vehicle)
}

const handleZoom = (delta) => {
  if (mapInstance.value) {
    const newZoom = mapInstance.value.getZoom() + delta
    mapInstance.value.setZoom(newZoom)
  }
}

const handleMapMove = () => {
  if (mapInstance.value) {
    const center = mapInstance.value.getCenter()
    emit('update:center', [center.lat, center.lng])
  }
}

const handleMapZoom = () => {
  if (mapInstance.value) {
    emit('update:zoom', mapInstance.value.getZoom())
  }
}
```

```
// Watch for center changes
watch(() => props.center, (newCenter) => {
  if (mapInstance.value && newCenter) {
    mapInstance.value.setView(newCenter, props.zoom)
  }
}, { deep: true })

// Watch for zoom changes
watch(() => props.zoom, (newZoom) => {
  if (mapInstance.value && newZoom) {
    mapInstance.value.setZoom(newZoom)
  }
})

// Cleanup
onUnmounted(() => {
  if (mapInstance.value) {
    mapInstance.value.remove()
  }
})
</script>

<style scoped>
.price-tooltip {
  background: white;
  padding: 4px 8px;
  border-radius: 4px;
  border: 2px solid #3b82f6;
  font-weight: bold;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

:deep(.custom-cluster) {
  background: white;
  border: 2px solid #3b82f6;
  display: flex;
  align-items: center;
  justify-content: center;
  border-radius: 50%;
  font-weight: bold;
  color: #3b82f6;
}

:deep(.cluster-marker) {
  width: 100%;
  height: 100%;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 14px;
}

:deep(.vehicle-popup .leaflet-popup-content-wrapper) {
  padding: 0;
  overflow: hidden;
  border-radius: 8px;
}

:deep(.vehicle-popup .leaflet-popup-content) {
  margin: 0;
  width: 300px !important;
}
</style>
```

3. Add the map performance optimization utilities:

```
// resources/js/utils/map-utils.js
import { debounce } from 'lodash'

export const optimizeMarkers = (map, markers) => {
  const bounds = map.getBounds()
  return markers.filter(marker => {
    const latLng = [marker.latitude, marker.longitude]
    return bounds.contains(latLng)
  })
}

export const debouncedMapUpdate = debounce((callback) => {
  callback()
}, 300)

export const calculateMapBounds = (vehicles) => {
  if (!vehicles.length) return null

  const latitudes = vehicles.map(v => v.latitude)
  const longitudes = vehicles.map(v => v.longitude)

  return [
    [Math.min(...latitudes), Math.min(...longitudes)],
    [Math.max(...latitudes), Math.max(...longitudes)]
  ]
}
```

4. Create custom map icons:

```
// resources/js/config/map-icons.js
import L from 'leaflet'

export const createCustomIcon = (price) => {
  return L.divIcon({
    className: 'custom-marker',
    html: `
      <div class="marker-content">
        <div class="price">€${price}</div>
      </div>
    `,
    iconSize: [40, 40],
    iconAnchor: [20, 40]
  })
}

export const createClusterIcon = (cluster) => {
  const count = cluster.getChildCount()
  const size = count < 10 ? 'small' : count < 50 ? 'medium' : 'large'

  return L.divIcon({
    className: `cluster-icon ${size}`,
    html: `<div class="cluster-content">${count}</div>`,
    iconSize: [40, 40]
  })
}
```

5. Add map-specific CSS:

```css
/* resources/css/map.css */
.custom-marker {
  background: none;
  border: none;
}

.marker-content {
  background: white;
  border: 2px solid #3b82f6;
  border-radius: 4px;
  padding: 4px 8px;
  font-weight: bold;
  color: #3b82f6;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.cluster-icon {
  background: white;
  border: 2px solid #3b82f6;
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  font-weight: bold;
  color: #3b82f6;
}

.cluster-icon.small {
  width: 30px;
  height: 30px;
}

.cluster-icon.medium {
  width: 35px;
  height: 35px;
}

.cluster-icon.large {
  width: 40px;
  height: 40px;
}
```

This completes PART 4 with all the map integration features including:

- Marker clustering with custom styles
- Performance optimizations
- Custom controls
- Responsive map handling
- Event management
- Custom icons and tooltips

Let's proceed with PART 5: Search Filters & Price Range Component.

1. First, let's create an enhanced PriceRangeFilter component:

```vue
<!-- resources/js/Components/Filters/PriceRangeFilter.vue -->
<template>
  <div class="price-range-filter p-4 border-b">
    <div class="flex justify-between items-center mb-4">
      <h3 class="font-semibold">Price Range</h3>
      <button
        @click="resetRange"
        class="text-sm text-blue-500 hover:text-blue-600"
      >
        Reset
      </button>
    </div>
```

```
<!-- Price Display -->
<div class="flex justify-between mb-6">
  <div class="text-center">
    <label class="text-sm text-gray-600">Min Price</label>
    <div class="font-semibold">€{{ formatPrice(localMin) }}</div>
  </div>
  <div class="text-center">
    <label class="text-sm text-gray-600">Max Price</label>
    <div class="font-semibold">€{{ formatPrice(localMax) }}</div>
  </div>
</div>

<!-- Range Slider -->
<div class="relative pt-2 pb-6">
  <!-- Track -->
  <div class="h-2 bg-gray-200 rounded-full">
    <!-- Selected Range -->
    <div
      class="absolute h-2 bg-blue-500 rounded-full"
      :style="{
        left: `${((localMin - minPrice) / (maxPrice - minPrice)) * 100}%`,
        right: `${100 - ((localMax - minPrice) / (maxPrice - minPrice)) * 100}%`
      }"
    ></div>
  </div>

  <!-- Handles -->
  <input
    type="range"
    :min="minPrice"
    :max="maxPrice"
    :step="step"
    v-model.number="localMin"
    class="absolute w-full top-0 h-2 appearance-none pointer-events-none"
    :style="{ background: 'transparent' }"
  />
  <input
    type="range"
    :min="minPrice"
    :max="maxPrice"
    :step="step"
    v-model.number="localMax"
    class="absolute w-full top-0 h-2 appearance-none pointer-events-none"
    :style="{ background: 'transparent' }"
  />
</div>

<!-- Price Distribution Graph -->
<div class="mt-4 h-20">
  <div class="relative h-full">
    <div
      v-for="(count, index) in priceDistribution"
      :key="index"
      class="absolute bottom-0 bg-gray-200 w-2"
      :style="{
        height: `${(count / maxCount) * 100}%`,
        left: `${(index / (priceDistribution.length - 1)) * 100}%`
      }"
    >
      <div
        class="absolute bottom-0 w-full bg-blue-500 transition-all duration-200"
        :style="{
          height: isInSelectedRange(index) ? '100%' : '0%'
        }"
      ></div>
    </div>
```

```
        </div>
      </div>
    </div>

    <!-- Apply Button -->
    <button
      @click="applyFilter"
      class="w-full mt-4 bg-blue-500 text-white py-2 rounded-lg hover:bg-blue-600 transition"
      :disabled="!hasChanged"
    >
      Apply Filter
    </button>
  </div>
</template>

<script setup>
import { ref, computed, watch } from 'vue'
import { debounce } from 'lodash'

const props = defineProps({
  minPrice: {
    type: Number,
    required: true
  },
  maxPrice: {
    type: Number,
    required: true
  },
  modelValue: {
    type: Object,
    default: () => ({
      min: null,
      max: null
    })
  },
  priceDistribution: {
    type: Array,
    default: () => []
  },
  step: {
    type: Number,
    default: 1
  }
})

const emit = defineEmits(['update:modelValue', 'filter'])

// Local state
const localMin = ref(props.modelValue.min || props.minPrice)
const localMax = ref(props.modelValue.max || props.maxPrice)
const initialMin = ref(localMin.value)
const initialMax = ref(localMax.value)

// Computed
const hasChanged = computed(() => {
  return localMin.value !== initialMin.value ||
         localMax.value !== initialMax.value
})

const maxCount = computed(() => {
  return Math.max(...props.priceDistribution)
})

// Methods
const formatPrice = (price) => {
  return price.toLocaleString('en-US', {
```

```
    minimumFractionDigits: 0,
    maximumFractionDigits: 0
  })
}

const isInSelectedRange = (index) => {
  const price = props.minPrice + (index * (props.maxPrice - props.minPrice) / (props.priceDistribution.length - 1))
  return price >= localMin.value && price <= localMax.value
}

const resetRange = () => {
  localMin.value = props.minPrice
  localMax.value = props.maxPrice
  applyFilter()
}

const applyFilter = debounce(() => {
  emit('update:modelValue', {
    min: localMin.value,
    max: localMax.value
  })
  emit('filter')

  initialMin.value = localMin.value
  initialMax.value = localMax.value
}, 300)

// Watch for external changes
watch(() => props.modelValue, (newValue) => {
  if (newValue.min !== localMin.value) {
    localMin.value = newValue.min
  }
  if (newValue.max !== localMax.value) {
    localMax.value = newValue.max
  }
}, { deep: true })

// Ensure min doesn't exceed max
watch(localMin, (newMin) => {
  if (newMin > localMax.value) {
    localMax.value = newMin
  }
})

// Ensure max doesn't fall below min
watch(localMax, (newMax) => {
  if (newMax < localMin.value) {
    localMin.value = newMax
  }
})
</script>

<style scoped>
input[type="range"] {
  -webkit-appearance: none;
  background: transparent;
}

input[type="range"]::-webkit-slider-thumb {
  -webkit-appearance: none;
  height: 16px;
  width: 16px;
  border-radius: 50%;
  background: #3b82f6;
  cursor: pointer;
```

```
  pointer-events: auto;
  margin-top: -7px;
  border: 2px solid white;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

input[type="range"]::-moz-range-thumb {
  height: 16px;
  width: 16px;
  border-radius: 50%;
  background: #3b82f6;
  cursor: pointer;
  pointer-events: auto;
  border: 2px solid white;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}
</style>
```

2. Let's create a FilterContainer component to manage all filters:

```
<!-- resources/js/Components/Filters/FilterContainer.vue -->
<template>
  <div class="filters-container">
    <!-- Active Filters -->
    <div v-if="hasActiveFilters" class="p-4 border-b">
      <div class="flex items-center justify-between mb-2">
        <h3 class="font-semibold">Active Filters</h3>
        <button
          @click="clearAllFilters"
          class="text-sm text-blue-500 hover:text-blue-600"
        >
          Clear All
        </button>
      </div>
      <div class="flex flex-wrap gap-2">
        <div
          v-for="filter in activeFilters"
          :key="filter.id"
          class="flex items-center bg-blue-50 text-blue-700 px-3 py-1 rounded-full text-sm"
        >
          {{ filter.label }}
          <XIcon
            class="w-4 h-4 ml-2 cursor-pointer"
            @click="removeFilter(filter)"
          />
        </div>
      </div>
    </div>

    <!-- Price Range Filter -->
    <PriceRangeFilter
      v-model="filters.price"
      :min-price="priceRange.min"
      :max-price="priceRange.max"
      :price-distribution="priceDistribution"
      @filter="applyFilters"
    />

    <!-- Feature Filter -->
    <FeatureFilter
      v-model="filters.features"
      :available-features="availableFeatures"
      @filter="applyFilters"
    />

    <!-- Vehicle Type Filter -->
```

```
    <!-- Vehicle Type Filter -->
    <VehicleTypeFilter
      v-model="filters.vehicleType"
      :available-types="availableTypes"
      @filter="applyFilters"
    />

    <!-- More filters can be added here -->
  </div>
</template>

<script setup>
import { ref, computed } from 'vue'
import { XIcon } from '@heroicons/vue/solid'
import PriceRangeFilter from './PriceRangeFilter.vue'
import FeatureFilter from './FeatureFilter.vue'
import VehicleTypeFilter from './VehicleTypeFilter.vue'

const props = defineProps({
  priceRange: {
    type: Object,
    required: true
  },
  availableFeatures: {
    type: Array,
    required: true
  },
  availableTypes: {
    type: Array,
    required: true
  },
  priceDistribution: {
    type: Array,
    required: true
  }
})

const emit = defineEmits(['filter'])

// Filter state
const filters = ref({
  price: {
    min: props.priceRange.min,
    max: props.priceRange.max
  },
  features: [],
  vehicleType: null
})

// Computed
const hasActiveFilters = computed(() => {
  return activeFilters.value.length > 0
})

const activeFilters = computed(() => {
  const active = []

  if (filters.value.price.min > props.priceRange.min ||
      filters.value.price.max < props.priceRange.max) {
    active.push({
      id: 'price',
      label: `€${filters.value.price.min} - €${filters.value.price.max}`,
      type: 'price'
    })
  }
```

```
  filters.value.features.forEach(feature => {
    active.push({
      id: `feature-${feature}`,
      label: feature,
      type: 'feature'
    })
  })

  if (filters.value.vehicleType) {
    active.push({
      id: 'vehicle-type',
      label: filters.value.vehicleType,
      type: 'vehicleType'
    })
  }

  return active
})

// Methods
const applyFilters = () => {
  emit('filter', filters.value)
}

const removeFilter = (filter) => {
  switch (filter.type) {
    case 'price':
      filters.value.price = {
        min: props.priceRange.min,
        max: props.priceRange.max
      }
      break
    case 'feature':
      filters.value.features = filters.value.features.filter(f => f !== filter.label)
      break
    case 'vehicleType':
      filters.value.vehicleType = null
      break
  }
  applyFilters()
}

const clearAllFilters = () => {
  filters.value = {
    price: {
      min: props.priceRange.min,
      max: props.priceRange.max
    },
    features: [],
    vehicleType: null
  }
  applyFilters()
}
</script>
```

Let's continue with the remaining filter components and functionality.

3. Let's create the VehicleTypeFilter component:

```
<!-- resources/js/Components/Filters/VehicleTypeFilter.vue -->
<template>
  <div class="vehicle-type-filter p-4 border-b">
    <h3 class="font-semibold mb-4">Vehicle Type</h3>

    <div class="space-y-2">
      <div
```

```
        v-for="type in vehicleTypes"
        :key="type.id"
        class="vehicle-type-option"
        :class="{ 'selected': modelValue === type.id }"
        @click="selectType(type.id)"
      >
        <div class="flex items-center p-3 rounded-lg cursor-pointer hover:bg-gray-50">
          <!-- Icon -->
          <div class="w-12 h-12 flex items-center justify-center bg-blue-50 rounded-lg">
            <component
              :is="type.icon"
              class="w-6 h-6 text-blue-500"
            />
          </div>

          <!-- Info -->
          <div class="ml-4 flex-1">
            <div class="font-medium">{{ type.name }}</div>
            <div class="text-sm text-gray-500">{{ type.description }}</div>
          </div>

          <!-- Count Badge -->
          <div
            v-if="type.count"
            class="ml-2 px-2 py-1 bg-gray-100 rounded-full text-sm"
          >
            {{ type.count }}
          </div>
        </div>
      </div>
    </div>
  </div>
</template>

<script setup>
import {
  TruckIcon,
  HomeIcon,
  UserGroupIcon,
  LightningBoltIcon
} from '@heroicons/vue/outline'

const props = defineProps({
  modelValue: String,
  availableTypes: {
    type: Array,
    required: true
  }
})

const emit = defineEmits(['update:modelValue', 'filter'])

// Vehicle types with icons and descriptions
const vehicleTypes = computed(() => [
  {
    id: 'rv',
    name: 'RV',
    icon: HomeIcon,
    description: 'Recreational vehicles & motorhomes',
    count: getTypeCount('rv')
  },
  {
    id: 'campervan',
    name: 'Campervan',
    icon: TruckIcon,
```

```
    description: 'Converted vans & smaller campers',
    count: getTypeCount('campervan')
  },
  {
    id: 'caravan',
    name: 'Caravan',
    icon: UserGroupIcon,
    description: 'Towable caravans & trailers',
    count: getTypeCount('caravan')
  },
  {
    id: 'compact',
    name: 'Compact RV',
    icon: LightningBoltIcon,
    description: 'Compact & easy to drive vehicles',
    count: getTypeCount('compact')
  }
])

// Methods
const getTypeCount = (typeId) => {
  return props.availableTypes.find(t => t.id === typeId)?.count || 0
}

const selectType = (typeId) => {
  const newValue = props.modelValue === typeId ? null : typeId
  emit('update:modelValue', newValue)
  emit('filter')
}
</script>

<style scoped>
.vehicle-type-option.selected {
  @apply bg-blue-50;
}

.vehicle-type-option.selected:hover {
  @apply bg-blue-50;
}
</style>
```

4. Add filter URL synchronization:

```
// resources/js/utils/filter-utils.js
export const encodeFilters = (filters) => {
  const params = new URLSearchParams()

  if (filters.price?.min) params.append('min_price', filters.price.min)
  if (filters.price?.max) params.append('max_price', filters.price.max)
  if (filters.features?.length) params.append('features', filters.features.join(','))
  if (filters.vehicleType) params.append('type', filters.vehicleType)

  return params.toString()
}

export const decodeFilters = (queryString) => {
  const params = new URLSearchParams(queryString)

  return {
    price: {
      min: params.get('min_price') ? Number(params.get('min_price')) : null,
      max: params.get('max_price') ? Number(params.get('max_price')) : null
    },
    features: params.get('features') ? params.get('features').split(',') : [],
    vehicleType: params.get('type') || null
  }
}
```

5. Create a FilterStore for state management:

```
// resources/js/stores/filter-store.js
import { defineStore } from 'pinia'
import { encodeFilters, decodeFilters } from '@/utils/filter-utils'

export const useFilterStore = defineStore('filters', {
  state: () => ({
    filters: {
      price: {
        min: null,
        max: null
      },
      features: [],
      vehicleType: null
    },
    priceRange: {
      min: 0,
      max: 1000
    },
    availableFeatures: [],
    availableTypes: []
  }),

  getters: {
    hasActiveFilters: (state) => {
      return (
        state.filters.features.length > 0 ||
        state.filters.vehicleType ||
        state.filters.price.min > state.priceRange.min ||
        state.filters.price.max < state.priceRange.max
      )
    },

    encodedFilters: (state) => {
      return encodeFilters(state.filters)
    }
  },

  actions: {
```

```
    initializeFilters(queryString) {
      const decoded = decodeFilters(queryString)
      this.filters = {
        ...this.filters,
        ...decoded
      }
    },

    setFilter(key, value) {
      this.filters[key] = value
    },

    clearFilters() {
      this.filters = {
        price: {
          min: this.priceRange.min,
          max: this.priceRange.max
        },
        features: [],
        vehicleType: null
      }
    },

    updatePriceRange(min, max) {
      this.priceRange.min = min
      this.priceRange.max = max
    },

    setAvailableFeatures(features) {
      this.availableFeatures = features
    },

    setAvailableTypes(types) {
      this.availableTypes = types
    }
  }
})
```

6. Update the Search page to use the filter store:

```
<!-- resources/js/Pages/Search/Index.vue -->
<template>
  <div class="flex flex-col h-screen">
    <!-- Search Header -->
    <SearchHeader />

    <div class="flex flex-1 overflow-hidden">
      <!-- Filters Sidebar -->
      <div class="w-80 border-r overflow-y-auto">
        <FilterContainer
          :price-range="filterStore.priceRange"
          :available-features="filterStore.availableFeatures"
          :available-types="filterStore.availableTypes"
          @filter="handleFilterChange"
        />
      </div>

      <!-- Map and Results -->
      <div class="flex-1 flex flex-col">
        <!-- Results count and view toggle -->
        <div class="p-4 border-b flex justify-between items-center">
          <div class="text-gray-600">
            {{ vehicles.length }} vehicles found
          </div>
          <div class="flex gap-2">
```

```
            <button
              @click="view = 'list'"
              :class="{ 'text-blue-500': view === 'list' }"
            >
              <ListIcon class="w-5 h-5" />
            </button>
            <button
              @click="view = 'map'"
              :class="{ 'text-blue-500': view === 'map' }"
            >
              <MapIcon class="w-5 h-5" />
            </button>
          </div>
        </div>

        <!-- Results view -->
        <div class="flex-1 overflow-hidden">
          <VehicleMap
            v-if="view === 'map'"
            :vehicles="vehicles"
            :center="mapCenter"
            @marker-click="showVehiclePreview"
          />
          <div
            v-else
            class="h-full overflow-y-auto p-4 grid grid-cols-2 gap-4"
          >
            <VehicleCard
              v-for="vehicle in vehicles"
              :key="vehicle.id"
              :vehicle="vehicle"
              @click="showVehiclePreview(vehicle)"
            />
          </div>
        </div>
      </div>
    </div>

    <!-- Vehicle Preview Modal -->
    <VehiclePreviewModal
      v-if="selectedVehicle"
      :vehicle="selectedVehicle"
      @close="selectedVehicle = null"
    />
  </div>
</template>

<script setup>
import { ref, onMounted } from 'vue'
import { useFilterStore } from '@/stores/filter-store'
import { router } from '@inertiajs/vue3'
import { ListIcon, MapIcon } from '@heroicons/vue/outline'

// Components imports...

const props = defineProps({
  vehicles: Array,
  filters: Object,
  priceRange: Object,
  availableFeatures: Array,
  availableTypes: Array
})

// Store
const filterStore = useFilterStore()
```

```
// State
const view = ref('map')
const selectedVehicle = ref(null)
const mapCenter = ref([25.2048, 55.2708]) // Default to Dubai

// Methods
const handleFilterChange = (filters) => {
  router.get(route('search'), filters, {
    preserveState: true,
    preserveScroll: true,
    replace: true
  })
}

const showVehiclePreview = (vehicle) => {
  selectedVehicle.value = vehicle
}

// Initialize store with URL params
onMounted(() => {
  filterStore.initializeFilters(window.location.search)
  filterStore.updatePriceRange(props.priceRange.min, props.priceRange.max)
  filterStore.setAvailableFeatures(props.availableFeatures)
  filterStore.setAvailableTypes(props.availableTypes)
})
</script>
```

Let's continue with the additional filter components and optimizations.

7. Let's create an AmenitiesFilter component:

```
<!-- resources/js/Components/Filters/AmenitiesFilter.vue -->
<template>
  <div class="amenities-filter p-4 border-b">
    <div class="flex justify-between items-center mb-4">
      <h3 class="font-semibold">Amenities</h3>
      <button
        v-if="selectedCount"
        @click="clearAll"
        class="text-sm text-blue-500 hover:text-blue-600"
      >
        Clear ({{ selectedCount }})
      </button>
    </div>

    <!-- Search -->
    <div class="mb-4">
      <input
        type="text"
        v-model="searchQuery"
        placeholder="Search amenities..."
        class="w-full p-2 border rounded-lg"
      />
    </div>

    <!-- Categories -->
    <div class="space-y-4">
      <div v-for="(group, category) in groupedAmenities" :key="category">
        <div
          class="flex items-center cursor-pointer mb-2"
          @click="toggleCategory(category)"
        >
          <ChevronRightIcon
            class="w-4 h-4 transform transition-transform"
            :class="{ 'rotate-90': expandedCategories[category] }"
```

```
          />
          <span class="ml-2 font-medium">{{ category }}</span>
          <span class="ml-2 text-sm text-gray-500">
            ({{ getSelectedCountInCategory(category) }}/{{ group.length }})
          </span>
        </div>

        <div v-show="expandedCategories[category]" class="ml-6 space-y-2">
          <label
            v-for="amenity in filteredAmenities(group)"
            :key="amenity.id"
            class="flex items-center p-2 hover:bg-gray-50 rounded-lg cursor-pointer"
          >
            <input
              type="checkbox"
              :value="amenity.id"
              v-model="selected"
              class="rounded border-gray-300"
            />
            <div class="ml-3 flex items-center">
              <component
                :is="amenity.icon"
                class="w-5 h-5 text-gray-500"
              />
              <span class="ml-2">{{ amenity.name }}</span>
            </div>
            <span
              v-if="amenity.count"
              class="ml-auto text-sm text-gray-500"
            >
              {{ amenity.count }}
            </span>
          </label>
        </div>
      </div>
    </div>

    <!-- Show More Button -->
    <button
      v-if="hasHiddenAmenities"
      @click="showAll = !showAll"
      class="mt-4 text-sm text-blue-500 hover:text-blue-600 w-full text-center"
    >
      {{ showAll ? 'Show Less' : `Show ${hiddenCount} More` }}
    </button>
  </div>
</template>

<script setup>
import { ref, computed } from 'vue'
import { ChevronRightIcon } from '@heroicons/vue/solid'
import {
  WifiIcon,
  TvIcon,
  DeviceMobileIcon,
  SparklesIcon,
  CogIcon,
  SunIcon,
  // ... import other icons as needed
} from '@heroicons/vue/outline'

const props = defineProps({
  modelValue: {
    type: Array,
    default: () => []
  },
```

```
  availableAmenities: {
    type: Array,
    required: true
  }
})

const emit = defineEmits(['update:modelValue', 'filter'])

// State
const searchQuery = ref('')
const selected = ref(props.modelValue)
const showAll = ref(false)
const expandedCategories = ref({})

// Constants
const INITIAL_SHOW_COUNT = 5
const AMENITIES_MAP = {
  'Connectivity': [
    { id: 'wifi', name: 'WiFi', icon: WifiIcon },
    { id: 'tv', name: 'TV', icon: TvIcon },
    { id: 'usb', name: 'USB Ports', icon: DeviceMobileIcon }
  ],
  'Comfort': [
    { id: 'ac', name: 'Air Conditioning', icon: SparklesIcon },
    { id: 'heating', name: 'Heating', icon: SunIcon }
  ],
  'Kitchen': [
    { id: 'fridge', name: 'Refrigerator', icon: CogIcon },
    // Add more kitchen amenities...
  ]
  // Add more categories...
}

// Computed
const groupedAmenities = computed(() => {
  const groups = {}
  Object.entries(AMENITIES_MAP).forEach(([category, amenities]) => {
    groups[category] = amenities.map(amenity => ({
      ...amenity,
      count: getAmenityCount(amenity.id)
    }))
  })
  return groups
})

const selectedCount = computed(() => selected.value.length)

const hasHiddenAmenities = computed(() => {
  return !showAll.value && getTotalAmenitiesCount() > INITIAL_SHOW_COUNT
})

const hiddenCount = computed(() => {
  return getTotalAmenitiesCount() - INITIAL_SHOW_COUNT
})

// Methods
const getAmenityCount = (amenityId) => {
  return props.availableAmenities.find(a => a.id === amenityId)?.count || 0
}

const getTotalAmenitiesCount = () => {
  return Object.values(AMENITIES_MAP).reduce((total, amenities) => total + amenities.length, 0)
}

const getSelectedCountInCategory = (category) => {
```

```
  const categoryAmenities = AMENITIES_MAP[category].map(a => a.id)
  return selected.value.filter(id => categoryAmenities.includes(id)).length
}

const filteredAmenities = (amenities) => {
  if (!searchQuery.value) return amenities
  return amenities.filter(amenity =>
    amenity.name.toLowerCase().includes(searchQuery.value.toLowerCase())
  )
}

const toggleCategory = (category) => {
  expandedCategories.value[category] = !expandedCategories.value[category]
}

const clearAll = () => {
  selected.value = []
  emit('update:modelValue', [])
  emit('filter')
}

// Watch for changes
watch(selected, (newValue) => {
  emit('update:modelValue', newValue)
  emit('filter')
})
</script>

<style scoped>
.amenities-filter {
  max-height: calc(100vh - 200px);
  overflow-y: auto;
}
</style>
```

8. Let's add a RatingFilter component:

```
<!-- resources/js/Components/Filters/RatingFilter.vue -->
<template>
  <div class="rating-filter p-4 border-b">
    <h3 class="font-semibold mb-4">Rating</h3>

    <div class="space-y-2">
      <label
        v-for="rating in ratings"
        :key="rating.value"
        class="flex items-center p-2 hover:bg-gray-50 rounded-lg cursor-pointer"
      >
        <input
          type="radio"
          :value="rating.value"
          v-model="selectedRating"
          class="text-blue-500"
          @change="updateRating"
        />
        <div class="ml-3 flex items-center">
          <StarRating :rating="rating.value" />
          <span class="ml-2">& up</span>
          <span class="ml-auto text-sm text-gray-500">
            ({{ rating.count }})
          </span>
        </div>
      </label>
    </div>
  </div>
</template>

<script setup>
import { ref, watch } from 'vue'
import StarRating from '@/Components/StarRating.vue'

const props = defineProps({
  modelValue: Number,
  ratingCounts: {
    type: Object,
    required: true
  }
})

const emit = defineEmits(['update:modelValue', 'filter'])

const selectedRating = ref(props.modelValue)

const ratings = computed(() => [
  { value: 4, count: props.ratingCounts[4] || 0 },
  { value: 3, count: props.ratingCounts[3] || 0 },
  { value: 2, count: props.ratingCounts[2] || 0 },
  { value: 1, count: props.ratingCounts[1] || 0 }
])

const updateRating = () => {
  emit('update:modelValue', selectedRating.value)
  emit('filter')
}

watch(() => props.modelValue, (newValue) => {
  selectedRating.value = newValue
})
</script>
```

9. Add mobile responsiveness to the FilterContainer:

```vue
<!-- resources/js/Components/Filters/MobileFilterDrawer.vue -->
<template>
  <div>
    <!-- Mobile filter button -->
    <button
      class="md:hidden fixed bottom-4 left-1/2 transform -translate-x-1/2
             bg-blue-500 text-white px-6 py-3 rounded-full shadow-lg"
      @click="isOpen = true"
    >
      <FilterIcon class="w-5 h-5 inline-block mr-2" />
      Filters
      <span v-if="activeFilterCount" class="ml-2 bg-white text-blue-500 px-2 rounded-full">
        {{ activeFilterCount }}
      </span>
    </button>

    <!-- Mobile filter drawer -->
    <TransitionRoot appear :show="isOpen" as="template">
      <Dialog as="div" @close="isOpen = false" class="relative z-50">
        <TransitionChild
          enter="ease-out duration-300"
          enter-from="opacity-0"
          enter-to="opacity-100"
          leave="ease-in duration-200"
          leave-from="opacity-100"
          leave-to="opacity-0"
        >
          <div class="fixed inset-0 bg-black bg-opacity-25" />
        </TransitionChild>

        <div class="fixed inset-0 overflow-hidden">
          <div class="absolute inset-0 overflow-hidden">
            <div class="pointer-events-none fixed inset-y-0 right-0 flex max-w-full pl-10">
              <TransitionChild
                enter="transform transition ease-in-out duration-300"
                enter-from="translate-x-full"
                enter-to="translate-x-0"
                leave="transform transition ease-in-out duration-300"
                leave-from="translate-x-0"
                leave-to="translate-x-full"
              >
                <DialogPanel class="pointer-events-auto w-screen max-w-md">
                  <div class="flex h-full flex-col overflow-y-scroll bg-white shadow-xl">
                    <!-- Header -->
                    <div class="px-4 py-6 sm:px-6">
                      <div class="flex items-start justify-between">
                        <h2 class="text-lg font-medium">Filters</h2>
                        <button
                          class="rounded-md text-gray-400 hover:text-gray-500"
                          @click="isOpen = false"
                        >
                          <XIcon class="h-6 w-6" />
                        </button>
                      </div>
                    </div>

                    <!-- Filter content -->
                    <div class="flex-1">
                      <slot></slot>
                    </div>

                    <!-- Footer -->
                    <div class="border-t border-gray-200 px-4 py-6">
                      <div class="flex justify-between">
                        <button
```

```
                        class="text-blue-500"
                        @click="$emit('clear')"
                      >
                        Clear all
                      </button>
                      <button
                        class="bg-blue-500 text-white px-6 py-2 rounded-lg"
                        @click="applyFilters"
                      >
                        Show {{ resultCount }} results
                      </button>
                    </div>
                  </div>
                </div>
              </DialogPanel>
            </TransitionChild>
          </div>
        </div>
      </div>
    </Dialog>
  </TransitionRoot>
</div>
</template>

<script setup>
import { ref } from 'vue'
import {
  Dialog,
  DialogPanel,
  TransitionChild,
  TransitionRoot
} from '@headlessui/vue'
import { FilterIcon, XIcon } from '@heroicons/vue/outline'

const props = defineProps({
  activeFilterCount: Number,
  resultCount: Number
})

const emit = defineEmits(['apply', 'clear'])

const isOpen = ref(false)

const applyFilters = () => {
  emit('apply')
  isOpen.value = false
}
</script>
```