Simple Pet Transportation Marketplace - MVP Specification

Tech Stack

- **Frontend**: Next.js 14 with TypeScript
- **Database**: Supabase (PostgreSQL)
- **Authentication**: Clerk
- **Payments**: Stripe Connect
- **Styling**: Tailwind CSS
- **Deployment**: Vercel

Core Features for MVP

1. User Types

- **Pet Owners**: Can request quotes and book transport
- **Transporters**: Can view requests and submit bids
- **Admin**: Basic oversight (you can use Supabase dashboard initially)

2. Essential Pages

Home Page

Header with navigation (Login/Signup, How it Works, For Transporters)
Hero section with quote request form:
- Origin zip code
- Destination zip code
- Pet type (Dog/Cat/Other)
- Pet size (Small/Medium/Large/Extra Large)
- Preferred date
- Get Quotes button

Trust indicators:
- "500+ Happy Pet Parents"
- "Background Checked Transporters"
- "Fully Insured Trips"

How it works (3 steps):
1. Tell us about your pet's journey
2. Get quotes from verified transporters
3. Book and track your pet's safe trip

Quote Dashboard (Pet Owner)

- List of active quote requests
- Incoming bids with transporter details
- Accept/decline bids
- Message transporters

Transporter Dashboard

- Available transport requests feed
- Submit bid form
- Manage active bids
- Transport history

Transporter Onboarding

- Basic profile setup
- Upload license/insurance docs
- Vehicle information
- Service areas

- Stripe Connect setup

3. Database Schema (Supabase)

```sql
```

```sql
-- Profiles (extends Clerk users)
create table profiles (
  id uuid references auth.users on delete cascade primary key,
  user_type text check (user_type in ('pet_owner', 'transporter')) not null,
  first_name text not null,
  last_name text not null,
  phone text,
  created_at timestamp with time zone default timezone('utc'::text, now()) not null,
  updated_at timestamp with time zone default timezone('utc'::text, now()) not null
);

-- Transporter profiles
create table transporter_profiles (
  id uuid references profiles(id) on delete cascade primary key,
  business_name text,
  license_number text,
  insurance_company text,
  insurance_policy text,
  vehicle_type text,
  vehicle_capacity text,
  service_radius integer default 100,
  base_rate decimal(10,2),
  rate_per_mile decimal(5,2),
  stripe_account_id text,
  is_approved boolean default false,
  created_at timestamp with time zone default timezone('utc'::text, now()) not null
);

-- Transport requests
create table transport_requests (
  id uuid default gen_random_uuid() primary key,
  user_id uuid references profiles(id) not null,
  origin_zip text not null,
  destination_zip text not null,
  pet_type text not null,
  pet_size text not null,
  pet_name text,
  preferred_date date,
  special_instructions text,
  status text default 'active' check (status in ('active', 'matched', 'completed', 'cancelled')),
  created_at timestamp with time zone default timezone('utc'::text, now()) not null
);
```

```sql
-- Bids
create table bids (
  id uuid default gen_random_uuid() primary key,
  request_id uuid references transport_requests(id) on delete cascade not null,
  transporter_id uuid references profiles(id) not null,
  price decimal(10,2) not null,
  pickup_date date not null,
  delivery_date date not null,
  message text,
  status text default 'pending' check (status in ('pending', 'accepted', 'declined', 'withdrawn')),
  created_at timestamp with time zone default timezone('utc'::text, now()) not null
);

-- Confirmed transports
create table confirmed_transports (
  id uuid default gen_random_uuid() primary key,
  request_id uuid references transport_requests(id) not null,
  bid_id uuid references bids(id) not null,
  customer_id uuid references profiles(id) not null,
  transporter_id uuid references profiles(id) not null,
  total_amount decimal(10,2) not null,
  platform_fee decimal(10,2) not null,
  stripe_payment_intent_id text,
  status text default 'scheduled' check (status in ('scheduled', 'in_progress', 'completed', 'cancelled')),
  pickup_date date not null,
  delivery_date date not null,
  created_at timestamp with time zone default timezone('utc'::text, now()) not null
);

-- Enable RLS
alter table profiles enable row level security;
alter table transporter_profiles enable row level security;
alter table transport_requests enable row level security;
alter table bids enable row level security;
alter table confirmed_transports enable row level security;

-- Basic RLS policies
create policy "Users can view their own profile" on profiles for select using (auth.uid() = id);
create policy "Users can update their own profile" on profiles for update using (auth.uid() = id);
```

4. Key Components to Build

Quote Request Form (Home Page)

tsx

```tsx
'use client'

import { useState } from 'react'
import { useRouter } from 'next/navigation'
import { useAuth } from '@clerk/nextjs'

export default function QuoteRequestForm() {
  const { isSignedIn } = useAuth()
  const router = useRouter()
  const [formData, setFormData] = useState({
    originZip: '',
    destinationZip: '',
    petType: '',
    petSize: '',
    preferredDate: '',
    petName: '',
    specialInstructions: ''
  })

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault()

    if (!isSignedIn) {
      // Redirect to sign up, then back to form
      router.push('/sign-up?redirect=/request-quote')
      return
    }

    // Create transport request via Supabase
    // Redirect to dashboard
  }

  return (
    <form onSubmit={handleSubmit} className="space-y-4 max-w-md mx-auto">
      <div className="grid grid-cols-2 gap-4">
        <input
          placeholder="From ZIP"
          value={formData.originZip}
          onChange={(e) => setFormData({...formData, originZip: e.target.value})}
          required
        />
        <input
          placeholder="To ZIP"
```

```
        value={formData.destinationZip}
        onChange={(e) => setFormData({...formData, destinationZip: e.target.value})}
        required
      />
    </div>

    <select
      value={formData.petType}
      onChange={(e) => setFormData({...formData, petType: e.target.value})}
      required
    >
      <option value="">Select Pet Type</option>
      <option value="dog">Dog</option>
      <option value="cat">Cat</option>
      <option value="other">Other</option>
    </select>

    <select
      value={formData.petSize}
      onChange={(e) => setFormData({...formData, petSize: e.target.value})}
      required
    >
      <option value="">Select Pet Size</option>
      <option value="small">Small (under 25 lbs)</option>
      <option value="medium">Medium (25-60 lbs)</option>
      <option value="large">Large (60-100 lbs)</option>
      <option value="extra-large">Extra Large (over 100 lbs)</option>
    </select>

    <input
      type="date"
      value={formData.preferredDate}
      onChange={(e) => setFormData({...formData, preferredDate: e.target.value})}
      required
    />

    <button
      type="submit"
      className="w-full bg-blue-600 text-white py-2 px-4 rounded hover:bg-blue-700"
    >
      Get Free Quotes
    </button>
  </form>
```

```
  )
}
```

Transporter Dashboard

tsx

```tsx
'use client'

import { useEffect, useState } from 'react'
import { useUser } from '@clerk/nextjs'
import { supabase } from '@/lib/supabase'

export default function TransporterDashboard() {
  const { user } = useUser()
  const [requests, setRequests] = useState([])
  const [activeTab, setActiveTab] = useState('available')

  useEffect(() => {
    fetchAvailableRequests()
  }, [])

  const fetchAvailableRequests = async () => {
    const { data } = await supabase
      .from('transport_requests')
      .select('*')
      .eq('status', 'active')
      .order('created_at', { ascending: false })

    setRequests(data || [])
  }

  const submitBid = async (requestId: string, bidData: any) => {
    const { error } = await supabase
      .from('bids')
      .insert({
        request_id: requestId,
        transporter_id: user?.id,
        ...bidData
      })

    if (!error) {
      // Refresh data or show success message
    }
  }

  return (
    <div className="container mx-auto p-4">
      <h1 className="text-2xl font-bold mb-6">Transporter Dashboard</h1>
```

```
      <div className="tabs mb-6">
        <button
          onClick={() => setActiveTab('available')}
          className={`px-4 py-2 mr-2 ${activeTab === 'available' ? 'bg-blue-600 text-white' : 'bg-gray-200'}`}
        >
          Available Requests
        </button>
        <button
          onClick={() => setActiveTab('mybids')}
          className={`px-4 py-2 ${activeTab === 'mybids' ? 'bg-blue-600 text-white' : 'bg-gray-200'}`}
        >
          My Bids
        </button>
      </div>

      {activeTab === 'available' && (
        <div className="space-y-4">
          {requests.map((request) => (
            <RequestCard
              key={request.id}
              request={request}
              onSubmitBid={submitBid}
            />
          ))}
        </div>
      )}
    </div>
  )
}
```

5. Copy/Content

Home Page Hero

"Safe, Reliable Pet Transportation Nationwide"

Whether you're moving across the country or your furry friend needs to get to grandma's house, our network of professional, background-checked transporters will get your pet there safely.

✓ Background-checked transporters
✓ Fully insured trips
✓ Real-time updates
✓ Starting at $1.50/mile

How It Works

1. Tell Us About Your Journey
Enter your pickup and destination, plus details about your pet

2. Get Competing Quotes
Verified transporters bid on your job with their best price

3. Choose & Book
Select your preferred transporter and pay securely online

6. Simple Payment Flow

1. Customer accepts a bid
2. Stripe payment intent created for full amount
3. Payment authorized (not captured yet)
4. On transport completion, payment captured
5. Platform fee deducted, remainder transferred to transporter

7. MVP Launch Checklist

Pre-Launch

- [ ] Set up Supabase project and database
- [ ] Configure Clerk authentication
- [ ] Set up Stripe Connect
- [ ] Build core pages (Home, Dashboard, Onboarding)
- [ ] Create basic quote request flow
- [ ] Build bidding system
- [ ] Set up basic payment processing
- [ ] Add simple email notifications

Post-Launch Features (Phase 2)

- Real-time messaging between users
- Photo uploads during transport
- Review/rating system
- Advanced search and filters
- Mobile app

- SEO landing pages

8. Pricing Strategy

- Platform fee: 15% of transaction value
- Customer service fee: $25 per booking
- Transporter payout: 85% of bid amount minus fees

This simplified version gets you to market quickly while maintaining the core value proposition. You can build and test with real users, then add complexity based on their feedback.

Want me to help you start with any specific component or walk through setting up the Supabase schema?