Etsy Seller Automater Frontend Architecture

Multi-Tenant SaaS Deep Dive for QNAP NAS Deployment

1. Overview

This document provides a comprehensive technical deep dive into the React.js frontend architecture for the Etsy Seller Automater, specifically designed to integrate with the multi-tenant SaaS backend deployed on your QNAP TBS-464 NAS. The frontend architecture emphasizes scalability, tenant isolation, and optimal performance within the Docker containerized environment.

2. Frontend Technology Stack

Core Technologies:

- **React 18+**: Modern functional components with hooks
- **JavaScript/TypeScript**: TypeScript recommended for production builds
- **Tailwind CSS**: Utility-first CSS framework for responsive design
- **Vite**: Fast build tool and development server
- **Docker**: Containerized deployment matching backend architecture

Key Libraries & Dependencies:

- **Axios/Fetch API**: HTTP client for backend communication
- **React Router v6**: Client-side routing and navigation
- **React Query/TanStack Query**: Server state management and caching
- **Zustand**: Lightweight client-side state management
- **React Hook Form**: Form validation and handling
- **Framer Motion**: UI animations and transitions
- **Chart.js/Recharts**: Analytics dashboard visualizations

3. Multi-Tenant Frontend Architecture

3.1 Tenant Context System

```javascript
```

```
// contexts/TenantContext.js
import { createContext, useContext, useState, useEffect } from 'react';

const TenantContext = createContext();

export const TenantProvider = ({ children }) => {
  const [tenant, setTenant] = useState(null);
  const [isLoading, setIsLoading] = useState(true);

  useEffect(() => {
    // Extract tenant from subdomain or path
    const detectTenant = () => {
      const hostname = window.location.hostname;
      const subdomain = hostname.split('.')[0];

      // tenant1.localhost, tenant2.localhost for local dev
      // tenant1.yourdomain.com for production
      if (subdomain !== 'localhost' && subdomain !== 'www') {
        return subdomain;
      }

      // Fallback to path-based routing: /tenant1/dashboard
      const pathSegments = window.location.pathname.split('/');
      return pathSegments[1] || 'default';
    };

    const tenantId = detectTenant();
    setTenant({
      id: tenantId,
      name: tenantId,
      theme: getTenantTheme(tenantId),
      config: getTenantConfig(tenantId)
    });
    setIsLoading(false);
  }, []);

  return (
    <TenantContext.Provider value={{ tenant, setTenant, isLoading }}>
      {children}
    </TenantContext.Provider>
  );
};
```

```
export const useTenant = () => {
  const context = useContext(TenantContext);
  if (!context) {
    throw new Error('useTenant must be used within TenantProvider');
  }
  return context;
};
```

3.2 Dynamic Theming & Branding

javascript

```
// utils/tenantThemes.js
export const getTenantTheme = (tenantId) => {
  const themes = {
    tenant1: {
      primary: '#3B82F6', // Blue
      secondary: '#10B981', // Green
      accent: '#F59E0B', // Amber
      logo: '/logos/tenant1-logo.svg',
      favicon: '/favicons/tenant1.ico'
    },
    tenant2: {
      primary: '#EF4444', // Red
      secondary: '#8B5CF6', // Purple
      accent: '#F97316', // Orange
      logo: '/logos/tenant2-logo.svg',
      favicon: '/favicons/tenant2.ico'
    },
    default: {
      primary: '#6B7280', // Gray
      secondary: '#374151',
      accent: '#9CA3AF',
      logo: '/logos/default-logo.svg',
      favicon: '/favicon.ico'
    }
  };

  return themes[tenantId] || themes.default;
};

// components/ThemeProvider.jsx
import { useTenant } from '../contexts/TenantContext';

export const ThemeProvider = ({ children }) => {
  const { tenant } = useTenant();

  useEffect(() => {
    if (tenant?.theme) {
      document.documentElement.style.setProperty('--primary-color', tenant.theme.primary);
      document.documentElement.style.setProperty('--secondary-color', tenant.theme.secondary);
      document.documentElement.style.setProperty('--accent-color', tenant.theme.accent);

      // Update favicon dynamically
      const favicon = document.querySelector('link[rel="icon"]');
      if (favicon) {
        favicon.href = tenant.theme.favicon;
      }
    }
  }, [tenant]);

  return <div className="theme-container">{children}</div>;
};
```

4. Component Architecture

4.1 Feature-Based Structure

```
frontend/
 ├── src/
 │   ├── components/
 │   │   ├── common/          # Shared UI components
 │   │   │   ├── Layout/
 │   │   │   ├── Forms/
 │   │   │   ├── Navigation/
 │   │   │   └── UI/
 │   │   ├── features/        # Feature-specific components
 │   │   │   ├── Analytics/
 │   │   │   ├── Designs/
 │   │   │   ├── MaskCreator/
 │   │   │   ├── OAuth/
 │   │   │   └── ShopManagement/
 │   │   └── tenant/          # Tenant-specific overrides
 │   │       ├── tenant1/
 │   │       └── tenant2/
 │   ├── hooks/               # Custom React hooks
 │   ├── services/            # API services and utilities
 │   ├── stores/              # State management
 │   ├── utils/               # Utility functions
 │   ├── contexts/            # React contexts
 │   └── pages/               # Page components
 └── public/
     ├── logos/               # Tenant logos
     ├── favicons/            # Tenant favicons
     └── tenant-assets/       # Tenant-specific assets
```

4.2 Core Components

Layout Component with Multi-Tenant Support:

javascript

```javascript
// components/common/Layout/AppLayout.jsx
import { useTenant } from '../../../contexts/TenantContext';
import { Navigation } from './Navigation';
import { Sidebar } from './Sidebar';
import { Header } from './Header';

export const AppLayout = ({ children }) => {
  const { tenant, isLoading } = useTenant();

  if (isLoading) {
    return <LoadingSpinner />;
  }

  return (
    <div className="min-h-screen bg-gray-50">
      <Header tenant={tenant} />
      <div className="flex">
        <Sidebar tenant={tenant} />
        <main className="flex-1 p-6">
          <div className="max-w-7xl mx-auto">
            {children}
          </div>
        </main>
      </div>
    </div>
  );
};
```

Tenant-Aware API Service:

javascript

```javascript
// services/apiService.js
import axios from 'axios';

class ApiService {
  constructor() {
    this.baseURL = process.env.REACT_APP_API_BASE_URL || 'http://localhost:3003';
    this.client = axios.create({
      baseURL: this.baseURL,
      timeout: 10000,
    });

    // Add tenant header to all requests
    this.client.interceptors.request.use((config) => {
      const tenant = this.getCurrentTenant();
      if (tenant) {
        config.headers['X-Tenant-ID'] = tenant.id;
      }

      const token = localStorage.getItem('auth_token');
      if (token) {
        config.headers.Authorization = `Bearer ${token}`;
      }

      return config;
    });

    // Handle tenant-specific errors
    this.client.interceptors.response.use(
      (response) => response,
      (error) => {
        if (error.response?.status === 403) {
          // Tenant access denied
          window.location.href = '/unauthorized';
        }
        return Promise.reject(error);
      }
    );
  }

  getCurrentTenant() {
    // Extract from URL or context
    const hostname = window.location.hostname;
    const subdomain = hostname.split('.')[0];
    return { id: subdomain };
  }

  // Etsy-specific API methods
  async getShopAnalytics(params = {}) {
    const response = await this.client.get('/api/shop-analytics', { params });
    return response.data;
  }

  async getTopSellers(year) {
    const response = await this.client.get(`/api/top-sellers?year=${year}`);
    return response.data;
  }

  async getLocalImages() {
    const response = await this.client.get('/api/local-images');
    return response.data;
  }

  async saveMaskData(maskData) {
    const response = await this.client.post('/api/masks', maskData);
    return response.data;
  }
}

export const apiService = new ApiService();
```

5. Feature Implementation

5.1 OAuth Integration Component

javascript

```
// components/features/OAuth/EtsyOAuthConnect.jsx
import { useState, useEffect } from 'react';
import { useTenant } from '../../../contexts/TenantContext';
import { apiService } from '../../../services/apiService';

export const EtsyOAuthConnect = () => {
  const { tenant } = useTenant();
  const [isConnected, setIsConnected] = useState(false);
  const [isLoading, setIsLoading] = useState(false);

  const handleConnect = async () => {
    setIsLoading(true);
    try {
      const oauthData = await apiService.getOAuthConfig();

      // Build OAuth URL with tenant-specific redirect
      const params = new URLSearchParams({
        client_id: oauthData.client_id,
        redirect_uri: `${window.location.origin}/oauth/redirect`,
        scope: oauthData.scope,
        state: tenant.id, // Include tenant ID in state
        response_type: 'code',
        code_challenge_method: 'S256',
        code_challenge: oauthData.code_challenge,
      });

      window.location.href = `https://www.etsy.com/oauth/connect?${params}`;
    } catch (error) {
      console.error('OAuth connection failed:', error);
    } finally {
      setIsLoading(false);
    }
  };

  return (
    <div className="bg-white rounded-lg shadow-md p-6">
      <h2 className="text-2xl font-bold mb-4">Connect Your Etsy Shop</h2>
      {!isConnected ? (
        <button
          onClick={handleConnect}
          disabled={isLoading}
          className="bg-primary text-white px-6 py-3 rounded-lg hover:bg-primary-dark disabled:opacity-50"
        >
```

```
          {isLoading ? 'Connecting...' : 'Connect to Etsy'}
        </button>
      ) : (
        <div className="text-green-600">
          ✅ Successfully connected to Etsy!
        </div>
      )}
    </div>
  );
};
```

5.2 Analytics Dashboard

```javascript
```

```jsx
// components/features/Analytics/AnalyticsDashboard.jsx
import { useState, useEffect } from 'react';
import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, ResponsiveContainer } from 'recharts';
import { apiService } from '../../../services/apiService';

export const AnalyticsDashboard = () => {
  const [analytics, setAnalytics] = useState(null);
  const [selectedYear, setSelectedYear] = useState(new Date().getFullYear());
  const [isLoading, setIsLoading] = useState(true);

  useEffect(() => {
    fetchAnalytics();
  }, [selectedYear]);

  const fetchAnalytics = async () => {
    try {
      setIsLoading(true);
      const [topSellers, monthlyData] = await Promise.all([
        apiService.getTopSellers(selectedYear),
        apiService.getMonthlyAnalytics(selectedYear)
      ]);

      setAnalytics({
        topSellers,
        monthlyData: monthlyData.map(item => ({
          month: item.month,
          sales: item.total_sales,
          orders: item.total_orders,
          revenue: item.total_revenue
        }))
      });
    } catch (error) {
      console.error('Failed to fetch analytics:', error);
    } finally {
      setIsLoading(false);
    }
  };

  if (isLoading) {
    return <div className="animate-pulse">Loading analytics...</div>;
  }

  return (
<div className="space-y-6">
  <div className="flex justify-between items-center">
    <h1 className="text-3xl font-bold">Shop Analytics</h1>
    <select
      value={selectedYear}
      onChange={(e) => setSelectedYear(parseInt(e.target.value))}
      className="border rounded-lg px-4 py-2"
    >
      {[2024, 2023, 2022].map(year => (
        <option key={year} value={year}>{year}</option>
      ))}
    </select>
  </div>

  {/* Revenue Chart */}
  <div className="bg-white rounded-lg shadow-md p-6">
    <h2 className="text-xl font-semibold mb-4">Monthly Revenue</h2>
    <ResponsiveContainer width="100%" height={300}>
      <LineChart data={analytics?.monthlyData}>
        <CartesianGrid strokeDasharray="3 3" />
        <XAxis dataKey="month" />
        <YAxis />
        <Tooltip />
        <Line type="monotone" dataKey="revenue" stroke="#3B82F6" strokeWidth={2} />
      </LineChart>
    </ResponsiveContainer>
  </div>

  {/* Top Sellers Grid */}
  <div className="bg-white rounded-lg shadow-md p-6">
    <h2 className="text-xl font-semibold mb-4">Top Selling Items</h2>
    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">
      {analytics?.topSellers?.map((item, index) => (
        <div key={item.listing_id} className="border rounded-lg p-4">
          <img
            src={item.image_url}
            alt={item.title}
            className="w-full h-32 object-cover rounded mb-2"
          />
          <h3 className="font-medium truncate">{item.title}</h3>
          <p className="text-gray-600">${item.price}</p>
          <p className="text-sm text-green-600">{item.total_sales} sales</p>
        </div>
      ))}
```

```
        </div>
      </div>
    </div>
  );
};
```

5.3 Mask Creator Tool

```
javascript
```

```jsx
// components/features/MaskCreator/MaskCreator.jsx
import { useState, useRef, useCallback } from 'react';
import { apiService } from '../../../services/apiService';

export const MaskCreator = () => {
  const canvasRef = useRef(null);
  const [image, setImage] = useState(null);
  const [masks, setMasks] = useState([]);
  const [isDrawing, setIsDrawing] = useState(false);
  const [drawMode, setDrawMode] = useState('point'); // 'point' or 'rectangle'

  const handleImageUpload = (event) => {
    const file = event.target.files[0];
    if (file) {
      const reader = new FileReader();
      reader.onload = (e) => {
        const img = new Image();
        img.onload = () => {
          const canvas = canvasRef.current;
          const ctx = canvas.getContext('2d');
          canvas.width = img.width;
          canvas.height = img.height;
          ctx.drawImage(img, 0, 0);
          setImage(img);
        };
        img.src = e.target.result;
      };
      reader.readAsDataURL(file);
    }
  };

  const handleCanvasClick = useCallback((event) => {
    if (!image) return;

    const canvas = canvasRef.current;
    const rect = canvas.getBoundingClientRect();
    const x = event.clientX - rect.left;
    const y = event.clientY - rect.top;

    if (drawMode === 'point') {
      const newMask = { type: 'point', x, y, id: Date.now() };
      setMasks(prev => [...prev, newMask]);
    // Draw point on canvas
    const ctx = canvas.getContext('2d');
    ctx.fillStyle = 'rgba(255, 0, 0, 0.7)';
    ctx.beginPath();
    ctx.arc(x, y, 5, 0, 2 * Math.PI);
    ctx.fill();
  }
}, [image, drawMode]);

const saveMasks = async () => {
  if (masks.length === 0) return;

  try {
    const canvas = canvasRef.current;
    const imageData = canvas.toDataURL();

    await apiService.saveMaskData({
      image_data: imageData,
      masks: masks,
      timestamp: new Date().toISOString()
    });

    alert('Masks saved successfully!');
  } catch (error) {
    console.error('Failed to save masks:', error);
    alert('Failed to save masks');
  }
};

return (
  <div className="bg-white rounded-lg shadow-md p-6">
    <h2 className="text-2xl font-bold mb-4">Mask Creator</h2>

    <div className="mb-4">
      <input
        type="file"
        accept="image/*"
        onChange={handleImageUpload}
        className="mb-4"
      />

      <div className="flex gap-4 mb-4">
        <button
          onClick={() => setDrawMode('point')}
          className={`px-4 py-2 rounded ${
            drawMode === 'point'
              ? 'bg-primary text-white'
              : 'bg-gray-200 text-gray-700'
          }`}
        >
          Point Mode
        </button>
        <button
          onClick={() => setDrawMode('rectangle')}
          className={`px-4 py-2 rounded ${
            drawMode === 'rectangle'
              ? 'bg-primary text-white'
              : 'bg-gray-200 text-gray-700'
          }`}
        >
          Rectangle Mode
        </button>
      </div>
    </div>

    <div className="border-2 border-dashed border-gray-300 rounded-lg p-4">
      <canvas
        ref={canvasRef}
        onClick={handleCanvasClick}
        className="max-w-full h-auto cursor-crosshair"
        style={{ maxHeight: '500px' }}
      />
    </div>

    <div className="mt-4 flex gap-4">
      <button
        onClick={() => setMasks([])}
        className="px-4 py-2 bg-gray-500 text-white rounded hover:bg-gray-600"
      >
        Clear Masks
      </button>
      <button
        onClick={saveMasks}
        disabled={masks.length === 0}
        className="px-4 py-2 bg-green-600 text-white rounded hover:bg-green-700 disabled:opacity-50"
      >
        Save Masks ({masks.length})
      </button>
      </div>
    </div>
  );
};
```

6. State Management

6.1 Zustand Store Configuration

```
javascript
```

```
// stores/appStore.js
import { create } from 'zustand';
import { persist } from 'zustand/middleware';

export const useAppStore = create(
  persist(
    (set, get) => ({
      // User state
      user: null,
      isAuthenticated: false,

      // Shop data
      shopData: null,
      listings: [],

      // UI state
      sidebarOpen: true,
      currentPage: 'dashboard',

      // Actions
      setUser: (user) => set({ user, isAuthenticated: !!user }),
      setShopData: (shopData) => set({ shopData }),
      setListings: (listings) => set({ listings }),
      toggleSidebar: () => set((state) => ({ sidebarOpen: !state.sidebarOpen })),
      setCurrentPage: (page) => set({ currentPage: page }),

      // Reset state on logout
      logout: () => set({
        user: null,
        isAuthenticated: false,
        shopData: null,
        listings: [],
      }),
    }),
    {
      name: 'etsy-automater-storage',
      partialize: (state) => ({
        user: state.user,
        isAuthenticated: state.isAuthenticated,
        sidebarOpen: state.sidebarOpen,
      }),
    }
  )
);
```

7. Docker Integration & Multi-Tenant Deployment

7.1 Dockerfile for Production

dockerfile

```dockerfile
# Dockerfile.frontend
FROM node:18-alpine as build

WORKDIR /app

# Copy package files
COPY package*.json ./
RUN npm ci --only=production

# Copy source code
COPY . .

# Build for production
ARG REACT_APP_API_BASE_URL
ARG REACT_APP_TENANT_MODE=multi
ENV REACT_APP_API_BASE_URL=$REACT_APP_API_BASE_URL
ENV REACT_APP_TENANT_MODE=$REACT_APP_TENANT_MODE

RUN npm run build

# Production stage
FROM nginx:alpine

# Copy custom nginx config for multi-tenant routing
COPY nginx.conf /etc/nginx/conf.d/default.conf
COPY --from=build /app/dist /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

7.2 Nginx Configuration for Multi-Tenant Routing

nginx

```
# nginx.conf
server {
  listen 80;
  server_name ~^(?<tenant>.+)\.localhost$ localhost;

  root /usr/share/nginx/html;
  index index.html index.htm;

  # Add tenant header
  add_header X-Tenant-ID $tenant always;

  # Handle React Router
  location / {
    try_files $uri $uri/ /index.html;

    # Pass tenant info to frontend
    add_header X-Tenant-ID $tenant always;
  }

  # API proxy to backend
  location /api/ {
    proxy_pass http://backend:3003;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Tenant-ID $tenant;
  }

  # OAuth callback handling
  location /oauth/ {
    proxy_pass http://backend:3003;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Tenant-ID $tenant;
  }
}
```

7.3 Docker Compose Integration

```
yaml
```

```yaml
# docker-compose.yml (frontend section)
version: '3.9'

services:
  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile.frontend
      args:
        REACT_APP_API_BASE_URL: http://localhost:3003
        REACT_APP_TENANT_MODE: multi
    ports:
      - "3000:80"
    depends_on:
      - backend
    environment:
      - NGINX_HOST=localhost
      - NGINX_PORT=80
    volumes:
      - ./tenant-assets:/usr/share/nginx/html/tenant-assets:ro
    networks:
      - app-network
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.frontend.rule=HostRegexp(`{subdomain:[a-zA-Z0-9-]+}.localhost`)"
      - "traefik.http.services.frontend.loadbalancer.server.port=80"

networks:
  app-network:
    driver: bridge
```

8. Performance Optimization

8.1 Code Splitting & Lazy Loading

javascript

```
// App.js
import { lazy, Suspense } from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';

// Lazy load components
const Dashboard = lazy(() => import('./pages/Dashboard'));
const Analytics = lazy(() => import('./pages/Analytics'));
const Designs = lazy(() => import('./pages/Designs'));
const MaskCreator = lazy(() => import('./pages/MaskCreator'));

export default function App() {
 return (
  <TenantProvider>
   <ThemeProvider>
    <Router>
     <AppLayout>
      <Suspense fallback={<LoadingSpinner />}>
       <Routes>
        <Route path="/" element={<Dashboard />} />
        <Route path="/analytics" element={<Analytics />} />
        <Route path="/designs" element={<Designs />} />
        <Route path="/tools/mask-creator" element={<MaskCreator />} />
       </Routes>
      </Suspense>
     </AppLayout>
    </Router>
   </ThemeProvider>
  </TenantProvider>
 );
}
```

8.2 Caching Strategy

javascript

```
// hooks/useApiQuery.js
import { useQuery } from '@tanstack/react-query';
import { apiService } from '../services/apiService';

export const useShopAnalytics = (params = {}) => {
  return useQuery({
    queryKey: ['shopAnalytics', params],
    queryFn: () => apiService.getShopAnalytics(params),
    staleTime: 5 * 60 * 1000, // 5 minutes
    cacheTime: 10 * 60 * 1000, // 10 minutes
    refetchOnWindowFocus: false,
  });
};

export const useTopSellers = (year) => {
  return useQuery({
    queryKey: ['topSellers', year],
    queryFn: () => apiService.getTopSellers(year),
    staleTime: 60 * 60 * 1000, // 1 hour
    enabled: !!year,
  });
};
```

9. Testing Strategy

9.1 Component Testing

javascript

```
// __tests__/components/Analytics/AnalyticsDashboard.test.jsx
import { render, screen, waitFor } from '@testing-library/react';
import { QueryClient, QueryClientProvider } from '@tanstack/react-query';
import { AnalyticsDashboard } from '../../../src/components/features/Analytics/AnalyticsDashboard';
import { TenantProvider } from '../../../src/contexts/TenantContext';

const queryClient = new QueryClient({
  defaultOptions: { queries: { retry: false } }
});

const renderWithProviders = (component) => {
  return render(
    <QueryClientProvider client={queryClient}>
      <TenantProvider>
        {component}
      </TenantProvider>
    </QueryClientProvider>
  );
};

describe('AnalyticsDashboard', () => {
  test('renders analytics dashboard with loading state', () => {
    renderWithProviders(<AnalyticsDashboard />);
    expect(screen.getByText('Loading analytics...')).toBeInTheDocument();
  });

  test('displays analytics data after loading', async () => {
    // Mock API response
    jest.spyOn(require('../../../src/services/apiService'), 'getTopSellers')
      .mockResolvedValue([
        { listing_id: '1', title: 'Test Product', price: '29.99', total_sales: 10 }
      ]);

    renderWithProviders(<AnalyticsDashboard />);

    await waitFor(() => {
      expect(screen.getByText('Test Product')).toBeInTheDocument();
    });
  });
});
```

10. Security Considerations

10.1 Tenant Data Isolation

javascript

```javascript
// utils/tenantSecurity.js
export const validateTenantAccess = (requestedTenant, userTenant) => {
  if (!userTenant || !requestedTenant) {
    throw new Error('Tenant information missing');
  }

  if (userTenant !== requestedTenant) {
    throw new Error('Access denied: Tenant mismatch');
  }

  return true;
};

// Enhanced API service with security
export class SecureApiService extends ApiService {
  async makeRequest(endpoint, options = {}) {
    const tenant = this.getCurrentTenant();

    // Validate tenant access before making request
    if (!tenant?.id) {
      throw new Error('No tenant context available');
    }

    return super.makeRequest(endpoint, {
      ...options,
      headers: {
        ...options.headers,
        'X-Tenant-ID': tenant.id,
        'X-Tenant-Signature': this.generateTenantSignature(tenant.id),
      }
    });
  }

  generateTenantSignature(tenantId) {
    // Generate HMAC signature for tenant validation
    const timestamp = Date.now();
    const payload = `${tenantId}:${timestamp}`;
    // In production, use proper HMAC with secret key
    return btoa(payload);
  }
}
```

11. Deployment & Monitoring

11.1 QNAP NAS Deployment Configuration

bash

```bash
# deploy-frontend.sh
#!/bin/bash

# QNAP Container Station deployment script
echo "Deploying Etsy Seller Automater Frontend to QNAP NAS..."

# Build production image
docker build -t etsy-frontend:latest \
  --build-arg REACT_APP_API_BASE_URL=http://your-qnap-ip:3003 \
  --build-arg REACT_APP_TENANT_MODE=multi \
  -f Dockerfile.frontend .

# Create tenant-specific volumes
docker volume create etsy_tenant_assets
docker volume create etsy_nginx_config

# Deploy with Container Station
docker run -d \
  --name etsy-frontend-prod \
  --restart unless-stopped \
  -p 80:80 \
  -p 443:443 \
  -v etsy_tenant_assets:/usr/share/nginx/html/tenant-assets \
  -v etsy_nginx_config:/etc/nginx/conf.d \
  --network etsy-network \
  etsy-frontend:latest

echo "Frontend deployed successfully!"
echo "Access at: http://tenant1.your-qnap-ip or http://your-qnap-ip"
```

11.2 Environment Configuration

javascript

```
// config/environment.js
const environments = {
  development: {
    API_BASE_URL: 'http://localhost:3003',
    TENANT_MODE: 'single',
    DEBUG: true,
    CACHE_TTL: 60000, // 1 minute
  },
  production: {
    API_BASE_URL: process.env.REACT_APP_API_BASE_URL || 'http://your-qnap-ip:3003',
    TENANT_MODE: 'multi',
    DEBUG: false,
    CACHE_TTL: 300000, // 5 minutes
  },
  qnap: {
    API_BASE_URL: 'http://192.168.1.100:3003', // Your QNAP IP
    TENANT_MODE: 'multi',
    DEBUG: false,
    CACHE_TTL: 600000, // 10 minutes
    ENABLE_ANALYTICS: true,
  }
};

export const config = environments[process.env.NODE_ENV] || environments.development;
```

11.3 Health Monitoring & Diagnostics

javascript

```
// utils/healthMonitor.js
class HealthMonitor {
  constructor() {
    this.checks = new Map();
    this.startMonitoring();
  }

  async checkApiHealth() {
    try {
      const response = await fetch(`${config.API_BASE_URL}/health`, {
        method: 'GET',
        timeout: 5000,
      });
      return response.ok;
    } catch (error) {
      console.error('API health check failed:', error);
      return false;
    }
  }

  async checkTenantConfig() {
    const tenant = this.getCurrentTenant();
    return !!(tenant?.id && tenant?.config);
  }

  async performHealthChecks() {
    const results = {
      api: await this.checkApiHealth(),
      tenant: await this.checkTenantConfig(),
      storage: this.checkLocalStorage(),
      timestamp: new Date().toISOString(),
    };

    this.checks.set('latest', results);
    return results;
  }

  checkLocalStorage() {
    try {
      const testKey = '__storage_test__';
      localStorage.setItem(testKey, 'test');
      localStorage.removeItem(testKey);
      return true;
    } catch (error) {
      return false;
    }
  }

  startMonitoring() {
    // Perform health checks every 2 minutes
    setInterval(() => {
      this.performHealthChecks();
    }, 120000);
  }

  getHealthStatus() {
    return this.checks.get('latest') || null;
  }
}

export const healthMonitor = new HealthMonitor();

// Health Status Component
export const HealthStatus = () => {
  const [health, setHealth] = useState(null);

  useEffect(() => {
    const updateHealth = () => {
      setHealth(healthMonitor.getHealthStatus());
    };

    updateHealth();
    const interval = setInterval(updateHealth, 30000); // Update every 30s

    return () => clearInterval(interval);
  }, []);

  if (!health) return null;

  const allHealthy = Object.values(health).every(status =>
    typeof status === 'boolean' ? status : true
  );

  return (
    <div className={`fixed bottom-4 right-4 p-2 rounded-lg text-sm ${
      allHealthy ? 'bg-green-100 text-green-800' : 'bg-red-100 text-red-800'
    }`}>
```

```
      <div className="flex items-center gap-2">
        <div className={`w-2 h-2 rounded-full ${
          allHealthy ? 'bg-green-500' : 'bg-red-500'
        }`} />
        System Status: {allHealthy ? 'Healthy' : 'Issues Detected'}
      </div>
    </div>
  );
};
```

12. Advanced Features

12.1 Real-time Updates with WebSocket

```
javascript
```

```
// services/websocketService.js
class WebSocketService {
 constructor() {
   this.ws = null;
   this.reconnectAttempts = 0;
   this.maxReconnectAttempts = 5;
   this.listeners = new Map();
 }

 connect(tenantId) {
   const wsUrl = `ws://${window.location.host}/ws?tenant=${tenantId}`;
   this.ws = new WebSocket(wsUrl);

   this.ws.onopen = () => {
     console.log('WebSocket connected');
     this.reconnectAttempts = 0;
   };

   this.ws.onmessage = (event) => {
     const data = JSON.parse(event.data);
     this.handleMessage(data);
   };

   this.ws.onclose = () => {
     console.log('WebSocket disconnected');
     this.attemptReconnect(tenantId);
   };

   this.ws.onerror = (error) => {
     console.error('WebSocket error:', error);
   };
 }

 handleMessage(data) {
   const { type, payload } = data;
   const listeners = this.listeners.get(type) || [];
   listeners.forEach(callback => callback(payload));
 }

 subscribe(eventType, callback) {
   if (!this.listeners.has(eventType)) {
     this.listeners.set(eventType, []);
   }
```

```
    this.listeners.get(eventType).push(callback);

    return () => {
      const listeners = this.listeners.get(eventType);
      const index = listeners.indexOf(callback);
      if (index > -1) {
        listeners.splice(index, 1);
      }
    };
  }

  attemptReconnect(tenantId) {
    if (this.reconnectAttempts < this.maxReconnectAttempts) {
      this.reconnectAttempts++;
      setTimeout(() => {
        console.log(`Reconnecting... (${this.reconnectAttempts}/${this.maxReconnectAttempts})`);
        this.connect(tenantId);
      }, 2000 * this.reconnectAttempts);
    }
  }
}

export const wsService = new WebSocketService();

// Real-time Analytics Hook
export const useRealTimeAnalytics = () => {
  const [liveData, setLiveData] = useState(null);
  const { tenant } = useTenant();

  useEffect(() => {
    if (!tenant?.id) return;

    wsService.connect(tenant.id);

    const unsubscribe = wsService.subscribe('analytics_update', (data) => {
      setLiveData(data);
    });

    return () => {
      unsubscribe();
    };
  }, [tenant?.id]);
  return liveData;
};
```

12.2 Advanced Caching with Service Worker

javascript

```
// public/sw.js - Service Worker for advanced caching
const CACHE_NAME = 'etsy-automater-v1';
const TENANT_CACHE_PREFIX = 'tenant-';

// Cache strategies for different resource types
const CACHE_STRATEGIES = {
  tenant_assets: 'cache-first',
  api_data: 'network-first',
  static_assets: 'cache-first',
};

self.addEventListener('install', (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
      return cache.addAll([
        '/',
        '/static/css/main.css',
        '/static/js/main.js',
        '/static/media/logo.svg',
      ]);
    })
  );
});

self.addEventListener('fetch', (event) => {
  const { request } = event;
  const url = new URL(request.url);

  // Handle tenant-specific caching
  if (url.pathname.startsWith('/tenant-assets/')) {
    event.respondWith(cacheFirstStrategy(request));
  }

  // Handle API requests
  else if (url.pathname.startsWith('/api/')) {
    event.respondWith(networkFirstStrategy(request));
  }

  // Handle static assets
  else {
    event.respondWith(cacheFirstStrategy(request));
  }
});
```

```
async function cacheFirstStrategy(request) {
  const cache = await caches.open(CACHE_NAME);
  const cachedResponse = await cache.match(request);

  if (cachedResponse) {
    // Update cache in background
    fetch(request).then(response => {
      if (response.ok) {
        cache.put(request, response.clone());
      }
    });
    return cachedResponse;
  }

  const response = await fetch(request);
  if (response.ok) {
    cache.put(request, response.clone());
  }
  return response;
}

async function networkFirstStrategy(request) {
  try {
    const response = await fetch(request);
    if (response.ok) {
      const cache = await caches.open(CACHE_NAME);
      cache.put(request, response.clone());
    }
    return response;
  } catch (error) {
    const cache = await caches.open(CACHE_NAME);
    const cachedResponse = await cache.match(request);
    return cachedResponse || new Response('Offline', { status: 503 });
  }
}
```

12.3 Progressive Web App Features

javascript

```
// hooks/usePWA.js
import { useState, useEffect } from 'react';

export const usePWA = () => {
  const [isInstallable, setIsInstallable] = useState(false);
  const [installPrompt, setInstallPrompt] = useState(null);
  const [isOnline, setIsOnline] = useState(navigator.onLine);

  useEffect(() => {
    // PWA install prompt
    const handleBeforeInstallPrompt = (e) => {
      e.preventDefault();
      setInstallPrompt(e);
      setIsInstallable(true);
    };

    // Online/offline status
    const handleOnline = () => setIsOnline(true);
    const handleOffline = () => setIsOnline(false);

    window.addEventListener('beforeinstallprompt', handleBeforeInstallPrompt);
    window.addEventListener('online', handleOnline);
    window.addEventListener('offline', handleOffline);

    return () => {
      window.removeEventListener('beforeinstallprompt', handleBeforeInstallPrompt);
      window.removeEventListener('online', handleOnline);
      window.removeEventListener('offline', handleOffline);
    };
  }, []);

  const installApp = async () => {
    if (installPrompt) {
      installPrompt.prompt();
      const result = await installPrompt.userChoice;
      if (result.outcome === 'accepted') {
        setIsInstallable(false);
        setInstallPrompt(null);
      }
    }
  };

  return {
    isInstallable,
    installApp,
    isOnline,
  };
};

// PWA Install Banner Component
export const PWAInstallBanner = () => {
  const { isInstallable, installApp } = usePWA();
  const [dismissed, setDismissed] = useState(
    localStorage.getItem('pwa-banner-dismissed') === 'true'
  );

  if (!isInstallable || dismissed) return null;

  const handleDismiss = () => {
    setDismissed(true);
    localStorage.setItem('pwa-banner-dismissed', 'true');
  };

  return (
    <div className="fixed top-0 left-0 right-0 bg-blue-600 text-white p-4 z-50">
      <div className="flex items-center justify-between max-w-4xl mx-auto">
        <div className="flex items-center gap-3">
          <span>📱</span>
          <span>Install Etsy Automater for quick access!</span>
        </div>
        <div className="flex gap-2">
          <button
            onClick={installApp}
            className="bg-blue-700 px-4 py-2 rounded hover:bg-blue-800"
          >
            Install
          </button>
          <button
            onClick={handleDismiss}
            className="bg-blue-700 px-4 py-2 rounded hover:bg-blue-800"
          >
            ✕
          </button>
        </div>
      </div>
    </div>
  );
};
```

13. Analytics & Performance Tracking

13.1 Custom Analytics Hook

```
javascript
```

```
// hooks/useAnalytics.js
import { useEffect } from 'react';
import { useTenant } from '../contexts/TenantContext';

class AnalyticsService {
  constructor() {
    this.events = [];
    this.sessionId = this.generateSessionId();
  }

  generateSessionId() {
    return `session_${Date.now()}_${Math.random().toString(36).substr(2, 9)}`;
  }

  track(event, properties = {}) {
    const eventData = {
      event,
      properties: {
        ...properties,
        timestamp: new Date().toISOString(),
        sessionId: this.sessionId,
        url: window.location.href,
        userAgent: navigator.userAgent,
      },
    };

    this.events.push(eventData);

    // Send to analytics endpoint
    this.sendToAnalytics(eventData);
  }

  async sendToAnalytics(eventData) {
    try {
      await fetch('/api/analytics', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify(eventData),
      });
    } catch (error) {
      console.error('Analytics tracking failed:', error);
    }
  }

  trackPageView(page, tenant) {
    this.track('page_view', {
      page,
      tenant: tenant?.id,
    });
  }

  trackUserAction(action, data = {}) {
    this.track('user_action', {
      action,
      ...data,
    });
  }

  trackPerformance(metric, value, tenant) {
    this.track('performance', {
      metric,
      value,
      tenant: tenant?.id,
    });
  }
}

const analytics = new AnalyticsService();

export const useAnalytics = () => {
  const { tenant } = useTenant();

  const trackEvent = (event, properties = {}) => {
    analytics.track(event, {
      ...properties,
      tenant: tenant?.id,
    });
  };

  const trackPageView = (page) => {
    analytics.trackPageView(page, tenant);
  };

  const trackUserAction = (action, data = {}) => {
    analytics.trackUserAction(action, {
      ...data,
      tenant: tenant?.id,
    });
  };

  return {
    trackEvent,
    trackPageView,
    trackUserAction,
  };
};

// Performance monitoring hook
export const usePerformanceMonitoring = () => {
  const { tenant } = useTenant();

  useEffect(() => {
    // Track Core Web Vitals
    import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
      getCLS((metric) => {
        analytics.trackPerformance('CLS', metric.value, tenant);
      });

      getFID((metric) => {
        analytics.trackPerformance('FID', metric.value, tenant);
      });

      getFCP((metric) => {
        analytics.trackPerformance('FCP', metric.value, tenant);
      });

      getLCP((metric) => {
        analytics.trackPerformance('LCP', metric.value, tenant);
      });

      getTTFB((metric) => {
        analytics.trackPerformance('TTFB', metric.value, tenant);
      });
    });

    // Track custom performance metrics
    const navigationStart = performance.timing.navigationStart;
    const domContentLoaded = performance.timing.domContentLoadedEventEnd;
    const loadTime = domContentLoaded - navigationStart;
    analytics.trackPerformance('page_load_time', loadTime, tenant);
  }, [tenant]);
};
```

14. Error Handling & Logging

14.1 Global Error Boundary

```
javascript
```

```jsx
// components/common/ErrorBoundary.jsx
import { Component } from 'react';

class ErrorBoundary extends Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false, error: null, errorInfo: null };
  }

  static getDerivedStateFromError(error) {
    return { hasError: true };
  }

  componentDidCatch(error, errorInfo) {
    this.setState({
      error,
      errorInfo,
    });

    // Log error to monitoring service
    this.logError(error, errorInfo);
  }

  logError(error, errorInfo) {
    const errorData = {
      message: error.message,
      stack: error.stack,
      componentStack: errorInfo.componentStack,
      timestamp: new Date().toISOString(),
      url: window.location.href,
      userAgent: navigator.userAgent,
      tenant: this.getTenantFromContext(),
    };

    // Send to error logging service
    fetch('/api/errors', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(errorData),
    }).catch(console.error);
  }
```

```
getTenantFromContext() {
 // Extract tenant from URL or context
 const hostname = window.location.hostname;
 return hostname.split('.')[0];
}

render() {
 if (this.state.hasError) {
  return (
   <div className="min-h-screen flex items-center justify-center bg-gray-50">
    <div className="max-w-md w-full bg-white rounded-lg shadow-md p-6">
     <div className="flex items-center justify-center w-12 h-12 mx-auto bg-red-100 rounded-full mb-4">
      <span className="text-red-600 text-xl">⚠️ </span>
     </div>
     <h1 className="text-xl font-semibold text-center text-gray-900 mb-2">
      Something went wrong
     </h1>
     <p className="text-gray-600 text-center mb-4">
      We're sorry for the inconvenience. The error has been logged and we'll look into it.
     </p>
     <button
      onClick={() => window.location.reload()}
      className="w-full bg-blue-600 text-white py-2 px-4 rounded hover:bg-blue-700"
     >
      Reload Page
     </button>
     {process.env.NODE_ENV === 'development' && (
      <details className="mt-4">
       <summary className="cursor-pointer text-sm text-gray-500">
        Error Details (Dev Mode)
       </summary>
       <pre className="mt-2 text-xs bg-gray-100 p-2 rounded overflow-auto">
        {this.state.error && this.state.error.toString()}
       </pre>
      </details>
     )}
    </div>
   </div>
  );
 }

 return this.props.children;
}
```

```
}

export default ErrorBoundary;
```

14.2 Centralized Logging Service

```
javascript
```

```
// utils/logger.js
class Logger {
  constructor() {
    this.logs = [];
    this.maxLogs = 1000;
  }

  log(level, message, data = {}) {
    const logEntry = {
      level,
      message,
      data,
      timestamp: new Date().toISOString(),
      tenant: this.getCurrentTenant(),
      url: window.location.href,
    };

    this.logs.push(logEntry);

    // Keep only the last N logs
    if (this.logs.length > this.maxLogs) {
      this.logs.shift();
    }

    // Console output for development
    if (process.env.NODE_ENV === 'development') {
      console[level](message, data);
    }

    // Send critical errors to server
    if (level === 'error') {
      this.sendToServer(logEntry);
    }
  }

  getCurrentTenant() {
    const hostname = window.location.hostname;
    return hostname.split('.')[0];
  }

  async sendToServer(logEntry) {
    try {
      await fetch('/api/logs', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify(logEntry),
      });
    } catch (error) {
      console.error('Failed to send log to server:', error);
    }
  }

  info(message, data) {
    this.log('info', message, data);
  }

  warn(message, data) {
    this.log('warn', message, data);
  }

  error(message, data) {
    this.log('error', message, data);
  }

  debug(message, data) {
    this.log('debug', message, data);
  }

  getLogs() {
    return this.logs;
  }

  clearLogs() {
    this.logs = [];
  }
}

export const logger = new Logger();
```

15. Future Enhancements & Scalability

15.1 Micro-Frontend Architecture Preparation

javascript

```
// utils/microfrontendLoader.js
class MicrofrontendLoader {
  constructor() {
    this.loadedMicrofrontends = new Map();
  }

  async loadMicrofrontend(name, url, tenant) {
    const key = `${name}-${tenant}`;

    if (this.loadedMicrofrontends.has(key)) {
      return this.loadedMicrofrontends.get(key);
    }

    try {
      const module = await import(/* @vite-ignore */ `${url}/${name}.js`);
      this.loadedMicrofrontends.set(key, module);
      return module;
    } catch (error) {
      logger.error(`Failed to load microfrontend: ${name}`, { error, tenant });
      throw error;
    }
  }

  async loadTenantSpecificComponent(componentName, tenant) {
    const tenantComponentUrl = `/tenant-components/${tenant}`;

    try {
      return await this.loadMicrofrontend(componentName, tenantComponentUrl, tenant);
    } catch (error) {
      // Fallback to default component
      logger.warn(`Tenant-specific component not found, using default`, {
        component: componentName,
        tenant
      });
      return await this.loadMicrofrontend(componentName, '/default-components', tenant);
    }
  }
}

export const microfrontendLoader = new MicrofrontendLoader();

// Dynamic Component Loader Hook
export const useDynamicComponent = (componentName) => {
```

```
const { tenant } = useTenant();
const [Component, setComponent] = useState(null);
const [loading, setLoading] = useState(true);
const [error, setError] = useState(null);

useEffect(() => {
  let mounted = true;

  const loadComponent = async () => {
    try {
      setLoading(true);
      const module = await microfrontendLoader.loadTenantSpecificComponent(
        componentName,
        tenant?.id
      );

      if (mounted) {
        setComponent(() => module.default);
        setError(null);
      }
    } catch (err) {
      if (mounted) {
        setError(err);
        logger.error('Failed to load dynamic component', {
          component: componentName,
          tenant: tenant?.id,
          error: err
        });
      }
    } finally {
      if (mounted) {
        setLoading(false);
      }
    }
  };

  if (tenant?.id) {
    loadComponent();
  }

  return () => {
    mounted = false;
  };
}, [componentName, tenant?.id]);
```

```
  return { Component, loading, error };
};
```

15.2 CDN Integration for Assets

javascript

```
// utils/assetManager.js
class AssetManager {
  constructor() {
    this.cdnBaseUrl = process.env.REACT_APP_CDN_BASE_URL || '';
    this.assetCache = new Map();
  }

  getTenantAssetUrl(tenant, assetPath) {
    const baseUrl = this.cdnBaseUrl || window.location.origin;
    return `${baseUrl}/tenant-assets/${tenant}/${assetPath}`;
  }

  getOptimizedImageUrl(src, options = {}) {
    const { width, height, quality = 80, format = 'webp' } = options;

    if (!this.cdnBaseUrl) {
      return src; // No CDN, return original
    }

    const params = new URLSearchParams();
    if (width) params.set('w', width);
    if (height) params.set('h', height);
    params.set('q', quality);
    params.set('f', format);

    return `${this.cdnBaseUrl}/optimize?url=${encodeURIComponent(src)}&${params}`;
  }

  async preloadAssets(assets) {
    const promises = assets.map(async (asset) => {
      if (this.assetCache.has(asset.url)) {
        return this.assetCache.get(asset.url);
      }

      const promise = new Promise((resolve, reject) => {
        if (asset.type === 'image') {
          const img = new Image();
          img.onload = () => resolve(asset.url);
          img.onerror = reject;
          img.src = asset.url;
        } else {
          // For other asset types, use fetch
          fetch(asset.url)
        .then(response => response.ok ? resolve(asset.url) : reject())
        .catch(reject);
      }
    });

    this.assetCache.set(asset.url, promise);
    return promise;
  });

  return Promise.allSettled(promises);
 }
}

export const assetManager = new AssetManager();

// Optimized Image Component
export const OptimizedImage = ({
 src,
 alt,
 width,
 height,
 className,
 lazy = true,
 ...props
}) => {
 const [imageSrc, setImageSrc] = useState(src);
 const [isLoading, setIsLoading] = useState(true);
 const [hasError, setHasError] = useState(false);
 const imgRef = useRef();

 useEffect(() => {
  // Generate optimized image URL
  const optimizedSrc = assetManager.getOptimizedImageUrl(src, {
   width,
   height,
  });

  setImageSrc(optimizedSrc);
 }, [src, width, height]);

 const handleLoad = () => {
  setIsLoading(false);
 };
```

```
  const handleError = () => {
    setHasError(true);
    setIsLoading(false);
    // Fallback to original image
    setImageSrc(src);
  };

  return (
    <div className={`relative ${className}`}>
      {isLoading && (
        <div className="absolute inset-0 bg-gray-200 animate-pulse rounded" />
      )}
      <img
        ref={imgRef}
        src={imageSrc}
        alt={alt}
        loading={lazy ? 'lazy' : 'eager'}
        onLoad={handleLoad}
        onError={handleError}
        className={`${className} ${isLoading ? 'opacity-0' : 'opacity-100'} transition-opacity duration-300`}
        {...props}
      />
      {hasError && (
        <div className="absolute inset-0 flex items-center justify-center bg-gray-100 text-gray-500">
          Image not available
        </div>
      )}
    </div>
  );
};
```

16. Summary & Best Practices

This comprehensive frontend architecture provides:

✅ Multi-Tenant Capabilities:

- Dynamic tenant detection and routing
- Isolated tenant contexts and theming
- Tenant-specific asset management

✅ Performance Optimization:

- Code splitting and lazy loading

- Advanced caching strategies
- Service Worker implementation
- Optimized asset delivery

✅ **Scalability Features:**

- Micro-frontend ready architecture
- CDN integration support
- Progressive Web App capabilities
- Real-time updates via WebSocket

✅ **Production Readiness:**

- Comprehensive error handling
- Monitoring and analytics
- Health checks and diagnostics
- Security best practices

✅ **QNAP Integration:**

- Docker containerization
- Nginx configuration for multi-tenant routing
- Resource optimization for NAS deployment
- Easy migration path to cloud platforms

This architecture ensures your Etsy Seller Automater frontend can scale from a local QNAP deployment to a full cloud-based multi-tenant SaaS platform while maintaining excellent performance and user experience.