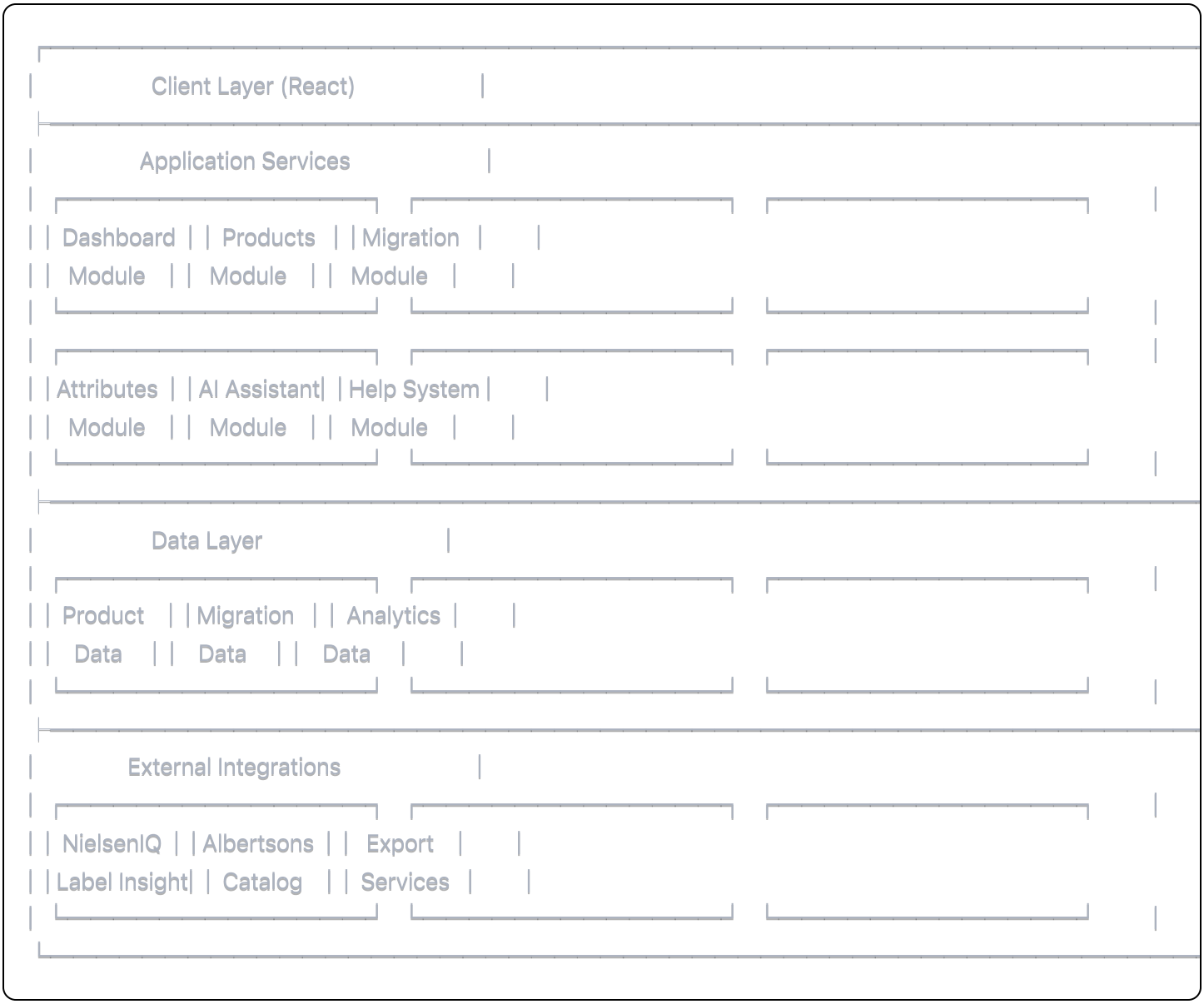
NielsenIQ Label Insight Platform - Technical Architecture

System Architecture Overview

High-Level Architecture



Frontend Architecture

Component Structure

```
src/
├── App.js                    # Main application component
├── components/
│   ├── Dashboard/
│   │   ├── MetricCard.js         # Individual metric display
│   │   ├── PerformanceInsights.js # Store brand performance
│   │   └── ActivityFeed.js       # Recent activity display
│   ├── Products/
│   │   ├── ProductCard.js        # Product information card
│   │   ├── SearchBar.js          # Search and filter interface
│   │   ├── AttributeModal.js     # Detailed attribute view
│   │   └── ImportModal.js        # Product import interface
│   ├── Migration/
│   │   ├── ProgressOverview.js   # Migration status summary
│   │   ├── CategoryProgress.js   # Individual category tracking
│   │   └── ActivityLog.js        # Migration activity timeline
│   ├── Attributes/
│   │   ├── AttributeCards.js     # Attribute distribution cards
│   │   ├── QualityScores.js      # Circular quality indicators
│   │   └── TrendingAttributes.js # Trending analysis
│   ├── AI/
│   │   ├── AiAssistant.js        # Chat interface
│   │   ├── MessageBubble.js      # Individual message display
│   │   └── ResponseEngine.js     # AI response logic
│   └── Help/
│       ├── HelpModal.js          # Comprehensive help system
│       ├── Tooltip.js            # Contextual tooltips
│       └── InfoIcon.js           # Help icon component
├── hooks/
│   ├── useProductSearch.js       # Product search logic
│   ├── useDataExport.js          # Export functionality
│   └── useAiChat.js              # AI chat management
├── utils/
│   ├── dataProcessing.js         # Data transformation utilities
│   ├── exportUtils.js            # Export helper functions
│   └── searchUtils.js            # Search algorithm utilities
└── data/
    ├── productData.js            # Sample product dataset
    ├── migrationData.js          # Migration status data
    └── helpContent.js            # Help system content
```

State Management Architecture

```javascript
// Global Application State
const AppState = {
  // Navigation
  activeTab: 'dashboard',

  // Product Management
  products: [...],
  filteredProducts: [...],
  searchTerm: '',
  selectedCategory: 'All',
  selectedBrand: 'All',
  expandedProduct: null,
  selectedProduct: null,

  // Modal Management
  showAttributeModal: false,
  showImportModal: false,
  showHelpModal: false,

  // AI Assistant
  aiChatOpen: false,
  aiMessages: [...],
  aiInput: '',

  // Help System
  tooltip: {
    show: false,
    content: '',
    position: { x: 0, y: 0 }
  }
};
```

Component Design Patterns

1. Container/Presentation Pattern

```javascript
```

```
// Container Component (Logic)
const ProductCatalogContainer = () => {
  const [products, setProducts] = useState([]);
  const [filters, setFilters] = useState({});

  const filteredProducts = useProductSearch(products, filters);

  return (
    <ProductCatalogPresentation
      products={filteredProducts}
      onFilterChange={setFilters}
      onExport={handleExport}
    />
  );
};

// Presentation Component (UI)
const ProductCatalogPresentation = ({ products, onFilterChange, onExport }) => {
  return (
    <div>
      <SearchInterface onFilterChange={onFilterChange} />
      <ProductGrid products={products} />
      <ExportControls onExport={onExport} />
    </div>
  );
};
```

2. Custom Hooks Pattern

javascript

```
// Product Search Hook
const useProductSearch = (products, filters) => {
  return useMemo(() => {
    return products.filter(product => {
      const matchesSearch = product.name.toLowerCase()
        .includes(filters.searchTerm?.toLowerCase() || '');
      const matchesCategory = filters.category === 'All' ||
        product.category === filters.category;
      const matchesBrand = filters.brand === 'All' ||
        product.brand === filters.brand;

      return matchesSearch && matchesCategory && matchesBrand;
    });
  }, [products, filters]);
};

// Export Functionality Hook
const useDataExport = () => {
  const exportJSON = useCallback((data, filename) => {
    const dataStr = JSON.stringify(data, null, 2);
    const dataUri = 'data:application/json;charset=utf-8,'+
      encodeURIComponent(dataStr);
    downloadFile(dataUri, filename);
  }, []);

  const exportCSV = useCallback((data, filename) => {
    const csvContent = convertToCSV(data);
    const dataUri = 'data:text/csv;charset=utf-8,'+
      encodeURIComponent(csvContent);
    downloadFile(dataUri, filename);
  }, []);

  return { exportJSON, exportCSV };
};
```

Data Architecture

Product Data Model

typescript

```typescript
interface Product {
  // Basic Information
  id: string;                 // UPC or SKU identifier
  name: string;               // Product name
  brand: string;              // Brand name
  category: string;           // Primary category
  subCategory: string;        // Secondary category
  price: number;              // Current price
  size: string;               // Package size

  // Data Quality
  coverage: number;           // Data completeness percentage
  lastUpdated: string;        // ISO date string
  migrationStatus: 'Complete' | 'In Progress' | 'Pending';
  dataQuality: number;        // Quality score 0-100

  // Claims and Certifications
  claims: string[];           // Marketing claims
  certifications: string[];   // Third-party certifications

  // Nutritional Information
  nutrition: {
    calories: number;
    protein: number;          // grams
    fiber: number;            // grams
    sugar: number;            // grams
    sodium: number;           // milligrams
    fat: number;              // grams
    carbs: number;            // grams
  };

  // Comprehensive Attributes
  fullAttributes: {
    'Dietary & Lifestyle': Record<string, boolean>;
    'Health & Nutrition': Record<string, boolean>;
    'Clean Label': Record<string, boolean>;
    'Sourcing & Production': Record<string, boolean>;
  };

  // Additional Information
  allergens: string[];        // Allergen information
```

```
  tags: string[];             // Search tags
}
```

Migration Data Model

typescript

```
interface MigrationData {
  overview: {
    totalProducts: number;
    processed: number;
    inProgress: number;
    pending: number;
    overallProgress: number;    // Percentage
  };

  categories: Array<{
    name: string;
    progress: number;          // Percentage
    total: number;
    completed: number;
    status: 'Complete' | 'In Progress' | 'Pending';
  }>;

  recentActivity: Array<{
    time: string;              // ISO date string
    action: string;            // Description of activity
    type: 'success' | 'warning' | 'info' | 'error';
  }>;
}
```

Analytics Data Model

typescript

```
interface AttributeAnalytics {
  attributeDistribution: Array<{
    name: string;
    count: number;
    percentage: number;
    color: string;
  }>;

  qualityMetrics: Array<{
    category: 'Completeness' | 'Accuracy' | 'Freshness' | 'Consistency';
    score: number;              // 0-100
    color: string;
  }>;

  trendingAttributes: Array<{
    name: string;
    trend: string;              // e.g., "+34%"
    count: number;
    color: string;
    trendType: 'up' | 'down';
  }>;
}
```

AI Assistant Architecture

Response Engine

javascript

```
class AIResponseEngine {
 constructor() {
  this.knowledgeBase = {
   coverage: "Coverage indicates how complete a product's data is...",
   attributes: "Product attributes include Dietary & Lifestyle...",
   search: "Use the search bar to find products by name...",
   export: "Export your product data in JSON format...",
   migration: "Data migration shows progress of integrating...",
   help: "I can help you with: Product coverage and quality..."
  };
 }

 generateResponse(userInput) {
  const normalizedInput = userInput.toLowerCase();

  // Keyword matching algorithm
  const keywords = this.extractKeywords(normalizedInput);
  const matchedResponses = this.matchKeywords(keywords);

  if (matchedResponses.length > 0) {
   return this.selectBestResponse(matchedResponses);
  }

  return this.getDefaultResponse();
 }

 extractKeywords(input) {
  const stopWords = ['the', 'is', 'at', 'which', 'on', 'what', 'how'];
  return input.split(/\s+/)
   .filter(word => !stopWords.includes(word))
   .filter(word => word.length > 2);
 }

 matchKeywords(keywords) {
  const matches = [];

  for (const [topic, response] of Object.entries(this.knowledgeBase)) {
   const matchScore = this.calculateMatchScore(keywords, topic);
   if (matchScore > 0.3) {
    matches.push({ topic, response, score: matchScore });
   }
  }
```

```
    return matches.sort((a, b) => b.score - a.score);
  }
}
```

Chat Interface Architecture

javascript

```
const ChatInterface = {
  // Message Types
  MessageTypes: {
    USER: 'user',
    AI: 'ai',
    SYSTEM: 'system'
  },

  // Chat State Management
  state: {
    messages: [],
    isTyping: false,
    isOpen: false
  },

  // Message Processing Pipeline
  processMessage: async (userInput) => {
    // 1. Add user message
    this.addMessage(userInput, MessageTypes.USER);

    // 2. Show typing indicator
    this.setTyping(true);

    // 3. Generate AI response
    const response = await this.generateResponse(userInput);

    // 4. Add AI response
    this.addMessage(response, MessageTypes.AI);

    // 5. Hide typing indicator
    this.setTyping(false);
  }
};
```

Performance Optimization

Code Splitting Strategy

javascript

```
// Lazy loading for route-based components
const Dashboard = lazy(() => import('./components/Dashboard'));
const Products = lazy(() => import('./components/Products'));
const Migration = lazy(() => import('./components/Migration'));
const Attributes = lazy(() => import('./components/Attributes'));

// Dynamic imports for heavy features
const loadAIAssistant = () => import('./components/AI/AiAssistant');
const loadAttributeModal = () => import('./components/Products/AttributeModal');
```

Memoization Strategy

javascript

```
// Expensive calculations memoized
const MemoizedProductCard = memo(ProductCard, (prevProps, nextProps) => {
  return (
    prevProps.product.id === nextProps.product.id &&
    prevProps.isExpanded === nextProps.isExpanded
  );
});

// Search results memoized
const useProductSearch = (products, filters) => {
  return useMemo(() => {
    return products.filter(product => {
      // Filter logic here
    });
  }, [products, filters.searchTerm, filters.category, filters.brand]);
};
```

Virtual Scrolling for Large Datasets

javascript

```
const VirtualizedProductList = ({ products }) => {
  const [visibleRange, setVisibleRange] = useState({ start: 0, end: 50 });

  const visibleProducts = useMemo(() => {
    return products.slice(visibleRange.start, visibleRange.end);
  }, [products, visibleRange]);

  return (
    <div onScroll={handleScroll}>
      {visibleProducts.map(product => (
        <ProductCard key={product.id} product={product} />
      ))}
    </div>
  );
};
```

Security Implementation

Input Validation

javascript

```javascript
// Search input sanitization
const sanitizeSearchInput = (input) => {
  return input
    .replace(/[<>\"']/g, '') // Remove potential XSS characters
    .trim()
    .substring(0, 100); // Limit length
};

// File upload validation
const validateUploadFile = (file) => {
  const allowedTypes = ['application/json', 'text/csv'];
  const maxSize = 10 * 1024 * 1024; // 10MB

  if (!allowedTypes.includes(file.type)) {
    throw new Error('Invalid file type');
  }

  if (file.size > maxSize) {
    throw new Error('File too large');
  }

  return true;
};
```

Data Privacy

javascript

```javascript
// Personal data anonymization
const anonymizeProductData = (product) => {
  return {
    ...product,
    // Remove any personally identifiable information
    customerId: undefined,
    customerEmail: undefined,
    // Hash sensitive identifiers
    supplierId: hashValue(product.supplierId)
  };
};
```

Testing Strategy

Unit Testing

javascript

```javascript
// Component testing with React Testing Library
import { render, screen, fireEvent } from '@testing-library/react';
import ProductCard from './ProductCard';

describe('ProductCard', () => {
  test('displays product information correctly', () => {
    const mockProduct = {
      id: 'UPC123',
      name: 'Test Product',
      brand: 'Test Brand',
      price: 4.99
    };

    render(<ProductCard product={mockProduct} />);

    expect(screen.getByText('Test Product')).toBeInTheDocument();
    expect(screen.getByText('$4.99')).toBeInTheDocument();
  });

  test('expands when More button is clicked', () => {
    render(<ProductCard product={mockProduct} />);

    fireEvent.click(screen.getByText('More'));

    expect(screen.getByText('Nutritional Profile')).toBeInTheDocument();
  });
});
```

Integration Testing

javascript

```
// API integration testing
describe('Product Search Integration', () => {
  test('filters products by search term', async () => {
    const { getByPlaceholderText, getAllByTestId } = render(<ProductCatalog />);

    fireEvent.change(getByPlaceholderText('Search products...'), {
      target: { value: 'organic' }
    });

    await waitFor(() => {
      const productCards = getAllByTestId('product-card');
      expect(productCards).toHaveLength(2); // Expected organic products
    });
  });
});
```

Performance Testing

javascript

```
// Load testing configuration
const performanceConfig = {
  scenarios: {
    product_search: {
      executor: 'ramping-vus',
      startVUs: 10,
      stages: [
        { duration: '2m', target: 50 },
        { duration: '5m', target: 100 },
        { duration: '2m', target: 0 }
      ]
    }
  },
  thresholds: {
    http_req_duration: ['p(95)<500'], // 95% of requests under 500ms
    http_req_failed: ['rate<0.1']     // Error rate under 10%
  }
};
```

Deployment Architecture

Build Configuration

javascript

```javascript
// webpack.config.js
module.exports = {
  entry: './src/index.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: '[name].[contenthash].js',
    clean: true
  },
  optimization: {
    splitChunks: {
      chunks: 'all',
      cacheGroups: {
        vendor: {
          test: /[\\/]node_modules[\\/]/,
          name: 'vendors',
          chunks: 'all'
        }
      }
    }
  },
  plugins: [
    new HtmlWebpackPlugin({
      template: './public/index.html'
    }),
    new MiniCssExtractPlugin({
      filename: '[name].[contenthash].css'
    })
  ]
};
```

Environment Configuration

javascript

```
// Environment-specific configurations
const config = {
  development: {
    API_BASE_URL: 'http://localhost:3001/api',
    LOG_LEVEL: 'debug',
    ENABLE_MOCK_DATA: true
  },
  staging: {
    API_BASE_URL: 'https://staging-api.nielseniq.com/api',
    LOG_LEVEL: 'info',
    ENABLE_MOCK_DATA: false
  },
  production: {
    API_BASE_URL: 'https://api.nielseniq.com/api',
    LOG_LEVEL: 'error',
    ENABLE_MOCK_DATA: false,
    ENABLE_ANALYTICS: true
  }
};
```

Monitoring and Logging

javascript

```
// Application monitoring setup
const monitoring = {
  errorTracking: {
    service: 'Sentry',
    dsn: process.env.SENTRY_DSN,
    environment: process.env.NODE_ENV
  },

  analytics: {
    service: 'Google Analytics',
    measurementId: process.env.GA_MEASUREMENT_ID
  },

  performance: {
    service: 'Web Vitals',
    thresholds: {
      LCP: 2500,  // Largest Contentful Paint
      FID: 100,   // First Input Delay
      CLS: 0.1    // Cumulative Layout Shift
    }
  }
};
```

Document Version: 1.0

Last Updated: January 22, 2025

Classification: Technical Documentation

Owner: Engineering Team