Amna Bootcamp Third Assignment: Markiting

Link to the website/vercel: Link

Starter Code Explanation:

App Folder:

The app/ directory is the core of a modern Next.js 13+ app.

It replaces the old pages/ system and introduces App Router, which lets you define routes, layouts, and even API endpoints in a cleaner way.

• Every folder under app/ becomes a **route**.

• Every page.tsx inside a folder becomes the **UI** for that route.

• Every layout.tsx defines how all those pages share global stuff (like navbar, fonts, metadata).

Why api/ is under app/?

That's how Next.js handles **server routes**.
Instead of having a separate backend folder, The APIs live under `app/api/...`.

This one fetches data from an external endpoint (`https://www.amanabootcamp.org/api/fs-classwork-data/...`) and passes it to the app.
It acts as a *middleware layer* between the frontend (React) and the outside data source.

Why there are multiple "views" under `app/`

The **dashboard** is built with different analytic views:

- `/campaign-view` → campaign performance
- `/demographic-view` → user demographics
- `/region-view` → performance by region
- `/weekly-view` → weekly insights

Each `page.tsx` is a self-contained page using shared UI components (from `src/components/ui/`).
For example, all pages use `<Navbar />`, `<Footer />`, and visual components like `<BarChart />` or `<CardMetric />`, so they all look consistent.

How it all ties together

Here’s the flow:

1. You open `/campaign-view`
2. That page calls `fetchMarketingData()` from `src/lib/api.ts`
3. That function fetches data from `/api/marketing-data`
4. The API route in `app/api/marketing-data/route.ts` calls the **external** API
5. Data comes back to the frontend view → rendered using React + Tailwind

Files under app/

- The global wrapper(layout.tsx):
  It’s where you set things that never change between pages, fonts, metadata, global layout, etc.
- App-wide styling(globals.css):
  This file contains **global CSS rules** and imports Tailwind.
  It defines the dark/light themes, base colors, and font variables.
- The homepage(page.tsx):
  This is the **main dashboard overview.**
- The backend endpoint(api/marketing-data/route.ts):
  This is the **server-side API route**.
  It handles fetching raw marketing data from an external source.

Public Folder:

- **Reusable UI building blocks(src/components/ui/):**
  This folder is the design system, every React component here is reusable and stylized using Tailwind CSS.
  i) navbar.tsx: The sidebar + top navigation. Handles routes (Overview, Campaign View, etc.), collapse/expand behavior, and active link highlighting.
  ii) footer.tsx: The bottom bar (copyright © 2025 Amana Bootcamp). Simple, shared across pages.
  iii) card-metric.tsx: A component that displays key metrics like revenue, conversions.

iv) bar-chart.tsx: Custom chart component, Creates dynamic bar charts without depending on external chart libraries.
v) dropdown-filter.tsx: Multi-select dropdown for filtering campaigns by type, region, etc. Uses lucide-react icons and internal state to manage options.
vi) search-filter.tsx: Text input with a search icon and clear (X) button, used to filter data lists.
vii) table.tsx: Fully featured table component with sorting, alignment, and pagination support. Used in your campaign list section.

- **src/lib/ → Helper functions, APIs, and backend utilities:**
  i) api.ts:
  This is the frontend API handler. It decides how the app fetches data from the backend
- **src/types/ → Data structure definitions (TypeScript interfaces)**
  i) This folder makes the TypeScript project strongly typed, no guessing what shape the data is.

Other Files:

- **`next.config.ts`** — Next.js configuration
  This is the main app config file, basically the "Next.js settings center."
  You can define:
  - Environment variables
  - Image domains
  - Experimental features
  - Custom build behavior
- **package.json — Dependency & script manager**
  This file defines everything about the project, Which npm packages are installed (react, next, tailwindcss, etc.)
- **tsconfig.json — TypeScript compiler config**
  Controls how TypeScript behaves — what rules it follows, where to find source files, etc.
- **postcss.config.mjs — Tailwind + PostCSS setup**
  This controls how the CSS is processed during the build.

Marketing Dashboard Views Documentation

Overview

This document outlines the implementation of four key marketing analytics views in the Next.js marketing dashboard application. Each view provides different perspectives on campaign performance data, utilizing various visualization components and data aggregation techniques.

i) Demographic View (/demographic-view)

Purpose

Analyzes marketing campaign performance across different demographic segments, focusing on gender and age group breakdowns.

Key Features Implemented

Data Aggregation

- **Gender-based metrics**: Aggregates clicks, spend, and revenue for male vs female audiences across all campaigns
- **Age group analysis**: Groups performance data by age ranges (18-24, 25-34, etc.)
- **Proportional allocation**: Distributes total campaign spend and revenue based on click distribution ratios

Visual Components

- **Card Metrics Grid**: 6 overview cards showing gender-specific performance metrics
  - Total clicks by males/females
  - Total spend by males/females
  - Total revenue by males/females
- **Bar Charts**: Two side-by-side charts displaying spend and revenue by age group
- **Data Tables**: Two sortable tables showing detailed performance by male and female age groups

Table Features

- **Columns**: Age Group, Impressions, Clicks, Conversions, CTR, Conversion Rate
- **Sorting**: All columns are sortable with custom sort types
- **Color Coding**: Different colors for metrics (blue for clicks, green for conversions, etc.)
- **Responsive Design**: Tables with horizontal scrolling and fixed heights

Technical Implementation

- Uses useMemo for efficient data aggregation
- Handles missing data gracefully with fallback values
- Responsive grid layouts that adapt to screen sizes

- Consistent error handling and loading states

2. Weekly View (/weekly-view)

Purpose

Shows temporal trends in marketing campaign performance over weekly intervals, displaying revenue and spend patterns over time.

Key Features Implemented

Data Processing

- **Weekly aggregation**: Combines performance data from all campaigns by week ranges
- **Time-based sorting**: Automatically sorts weeks chronologically by start date
- **Dual metrics tracking**: Tracks both spend and revenue for each week

Visual Components

- **Line Chart**: Single comprehensive chart showing both revenue and spend trends
  - Revenue line (green color)
  - Spend line (blue color)
  - Clean week labels showing only start dates
  - Currency formatting for all values

Chart Configuration

- **Series Configuration**: Two data series with distinct colors and labels
- **Responsive Design**: Chart adapts to container width
- **Interactive Tooltips**: Hover effects showing exact values
- **Clean Labeling**: Simplified week labels for better readability

Technical Implementation

- Efficient data aggregation using useMemo
- Date-based sorting with proper chronological ordering
- Currency formatting with locale-specific number formatting
- Minimal, focused visualization approach

3. Region View (/region-view)

Purpose

Provides geographical analysis of marketing campaign performance using interactive maps with real-world coordinates.

Key Features Implemented

Geographical Data

- **Real coordinates**: Uses actual latitude/longitude coordinates for major cities worldwide

- **UAE focus**: Centered on UAE with comprehensive city coverage
- **International scope**: Includes major cities from Middle East, Europe, Asia, and Americas

Interactive Map Visualization

- **Leaflet Integration**: Full-featured interactive maps with OpenStreetMap tiles
- **Dual heat maps**: Separate maps for revenue and spend visualization
- **Dynamic circles**: Circle sizes proportional to performance values
- **Interactive features**:
  - Clickable popups with detailed region information
  - Hover tooltips showing region names and values
  - Zoom and pan controls
  - Auto-fitting bounds to show all data points

Map Components

- **Circle overlays**: Color-coded circles representing performance intensity
- **Marker labels**: Custom markers with region names
- **Popup content**: Rich popups showing region, country, and formatted values
- **Responsive design**: Maps adapt to different screen sizes

Technical Implementation

- **SSR compatibility**: Dynamic imports to avoid server-side rendering issues
- **Client-side rendering**: Maps only render after Leaflet loads
- **Memory management**: Proper cleanup of map instances
- **Error handling**: Graceful fallbacks for missing coordinates

4. Device View (/device-view)

Purpose

Compares marketing campaign performance between mobile and desktop devices, highlighting platform-specific engagement patterns.

Key Features Implemented

Device Performance Metrics

- **Comprehensive metrics**: Tracks impressions, clicks, conversions, spend, revenue, CTR, conversion rate, and traffic percentage
- **Cross-device comparison**: Side-by-side analysis of mobile vs desktop performance
- **Aggregated data**: Combines performance across all campaigns by device type

Visual Components

- **Card Metrics Grid**: 16 metric cards (8 for mobile, 8 for desktop)
  - Organized in 2x4 grids for each device type
  - Color-coded sections (blue for mobile, green for desktop)
  - Appropriate icons for each metric type

Performance Comparison

- **Ratio calculations**: Shows percentage ratios comparing mobile vs desktop performance
- **Key metrics comparison**: Clicks, conversions, and revenue ratios
- **Visual indicators**: Color-coded percentage displays

Layout Design

- **Responsive grid**: Adapts from single column on mobile to 4-column layout on desktop
- **Section headers**: Clear device identification with icons
- **Comparison summary**: Dedicated section showing performance ratios
- **Professional styling**: Consistent with overall dashboard theme

Technical Implementation

- **Data aggregation**: Efficiently combines device performance across campaigns
- **Metric recalculation**: Computes CTR and conversion rates from aggregated data
- **Responsive design**: Flexible grid layouts for different screen sizes
- **Type safety**: Full TypeScript support with proper interfaces

Navigation Updates

Sidebar Enhancement

- Added "Device View" navigation item with Monitor icon
- Updated navigation array to include the new route
- Maintained consistent styling and interaction patterns

Technical Architecture

Shared Patterns

- **Consistent structure**: All views follow the same Next.js App Router pattern
- **Error handling**: Uniform error display and loading states
- **Responsive design**: Mobile-first approach with progressive enhancement
- **Data fetching:** Centralized API integration with proper error handling
- **Type safety**: Full TypeScript implementation with proper interfaces

Component Reuse

- **CardMetric**: Reused across views for consistent metric display

- **Chart components**: LineChart, BarChart, and HeatMap with standardized APIs
- **Table component**: Consistent data table implementation with sorting
- **Layout components**: Shared Navbar and Footer across all views

Performance Optimizations

- **Memoization**: useMemo hooks for expensive data transformations
- **Lazy loading**: Dynamic imports for heavy components (Leaflet maps)
- **Efficient rendering**: Optimized re-renders with proper dependency arrays
- **Memory management**: Proper cleanup of interactive components

Data Flow Architecture

API Integration

- **Centralized fetching**: Single fetchMarketingData function
- **Error boundaries**: Comprehensive error handling at component level
- **Loading states**: User-friendly loading indicators
- **Data validation**: Type-safe data structures with proper interfaces

State Management

- **Local state**: Component-level state for UI interactions
- **Derived state**: Computed metrics using useMemo for performance
- **Effect management**: Proper cleanup and dependency handling