```typescript
// app/types/stock.ts


interface ProfileData {
    name: string;
    exchange: string;
    mic_code: string;
    sector: string;
    industry: string;
    employees: number;
    website: string;
    description: string;
    type: string;
    CEO: string;
    address: string;
    address2: string;
    city: string;
    zip: string;
    state: string;
    country: string;
    phone: string;
  }

// Quote data interface
export interface QuoteData {
    symbol: string;
    name: string;
    close: string;
    change: string;
    percent_change: string;
    high: string;
    low: string;
    open: string;
    volume: string;
  }

  // Time series data interface
  export interface TimeSeriesData {
    datetime: string;
    open: string;
    high: string;
    low: string;
    close: string;
    volume: string;
  }

  // Valuation metrics interface
  export interface ValuationMetrics {
    market_capitalization: number;    // Market cap
    enterprise_value: number;          // Enterprise value (EV)
    trailing_pe: number;              // Price to trailing earnings ratio (TTM)
    forward_pe: number;               // Price to forward earnings ratio
    peg_ratio: number;                // P/E to growth ratio
    price_to_sales_ttm: number;       // Price to sales ratio (TTM)
    price_to_book_mrq: number;        // Price to book ratio
    enterprise_to_revenue: number;     // Enterprise value to revenue
    enterprise_to_ebitda: number;     // Enterprise value to EBITDA
  }

  // Financial statement interfaces
  export interface IncomeStatement {
    revenue_ttm: number;                 // Trailing 12m revenue
    revenue_per_share_ttm: number;     // Revenue per share
    quarterly_revenue_growth: number;  // QoQ revenue growth
    gross_profit_ttm: number;          // Gross profit
    ebitda: number;                    // EBITDA
    net_income_to_common_ttm: number;  // Net income
    diluted_eps_ttm: number;           // Diluted EPS
    quarterly_earnings_growth_yoy: number; // YoY earnings growth
  }

  export interface BalanceSheet {
    total_cash_mrq: number;            // Total cash
    total_cash_per_share_mrq: number;  // Cash per share
    total_debt_mrq: number;            // Total debt
    total_debt_to_equity_mrq: number;  // Debt/Equity ratio
    current_ratio_mrq: number;         // Current ratio
    book_value_per_share_mrq: number;  // Book value per share
  }

  export interface CashFlow {
    operating_cash_flow_ttm: number;    // Operating cash flow
    levered_free_cash_flow_ttm: number; // Free cash flow
  }

  export interface Financials {
```

```typescript
    fiscal_year_ends: string;           // Last 12-month period end
    most_recent_quarter: string;        // Most recent quarter end
    gross_margin: number;               // Gross margin
    profit_margin: number;              // Net profit margin
    operating_margin: number;           // Operating margin
    return_on_assets_ttm: number;       // Return on assets
    return_on_equity_ttm: number;       // Return on equity
    income_statement: IncomeStatement;
    balance_sheet: BalanceSheet;
    cash_flow: CashFlow;
  }

  // Stock statistics interfaces
  export interface StockStatistics {
    shares_outstanding: number;
    float_shares: number;
    avg_10_volume: number;
    avg_90_volume: number;
  }

  export interface StockPriceSummary {
    fifty_two_week_low: number;
    fifty_two_week_high: number;
    fifty_two_week_change: number;
    beta: number;
    day_50_ma: number;
    day_200_ma: number;
  }

  export interface DividendData {
    forward_annual_dividend_rate: number;
    forward_annual_dividend_yield: number;
    trailing_annual_dividend_rate: number;
    trailing_annual_dividend_yield: number;
    "5_year_average_dividend_yield": number;
    payout_ratio: number;
    dividend_date: string;
    ex_dividend_date: string;
  }

  // Main stock data interface
  export interface StockData {
    quote: {
    symbol: string;
    name: string;
    close: string;
    change: string;
    percent_change: string;
    high: string;
    low: string;
    open: string;
    volume: string;
  };
  timeSeries: {
    values: Array<{
      datetime: string;
      open: string;
      high: string;
      low: string;
      close: string;
      volume: string;
    }>;
  };
  statistics: any;
  profile?: ProfileData};
//     statistics: {
//       valuations_metrics: ValuationMetrics;
//       financials: Financials;
//       stock_statistics: StockStatistics;
//       stock_price_summary: StockPriceSummary;
//       dividends_and_splits: DividendData;
//     };
//   }

  // Helper interface for metric display
  export interface MetricItem {
    label: string;
    value: number | string | null;
    format?: 'number' | 'percent' | 'currency' | 'compact';
    description?: string;
  }

  // Helper function to format numbers based on type
  export function formatValuationMetric(
    value: number | undefined | null,
    format: 'number' | 'percent' | 'currency' | 'compact' = 'number'
```

```typescript
): string {
  if (value === null || value === undefined || isNaN(value)) return '\u2014';

  switch (format) {
    case 'number':
      return value.toFixed(2);
    case 'percent':
      return `${(value * 100).toFixed(2)}%`;
    case 'currency':
      return new Intl.NumberFormat('en-US', {
        style: 'currency',
        currency: 'SAR',
        minimumFractionDigits: 0,
        maximumFractionDigits: 0
      }).format(value);
    case 'compact':
      return new Intl.NumberFormat('en-US', {
        notation: 'compact',
        compactDisplay: 'short'
      }).format(value);
    default:
      return value.toString();
  }
}

// Get formatted valuation metrics for display
export function getValuationMetricsDisplay(metrics: ValuationMetrics): Array<MetricItem> {
  return [
    {
      label: 'P/E (TTM)',
      value: metrics.trailing_pe,
      format: 'number',
      description: 'Price to trailing earnings ratio'
    },
    {
      label: 'Forward P/E',
      value: metrics.forward_pe,
      format: 'number',
      description: 'Price to forward earnings ratio'
    },
    {
      label: 'PEG Ratio',
      value: metrics.peg_ratio,
      format: 'number',
      description: 'P/E ratio divided by growth rate'
    },
    {
      label: 'P/S (TTM)',
      value: metrics.price_to_sales_ttm,
      format: 'number',
      description: 'Price to sales ratio'
    },
    {
      label: 'P/B',
      value: metrics.price_to_book_mrq,
      format: 'number',
      description: 'Price to book ratio'
    },
    {
      label: 'EV/EBITDA',
      value: metrics.enterprise_to_ebitda,
      format: 'number',
      description: 'Enterprise value to EBITDA ratio'
    }
  ];
}

// Get formatted margin metrics for display
export function getMarginMetricsDisplay(financials: Financials): Array<MetricItem> {
  return [
    {
      label: 'Gross Margin',
      value: financials.gross_margin,
      format: 'percent',
      description: 'Gross profit as a percentage of revenue'
    },
    {
      label: 'Operating Margin',
      value: financials.operating_margin,
      format: 'percent',
      description: 'Operating income as a percentage of revenue'
    },
    {
      label: 'Profit Margin',
      value: financials.profit_margin,
      format: 'percent',
```

```typescript
      description: 'Net income as a percentage of revenue'
    }
  ];
}

// Validation helper function for valuation metrics
export function validateValuationMetrics(data: any): ValuationMetrics {
  return {
    market_capitalization: Number(data?.market_capitalization) || 0,
    enterprise_value: Number(data?.enterprise_value) || 0,
    trailing_pe: Number(data?.trailing_pe) || 0,
    forward_pe: Number(data?.forward_pe) || 0,
    peg_ratio: Number(data?.peg_ratio) || 0,
    price_to_sales_ttm: Number(data?.price_to_sales_ttm) || 0,
    price_to_book_mrq: Number(data?.price_to_book_mrq) || 0,
    enterprise_to_revenue: Number(data?.enterprise_to_revenue) || 0,
    enterprise_to_ebitda: Number(data?.enterprise_to_ebitda) || 0
  };
}

// Validation helper function for financial metrics
export function validateFinancials(data: any): Financials {
  return {
    fiscal_year_ends: data?.fiscal_year_ends || '',
    most_recent_quarter: data?.most_recent_quarter || '',
    gross_margin: Number(data?.gross_margin) || 0,
    profit_margin: Number(data?.profit_margin) || 0,
    operating_margin: Number(data?.operating_margin) || 0,
    return_on_assets_ttm: Number(data?.return_on_assets_ttm) || 0,
    return_on_equity_ttm: Number(data?.return_on_equity_ttm) || 0,
    income_statement: {
      revenue_ttm: Number(data?.income_statement?.revenue_ttm) || 0,
      revenue_per_share_ttm: Number(data?.income_statement?.revenue_per_share_ttm) || 0,
      quarterly_revenue_growth: Number(data?.income_statement?.quarterly_revenue_growth) || 0,
      gross_profit_ttm: Number(data?.income_statement?.gross_profit_ttm) || 0,
      ebitda: Number(data?.income_statement?.ebitda) || 0,
      net_income_to_common_ttm: Number(data?.income_statement?.net_income_to_common_ttm) || 0,
      diluted_eps_ttm: Number(data?.income_statement?.diluted_eps_ttm) || 0,
      quarterly_earnings_growth_yoy: Number(data?.income_statement?.quarterly_earnings_growth_yoy) || 0
    },
    balance_sheet: {
      total_cash_mrq: Number(data?.balance_sheet?.total_cash_mrq) || 0,
      total_cash_per_share_mrq: Number(data?.balance_sheet?.total_cash_per_share_mrq) || 0,
      total_debt_mrq: Number(data?.balance_sheet?.total_debt_mrq) || 0,
      total_debt_to_equity_mrq: Number(data?.balance_sheet?.total_debt_to_equity_mrq) || 0,
      current_ratio_mrq: Number(data?.balance_sheet?.current_ratio_mrq) || 0,
      book_value_per_share_mrq: Number(data?.balance_sheet?.book_value_per_share_mrq) || 0
    },
    cash_flow: {
      operating_cash_flow_ttm: Number(data?.cash_flow?.operating_cash_flow_ttm) || 0,
      levered_free_cash_flow_ttm: Number(data?.cash_flow?.levered_free_cash_flow_ttm) || 0
    }
  };
}

// Validation helper function for stock statistics
export function validateStockStatistics(data: any): StockStatistics {
  return {
    shares_outstanding: Number(data?.shares_outstanding) || 0,
    float_shares: Number(data?.float_shares) || 0,
    avg_10_volume: Number(data?.avg_10_volume) || 0,
    avg_90_volume: Number(data?.avg_90_volume) || 0
  };
}

// Main validation function for entire stock data
export function validateStockData(data: any): Partial<StockData> {
  try {
    return {
      quote: {
        symbol: data?.quote?.symbol || '',
        name: data?.quote?.name || '',
        close: String(data?.quote?.close || '0'),
        change: String(data?.quote?.change || '0'),
        percent_change: String(data?.quote?.percent_change || '0'),
        high: String(data?.quote?.high || '0'),
        low: String(data?.quote?.low || '0'),
        open: String(data?.quote?.open || '0'),
        volume: String(data?.quote?.volume || '0')
      },
      timeSeries: {
        values: Array.isArray(data?.timeSeries?.values)
          ? data.timeSeries.values.map((item: any) => ({
              datetime: String(item.datetime || ''),
              open: String(item.open || '0'),
              high: String(item.high || '0'),
```

```typescript
                  low: String(item.low || '0'),
                  close: String(item.close || '0'),
                  volume: String(item.volume || '0')
                }))
              : []
        },
        statistics: {
          valuations_metrics: validateValuationMetrics(data?.statistics?.valuations_metrics),
          financials: validateFinancials(data?.statistics?.financials),
          stock_statistics: validateStockStatistics(data?.statistics?.stock_statistics),
          stock_price_summary: {
            fifty_two_week_low: Number(data?.statistics?.stock_price_summary?.fifty_two_week_low) || 0,
            fifty_two_week_high: Number(data?.statistics?.stock_price_summary?.fifty_two_week_high) || 0,
            fifty_two_week_change: Number(data?.statistics?.stock_price_summary?.fifty_two_week_change) || 0,
            beta: Number(data?.statistics?.stock_price_summary?.beta) || 0,
            day_50_ma: Number(data?.statistics?.stock_price_summary?.day_50_ma) || 0,
            day_200_ma: Number(data?.statistics?.stock_price_summary?.day_200_ma) || 0
          },
          dividends_and_splits: {
            forward_annual_dividend_rate: Number(data?.statistics?.dividends_and_splits?.forward_annual_dividen
d_rate) || 0,
            forward_annual_dividend_yield: Number(data?.statistics?.dividends_and_splits?.forward_annual_divide
nd_yield) || 0,
            trailing_annual_dividend_rate: Number(data?.statistics?.dividends_and_splits?.trailing_annual_divid
end_rate) || 0,
            trailing_annual_dividend_yield: Number(data?.statistics?.dividends_and_splits?.trailing_annual_divi
dend_yield) || 0,
            "5_year_average_dividend_yield": Number(data?.statistics?.dividends_and_splits?.["5_year_average_di
vidend_yield"]) || 0,
            payout_ratio: Number(data?.statistics?.dividends_and_splits?.payout_ratio) || 0,
            dividend_date: String(data?.statistics?.dividends_and_splits?.dividend_date || ''),
            ex_dividend_date: String(data?.statistics?.dividends_and_splits?.ex_dividend_date || '')
          }
        }
      };
    } catch (error) {
      console.error('Error validating stock data:', error);
      return {};
    }
  }

  // Get formatted return metrics for display
  export function getReturnMetricsDisplay(financials: Financials): Array<MetricItem> {
    return [
      {
        label: 'ROE',
        value: financials.return_on_equity_ttm,
        format: 'percent',
        description: 'Return on equity'
      },
      {
        label: 'ROA',
        value: financials.return_on_assets_ttm,
        format: 'percent',
        description: 'Return on assets'
      }
    ];
  }
```

```ts
// config/navigation.ts

import { LayoutDashboard, BarChart2, FileText } from "lucide-react";

export const mainNav = [
  {
    title: "Chat",
    href: "/",
    icon: LayoutDashboard,
  },
  {
    title: "Market",
    href: "/market",
    icon: BarChart2,
  },
  {
    title: "Documents",
    href: "/documents",
    icon: FileText,
  },
];
```

```typescript
// /app/config/market.ts

export const API_KEY = process.env.NEXT_PUBLIC_TWELVE_DATA_API_KEY || 'b82e4abed0b347e7bd07c5d33cc753e3';
export const API_URL = 'https://api.twelvedata.com';
export const WS_URL = 'wss://ws.twelvedata.com/v1/quotes/price';

export interface CompanyLogo {
  meta: {
    symbol: string;
    name: string;
    currency: string;
    exchange: string;
    mic_code: string;
    exchange_timezone: string;
  };
  url: string;
}


// Helper function to format symbol for API calls
export function formatSymbol(tickerSymbol: string) {
    return `${tickerSymbol}:TADAWUL`;
  }




export const SAUDI_MARKET = {
  exchange: 'Tadawul',
  mic_code: 'XSAU',
  currency: 'SAR',
  timezone: 'Asia/Riyadh',
  trading_hours: 'Su-Th, 10:00am - 3:00pm',
  data_delay: '15min'
} as const;

// Updated with correct symbols and standardized format
export const SAUDI_SYMBOLS = [
  // Banks & Financial Services

    {
        symbol: '2222:TADAWUL',
        tickerSymbol: '2222',
        name: 'Saudi Aramco',
        name_ar: 'أرامكÙ\205Ù\203Ù\210 Ø§Ù\204Ø³Ø¹Ù\210Ø¯Ù\212Ø©',
        sector: 'Energy',
        website: 'www.saudiaramco.com'
    },
    {
        symbol: '2381:TADAWUL',
        tickerSymbol: '2381',
        name: 'Arabian Drilling',
        name_ar: 'Ø´Ø±Ù\203Ø© Ø§Ù\204ØÙ\201Ø± Ø§Ù\204Ø¹Ø±Ø¨Ù\212Ø©',
        sector: 'Energy',
        website: 'www.arabdrill.com'
    },
    {
        symbol: '2382:TADAWUL',
        tickerSymbol: '2382',
        name: 'ADES',
        name_ar: 'Ø£Ø¯Ù\212Ø³',
        sector: 'Energy',
        website: 'www.adesgroup.com'
    },
    {
        symbol: '2380:TADAWUL',
        tickerSymbol: '2380',
        name: 'Petro Rabigh',
        name_ar: 'Ø¨ØªØ±Ù\210Ø±Ø§Ø¨Ø°',
        sector: 'Energy',
        website: 'www.petrorabigh.com'
    },
    {
        symbol: '4030:TADAWUL',
        tickerSymbol: '4030',
        name: 'Bahri',
        name_ar: 'Ø§Ù\204Ø¨ØØ±Ù\212',
        sector: 'Transportation',
        website: 'www.bahri.sa'
    },
    {
        symbol: '4200:TADAWUL',
        tickerSymbol: '4200',
        name: 'Aldrees',
        name_ar: 'Ø§Ù\204Ø¯Ø±Ù\212Ø³',
        sector: 'Energy',
```

```typescript
      website: 'www.aldrees.com'
   },
   {
      symbol: '2030:TADAWUL',
      tickerSymbol: '2030',
      name: 'SARCO',
      name_ar: 'Ø³Ø§Ø±Ù\203Ù\210',
      sector: 'Materials',
      website: 'www.sarco.com.sa'
   },
   {
      symbol: '1202:TADAWUL',
      tickerSymbol: '1202',
      name: 'MEPCO',
      name_ar: 'Ù\205Ø¨Ù\203Ù\210',
      sector: 'Materials',
      website: 'www.mepco.biz'
   },
   {
      symbol: '3007:TADAWUL',
      tickerSymbol: '3007',
      name: 'Oasis',
      name_ar: 'Ù\210Ø§Ø-Ø©',
      sector: 'Materials',
      website: 'www.oasis.com.sa'
   },
   {
      symbol: '1201:TADAWUL',
      tickerSymbol: '1201',
      name: 'Takween',
      name_ar: 'ØªÙ\203Ù\210Ù\212Ù\206',
      sector: 'Materials',
      website: 'www.takween.com.sa'
   },
   {
      symbol: '1322:TADAWUL',
      tickerSymbol: '1322',
      name: 'AMAK',
      name_ar: 'Ø£Ù\205Ù\203',
      sector: 'Materials',
      website: 'www.amak.com.sa'
   },
   {
      symbol: '3008:TADAWUL',
      tickerSymbol: '3008',
      name: 'Alkathiri',
      name_ar: 'Ø§Ù\204Ù\203Ø«Ù\212Ø±Ù\212',
      sector: 'Materials',
      website: 'www.alkathiri.com'
   },
   {
      symbol: '2001:TADAWUL',
      tickerSymbol: '2001',
      name: 'Chemanol',
      name_ar: 'Ù\203Ù\212Ù\205Ø§Ù\206Ù\210Ù\204',
      sector: 'Materials',
      website: 'www.chemanol.com'
   },
   {
      symbol: '2010:TADAWUL',
      tickerSymbol: '2010',
      name: 'SABIC',
      name_ar: 'Ø³Ø§Ø¨Ù\203',
      sector: 'Materials',
      website: 'www.sabic.com'
   },
   {
      symbol: '2020:TADAWUL',
      tickerSymbol: '2020',
      name: 'SABIC Agri-Nutrients',
      name_ar: 'Ø³Ø§Ø¨Ù\203 Ù\204Ù\204Ù\205Ø°Ø°Ù\212Ø§Øª Ø§Ù\204Ø²Ø±Ø§Ø¹Ù\212Ø©',
      sector: 'Materials',
      website: 'www.sabic-agri.com'
   },
   {
      symbol: '2060:TADAWUL',
      tickerSymbol: '2060',
      name: 'Tasnee',
      name_ar: 'ØªØµÙ\206Ù\212Ø¹',
      sector: 'Materials',
      website: 'www.tasnee.com'
   },
   {
      symbol: '2170:TADAWUL',
      tickerSymbol: '2170',
      name: 'Alujain',
```

```
        name_ar: 'اﻷﻟﺠﻳﻥ',
        sector: 'Materials',
        website: 'www.alujain.com'
    },
    {
        symbol: '2210:TADAWUL',
        tickerSymbol: '2210',
        name: 'Nama Chemicals',
        name_ar: 'ﻧﻣﺎء ﻟﻠﻛﻳﻣﺎﻭﻳﺎﺕ',
        sector: 'Materials',
        website: 'www.nama.com.sa'
    },
    {
        symbol: '2250:TADAWUL',
        tickerSymbol: '2250',
        name: 'SIIG',
        name_ar: 'اﻟﻣﺟﻣﻭﻋﺔ اﻟﺳﻌﻭﺩﻳﺔ',
        sector: 'Materials',
        website: 'www.siig.com.sa'
    },
    {
        symbol: '2290:TADAWUL',
        tickerSymbol: '2290',
        name: 'Yansab',
        name_ar: 'ﻳﻧﺳﺎﺏ',
        sector: 'Materials',
        website: 'www.yansab.com.sa'
    },
    {
        symbol: '2310:TADAWUL',
        tickerSymbol: '2310',
        name: 'Sipchem',
        name_ar: 'ﺳﺎﺑﻛﻡ',
        sector: 'Materials',
        website: 'www.sipchem.com'
    },
    {
        symbol: '2330:TADAWUL',
        tickerSymbol: '2330',
        name: 'Advanced',
        name_ar: 'اﻟﻣﺗﻘﺩﻣﺔ',
        sector: 'Materials',
        website: 'www.advancedpetrochem.com'
    },
    {
        symbol: '2350:TADAWUL',
        tickerSymbol: '2350',
        name: 'Saudi Kayan',
        name_ar: 'ﻛﻳﺎﻥ اﻟﺳﻌﻭﺩﻳﺔ',
        sector: 'Materials',
        website: 'www.saudikayan.com'
    },
    {
        symbol: '2223:TADAWUL',
        tickerSymbol: '2223',
        name: 'Luberef',
        name_ar: 'ﻟﻭﺑﺭﻳﻑ',
        sector: 'Energy',
        website: 'www.luberef.com'
    },
    {
        symbol: '1301:TADAWUL',
        tickerSymbol: '1301',
        name: 'Aslak',
        name_ar: 'أﺳﻼﻙ',
        sector: 'Materials',
        website: 'www.aslak.com.sa'
    },
    {
        symbol: '1320:TADAWUL',
        tickerSymbol: '1320',
        name: 'SSP',
        name_ar: 'ﺳﺎﺑﻛﻡ',
        sector: 'Materials',
        website: 'www.ssp.com.sa'
    },
    {
        symbol: '2090:TADAWUL',
        tickerSymbol: '2090',
        name: 'NGC',
        name_ar: 'ﻏﺎﺯ',
        sector: 'Materials',
        website: 'www.ngc.com.sa'
    },
    {
        symbol: '2200:TADAWUL',
```

```typescript
        tickerSymbol: '2200',
        name: 'APC',
        name_ar: 'اÙ\204أÙ£Ù\205Ø¯Ø© اÙ\204Ø¹Ø±Ø¨Ù\212Ø©',
        sector: 'Materials',
        website: 'www.apc.com.sa'
    },
    {
        symbol: '2240:TADAWUL',
        tickerSymbol: '2240',
        name: 'Zamil Industrial',
        name_ar: 'اÙ\204Ø²اÙ\205Ù\204 Ù\204Ù\204ØµÙ\206اØ¹Ø©',
        sector: 'Materials',
        website: 'www.zamilindustrial.com'
    },
    {
        symbol: '2360:TADAWUL',
        tickerSymbol: '2360',
        name: 'SVCP',
        name_ar: 'اÙ\204Ù\201اØ¨اØ±Ù\212Ø©',
        sector: 'Materials',
        website: 'www.svcp.com.sa'
    },
    {
        symbol: '1304:TADAWUL',
        tickerSymbol: '1304',
        name: 'Alyamamah Steel',
        name_ar: 'اÙ\204Ù\212Ù\205اÙ\205Ø© Ù\204Ù\204ØØ¯Ù\212Ø¯',
        sector: 'Materials',
        website: 'www.alyamamahsteel.com'
    },
    {
        symbol: '1321:TADAWUL',
        tickerSymbol: '1321',
        name: 'East Pipes',
        name_ar: 'Ø£Ù\206Ø¨اØ¨ Ù\212Ø¨ اÙ\204Ø´Ø±Ù\202',
        sector: 'Materials',
        website: 'www.eastpipes.com'
    },
    {
        symbol: '1210:TADAWUL',
        tickerSymbol: '1210',
        name: 'BCI',
        name_ar: 'اÙ\204Ù\203Ù\212Ù\205Ù\212اØ¡Ù\212Ø© اÙ\204أØ³اØ³Ù\212Ø©',
        sector: 'Materials',
        website: 'www.bci.com.sa'
    },
    {
        symbol: '1211:TADAWUL',
        tickerSymbol: '1211',
        name: 'Maaden',
        name_ar: 'Ù\205Ø¹Ø§Ø¯Ù\206',
        sector: 'Materials',
        website: 'www.maaden.com.sa'
    },
    {
        symbol: '2150:TADAWUL',
        tickerSymbol: '2150',
        name: 'Zoujaj',
        name_ar: 'اÙ\204Ø²Ø¬Ø§Ø¬',
        sector: 'Materials',
        website: 'www.zoujaj.com'
    },
    {
        symbol: '2180:TADAWUL',
        tickerSymbol: '2180',
        name: 'FIPCO',
        name_ar: 'Ù\201Ù\212Ø¨Ù\203Ù\210',
        sector: 'Materials',
        website: 'www.fipco.com.sa'
    },
    {
        symbol: '2220:TADAWUL',
        tickerSymbol: '2220',
        name: 'Maadaniyah',
        name_ar: 'Ù\205Ø¹Ø§Ø¯Ù\206',
        sector: 'Materials',
        website: 'www.maadaniyah.com'
    },
    {
        symbol: '2300:TADAWUL',
        tickerSymbol: '2300',
        name: 'SPM',
        name_ar: 'اÙ\204Ø³Ø¨Ù\203',
        sector: 'Materials',
        website: 'www.spm.com.sa'
    },
```

```
    {
        symbol: '3002:TADAWUL',
        tickerSymbol: '3002',
        name: 'Najran Cement',
        name_ar: 'Ø£Ø³Ù\205Ù\206Øª Ù\206Ø¬Ø±Ø§Ù\206',
        sector: 'Materials',
        website: 'www.najrancement.com'
    },
    {
        symbol: '3003:TADAWUL',
        tickerSymbol: '3003',
        name: 'City Cement',
        name_ar: 'Ø£Ø³Ù\205Ù\206Øª Ø§Ù\204Ù\205Ø¯Ù\212Ù\206Ø©',
        sector: 'Materials',
        website: 'www.citycement.com.sa'
    },
    {
        symbol: '3004:TADAWUL',
        tickerSymbol: '3004',
        name: 'Northern Cement',
        name_ar: 'Ø£Ø³Ù\205Ù\206Øª Ø§Ù\204Ø´Ù\205Ø§Ù\204Ù\212Ø©',
        sector: 'Materials',
        website: 'www.northerncement.com'
    },
    {
        symbol: '3010:TADAWUL',
        tickerSymbol: '3010',
        name: 'ACC',
        name_ar: 'Ø£Ø³Ù\205Ù\206Øª Ø§Ù\204Ø¹Ø±Ø¨Ù\212Ø©',
        sector: 'Materials',
        website: 'www.arabiacement.com.sa'
    },
    {
        symbol: '3020:TADAWUL',
        tickerSymbol: '3020',
        name: 'YC',
        name_ar: 'Ø£Ø³Ù\205Ù\206Øª Ù\212Ù\206Ø¨Ø¹',
        sector: 'Materials',
        website: 'www.yanbu-cement.com'
    },
    {
        symbol: '3030:TADAWUL',
        tickerSymbol: '3030',
        name: 'Saudi Cement',
        name_ar: 'Ø£Ø³Ù\205Ù\206Øª Ø§Ù\204Ø³Ø¹Ù\210Ø¯Ù\212Ø©',
        sector: 'Materials',
        website: 'www.saudicement.com.sa'
    },
    {
        symbol: '3040:TADAWUL',
        tickerSymbol: '3040',
        name: 'QACCO',
        name_ar: 'Ø£Ø³Ù\205Ù\206Øª Ø§Ù\204Ù\202ØµÙ\212Ù\205',
        sector: 'Materials',
        website: 'www.qcc.com.sa'
    },
    {
        symbol: '3050:TADAWUL',
        tickerSymbol: '3050',
        name: 'SPCC',
        name_ar: 'Ø£Ø³Ù\205Ù\206Øª Ø§Ù\204Ø¬Ù\206Ù\210Ø¨',
        sector: 'Materials',
        website: 'www.spcc.com.sa'
    },
    {
        symbol: '3060:TADAWUL',
        tickerSymbol: '3060',
        name: 'YCC',
        name_ar: 'Ø£Ø³Ù\205Ù\206Øª Ù\212Ù\206Ø¨Ø¹',
        sector: 'Materials',
        website: 'www.yanbu-cement.com'
    },
    {
        symbol: '3080:TADAWUL',
        tickerSymbol: '3080',
        name: 'EPCCO',
        name_ar: 'Ø£Ø³Ù\205Ù\206Øª Ø§Ù\204Ø´Ø±Ù\202Ù\212Ø©',
        sector: 'Materials',
        website: 'www.epcco.com.sa'
    },
    {
        symbol: '3090:TADAWUL',
        tickerSymbol: '3090',
        name: 'TCC',
        name_ar: 'Ø£Ø³Ù\205Ù\206Øª ØªØ¨Ù\210Ù\203',
        sector: 'Materials',
```

```
            website: 'www.tcc.sa'
        },
        {
            symbol: '3091:TADAWUL',
            tickerSymbol: '3091',
            name: 'Jouf Cement',
            name_ar: 'Ø£Ø³Ù\205Ù\206Øª Ø§Ù\204Ø¬Ù\210Ù\201',
            sector: 'Materials',
            website: 'www.joufcement.com'
        },
        {
            symbol: '3005:TADAWUL',
            tickerSymbol: '3005',
            name: 'UACC',
            name_ar: 'Ø£Ø³Ù\205Ù\206Øª Ø§Ù\204Ù\205ØªØ¯Ø©',
            sector: 'Materials',
            website: 'www.uacc.com.sa'
        },
        {
            symbol: '3092:TADAWUL',
            tickerSymbol: '3092',
            name: 'Riyadh Cement',
            name_ar: 'Ø£Ø³Ù\205Ù\206Øª Ø§Ù\204Ø±Ù\212Ø§Ø¶',
            sector: 'Materials',
            website: 'www.riyadhcement.com.sa'
        },
        {
            symbol: '4142:TADAWUL',
            tickerSymbol: '4142',
            name: 'Riyadh Cables',
            name_ar: 'Ø§Ù\204Ù\203Ø§Ø¨Ù\204Ø§Øª Ø§Ù\204Ø³Ø¹Ù\210Ø¯Ù\212Ø©',
            sector: 'Capital Goods',
            website: 'www.riyadh-cables.com'
        },
        {
            symbol: '2160:TADAWUL',
            tickerSymbol: '2160',
            name: 'Amiantit',
            name_ar: 'Ø£Ù\205Ù\212Ø§Ù\206ØªÙ\212Øª',
            sector: 'Capital Goods',
            website: 'www.amiantit.com'
        },
        {
            symbol: '4140:TADAWUL',
            tickerSymbol: '4140',
            name: 'SIECO',
            name_ar: 'Ø³Ù\212Ø³Ù\203Ù\210',
            sector: 'Capital Goods',
            website: 'www.sieco.com.sa'
        },
        {
            symbol: '1212:TADAWUL',
            tickerSymbol: '1212',
            name: 'Astra Industrial',
            name_ar: 'Ù\205Ø¬Ù\205Ù\210Ø¹Ø© Ø£Ø³ØªØ±Ø§ Ø§Ù\204µÙ\206Ø§Ø¹Ù\212Ø©',
            sector: 'Capital Goods',
            website: 'www.astraindustrial.com'
        },
        {
            symbol: '1302:TADAWUL',
            tickerSymbol: '1302',
            name: 'Bawan',
            name_ar: 'Ø¨Ù\210Ø§Ù\206',
            sector: 'Capital Goods',
            website: 'www.bawan.com.sa'
        },
        {
            symbol: '2370:TADAWUL',
            tickerSymbol: '2370',
            name: 'MESC',
            name_ar: 'Ù\205Ø³Ù\203',
            sector: 'Capital Goods',
            website: 'www.mesc.com.sa'
        },
        {
            symbol: '1303:TADAWUL',
            tickerSymbol: '1303',
            name: 'EIC',
            name_ar: 'Ø¥Ù\212Ù\203',
            sector: 'Capital Goods',
            website: 'www.eic.com.sa'
        },
        {
            symbol: '1214:TADAWUL',
            tickerSymbol: '1214',
            name: 'Shaker',
```

```
        name_ar: 'شاكر',
        sector: 'Capital Goods',
        website: 'www.shaker.com.sa'
    },
    {
        symbol: '2320:TADAWUL',
        tickerSymbol: '2320',
        name: 'Albabtain',
        name_ar: 'البابطين',
        sector: 'Capital Goods',
        website: 'www.albabtain.com.sa'
    },
    {
        symbol: '2110:TADAWUL',
        tickerSymbol: '2110',
        name: 'Saudi Cable',
        name_ar: 'الكابلات السعودية',
        sector: 'Capital Goods',
        website: 'www.saudicable.com'
    },
    {
        symbol: '2040:TADAWUL',
        tickerSymbol: '2040',
        name: 'Saudi Ceramics',
        name_ar: 'الخزف السعودي',
        sector: 'Capital Goods',
        website: 'www.saudiceramics.com'
    },
    {
        symbol: '4110:TADAWUL',
        tickerSymbol: '4110',
        name: 'BATIC',
        name_ar: 'باتك',
        sector: 'Capital Goods',
        website: 'www.batic.com.sa'
    },
    {
        symbol: '4141:TADAWUL',
        tickerSymbol: '4141',
        name: 'Alomran',
        name_ar: 'العمران',
        sector: 'Capital Goods',
        website: 'www.alomran.com.sa'
    },
    {
        symbol: '4143:TADAWUL',
        tickerSymbol: '4143',
        name: 'TALCO',
        name_ar: 'تالكو',
        sector: 'Capital Goods',
        website: 'www.talco.com.sa'
    },
    {
        symbol: '4270:TADAWUL',
        tickerSymbol: '4270',
        name: 'SPPC',
        name_ar: 'المطبعة',
        sector: 'Commercial & Professional Services',
        website: 'www.sppc.com.sa'
    },
    {
        symbol: '1831:TADAWUL',
        tickerSymbol: '1831',
        name: 'Maharah',
        name_ar: 'مهارة',
        sector: 'Commercial & Professional Services',
        website: 'www.maharah.com.sa'
    },
    {
        symbol: '6004:TADAWUL',
        tickerSymbol: '6004',
        name: 'Catrion',
        name_ar: 'كاترين',
        sector: 'Commercial & Professional Services',
        website: 'www.catrion.com.sa'
    },
    {
        symbol: '1833:TADAWUL',
        tickerSymbol: '1833',
        name: 'Almawarid',
        name_ar: 'الموارد',
        sector: 'Commercial & Professional Services',
        website: 'www.almawarid.com.sa'
    },
    {
        symbol: '1832:TADAWUL',
```

```
                tickerSymbol: '1832',
                name: 'Sadr',
                name_ar: 'صدر',
                sector: 'Commercial & Professional Services',
                website: 'www.sadr.com.sa'
        },
        {
                symbol: '1834:TADAWUL',
                tickerSymbol: '1834',
                name: 'Smasco',
                name_ar: 'سماسكو',
                sector: 'Commercial & Professional Services',
                website: 'www.smasco.com.sa'
        },
        {
                symbol: '4040:TADAWUL',
                tickerSymbol: '4040',
                name: 'SAPTCO',
                name_ar: 'سابتكو',
                sector: 'Transportation',
                website: 'www.saptco.com.sa'
        },
        {
                symbol: '4260:TADAWUL',
                tickerSymbol: '4260',
                name: 'Budget Saudi',
                name_ar: 'بدجت السعودية',
                sector: 'Transportation',
                website: 'www.budgetsaudi.com'
        },
        {
                symbol: '4031:TADAWUL',
                tickerSymbol: '4031',
                name: 'SGS',
                name_ar: 'الخدمات الأرضية',
                sector: 'Transportation',
                website: 'www.sgs.com.sa'
        },
        {
                symbol: '4261:TADAWUL',
                tickerSymbol: '4261',
                name: 'Theeb',
                name_ar: 'ذيب',
                sector: 'Transportation',
                website: 'www.theeb.com.sa'
        },
        {
                symbol: '4262:TADAWUL',
                tickerSymbol: '4262',
                name: 'Lumi',
                name_ar: 'لومي',
                sector: 'Transportation',
                website: 'www.lumi.com.sa'
        },
        {
                symbol: '4263:TADAWUL',
                tickerSymbol: '4263',
                name: 'SAL',
                name_ar: 'سال',
                sector: 'Transportation',
                website: 'www.sal.com.sa'
        },
        {
                symbol: '2190:TADAWUL',
                tickerSymbol: '2190',
                name: 'SISCO Holding',
                name_ar: 'سيسكو',
                sector: 'Transportation',
                website: 'www.sisco.com.sa'
        },
        {
                symbol: '4180:TADAWUL',
                tickerSymbol: '4180',
                name: 'Fitaihi Group',
                name_ar: 'مجموعة فتيحي',
                sector: 'Consumer Durables & Apparel',
                website: 'www.fitaihi.com.sa'
        },
        {
                symbol: '4012:TADAWUL',
                tickerSymbol: '4012',
                name: 'Alaseel',
                name_ar: 'الأصيل',
                sector: 'Consumer Durables & Apparel',
                website: 'www.alaseel.com.sa'
        },
```

```ts
    {
        symbol: '4011:TADAWUL',
        tickerSymbol: '4011',
        name: 'Lazurde',
        name_ar: 'Ù\204Ø§Ø²Ù\210±Ø¯Ù\212',
        sector: 'Consumer Durables & Apparel',
        website: 'www.lazurde.com'
    },
    {
        symbol: '2130:TADAWUL',
        tickerSymbol: '2130',
        name: 'SIDC',
        name_ar: 'Ø³Ù\212Ø¯Ù\203',
        sector: 'Consumer Durables & Apparel',
        website: 'www.sidc.com.sa'
    },
    {
        symbol: '1213:TADAWUL',
        tickerSymbol: '1213',
        name: 'Naseej',
        name_ar: 'Ù\206Ø³Ù\212Ø¬',
        sector: 'Consumer Durables & Apparel',
        website: 'www.naseej.com.sa'
    },
    {
        symbol: '2340:TADAWUL',
        tickerSymbol: '2340',
        name: 'Artex',
        name_ar: 'Ø£Ø±ØªÙ\203Ø³',
        sector: 'Consumer Durables & Apparel',
        website: 'www.artex.com.sa'
    },
    {
        symbol: '1810:TADAWUL',
        tickerSymbol: '1810',
        name: 'Seera',
        name_ar: 'Ø³Ù\212Ø±Ø§',
        sector: 'Consumer Services',
        website: 'www.seera.sa'
    },
    {
        symbol: '4170:TADAWUL',
        tickerSymbol: '4170',
        name: 'TECO',
        name_ar: 'ØªÙ\212Ù\203Ù\210',
        sector: 'Consumer Services',
        website: 'www.teco.com.sa'
    },
    {
        symbol: '1820:TADAWUL',
        tickerSymbol: '1820',
        name: 'Alhokair Group',
        name_ar: 'Ù\205Ø¬Ù\205Ù\210Ø¹Ø© Ø§Ù\204ØÙ\203Ù\212Ø±',
        sector: 'Consumer Services',
        website: 'www.alhokair.com.sa'
    },
    {
        symbol: '6002:TADAWUL',
        tickerSymbol: '6002',
        name: 'Herfy Foods',
        name_ar: 'Ù\207Ø±Ù\201Ù\212 Ù\204Ù\204Ø£ºØ°Ù\212Ø©',
        sector: 'Consumer Services',
        website: 'www.herfy.com'
    },
    {
        symbol: '6016:TADAWUL',
        tickerSymbol: '6016',
        name: 'Burgerizzr',
        name_ar: 'Ø¨Ø±Ø°Ø±Ø§Ù\212Ø²Ø²Ø±',
        sector: 'Consumer Services',
        website: 'www.burgerizzr.com'
    },
    {
        symbol: '6013:TADAWUL',
        tickerSymbol: '6013',
        name: 'DWF',
        name_ar: 'Ø¯Ù\210Ù\201',
        sector: 'Consumer Services',
        website: 'www.dwf.com.sa'
    },
    {
        symbol: '6015:TADAWUL',
        tickerSymbol: '6015',
        name: 'Americana',
        name_ar: 'Ø£Ù\205Ø±Ù\212Ù\203Ø§Ù\206Ø§',
        sector: 'Consumer Services',
```

```
        website: 'www.americana.com.sa'
    },
    {
        symbol: '1830:TADAWUL',
        tickerSymbol: '1830',
        name: 'Leejam Sports',
        name_ar: 'Ù\204Ø¬Ø§Ù\205 Ù\204Ù\204Ø±Ù\212Ø§Ø¶Ø©',
        sector: 'Consumer Services',
        website: 'www.leejam.com.sa'
    },
    {
        symbol: '4291:TADAWUL',
        tickerSymbol: '4291',
        name: 'NCLE',
        name_ar: 'Ù\206Ù\203Ù\204Ù\212',
        sector: 'Consumer Services',
        website: 'www.ncle.com.sa'
    },
    {
        symbol: '4292:TADAWUL',
        tickerSymbol: '4292',
        name: 'ATAA',
        name_ar: 'Ø¹Ø•Ø§Ø¡',
        sector: 'Consumer Services',
        website: 'www.ataa.com.sa'
    },
    {
        symbol: '6012:TADAWUL',
        tickerSymbol: '6012',
        name: 'Raydan',
        name_ar: 'Ø±Ø§Ù\212Ø¯Ø§Ù\206',
        sector: 'Consumer Services',
        website: 'www.raydan.com.sa'
    },
    {
        symbol: '6014:TADAWUL',
        tickerSymbol: '6014',
        name: 'Alamar',
        name_ar: 'Ø§Ù\204Ù\205Ø±',
        sector: 'Consumer Services',
        website: 'www.alamar.com.sa'
    },
    {
        symbol: '4290:TADAWUL',
        tickerSymbol: '4290',
        name: 'Alkhaleej Trng',
        name_ar: 'Ø§Ù\204Ø®Ù\204Ù\212Ø¬ Ù\204Ù\204ªØ¯Ø±Ù\212Ø¨',
        sector: 'Consumer Services',
        website: 'www.alkhaleej.com.sa'
    },
    {
        symbol: '4070:TADAWUL',
        tickerSymbol: '4070',
        name: 'TAPRCO',
        name_ar: 'Ø•Ø§Ø¨Ù\203Ù\210',
        sector: 'Media and Entertainment',
        website: 'www.taprco.com.sa'
    },
    {
        symbol: '4210:TADAWUL',
        tickerSymbol: '4210',
        name: 'SRMG',
        name_ar: 'Ø§Ù\204Ù\205Ø¬Ù\205Ù\210Ø¹© Ø§Ù\204Ø³Ø¹Ù\210Ø¯Ù\212Ø©',
        sector: 'Media and Entertainment',
        website: 'www.srmg.com'
    },
    {
        symbol: '4071:TADAWUL',
        tickerSymbol: '4071',
        name: 'Alarabiya',
        name_ar: 'Ø§Ù\204Ø¹Ø±Ø¨Ù\212Ø©',
        sector: 'Media and Entertainment',
        website: 'www.alarabiya.com.sa'
    },
    {
        symbol: '4072:TADAWUL',
        tickerSymbol: '4072',
        name: 'MBC Group',
        name_ar: 'Ù\205Ø¬Ù\205Ù\210Ø¹© Ø¥Ù\205 Ø¨Ù\212 Ø³Ù\212',
        sector: 'Media and Entertainment',
        website: 'www.mbc.net'
    },
    {
        symbol: '4240:TADAWUL',
        tickerSymbol: '4240',
        name: 'Cenomi Retail',
```

```typescript
        name_ar: 'Ø³Ù\212Ù\206Ù\210Ù\205Ù\212 Ù\204Ù\204Ø¨Ù\212Ø¹ Ø¨Ø§Ù\204ØªØ¬Ø²Ø¦Ø©',
        sector: 'Consumer Discretionary Distribution & Retail',
        website: 'www.cenomi.com'
    },
    {
        symbol: '4190:TADAWUL',
        tickerSymbol: '4190',
        name: 'Jarir',
        name_ar: 'Ø¬Ø±Ù\212Ø±',
        sector: 'Consumer Discretionary Distribution & Retail',
        website: 'www.jarir.com'
    },
    {
        symbol: '4003:TADAWUL',
        tickerSymbol: '4003',
        name: 'Extra',
        name_ar: 'Ø¥Ù\203Ø³ØªØ±Ø§',
        sector: 'Consumer Discretionary Distribution & Retail',
        website: 'www.extra.com.sa'
    },
    {
        symbol: '4050:TADAWUL',
        tickerSymbol: '4050',
        name: 'SASCO',
        name_ar: 'Ø³Ø§Ø³Ù\203Ù\210',
        sector: 'Consumer Discretionary Distribution & Retail',
        website: 'www.sasco.com.sa'
    },
    {
        symbol: '4192:TADAWUL',
        tickerSymbol: '4192',
        name: 'Alsaif Gallery',
        name_ar: 'Ù\205Ø¹Ø±Ø¶ Ø§Ù\204Ø³Ù\212Ù\201',
        sector: 'Consumer Discretionary Distribution & Retail',
        website: 'www.alsaifgallery.com'
    },
    {
        symbol: '4051:TADAWUL',
        tickerSymbol: '4051',
        name: 'Baazeem',
        name_ar: 'Ø¨Ø§Ø²Ù\212Ù\205',
        sector: 'Consumer Discretionary Distribution & Retail',
        website: 'www.baazeem.com.sa'
    },
    {
        symbol: '4191:TADAWUL',
        tickerSymbol: '4191',
        name: 'Abo Moati',
        name_ar: 'Ø£Ø¨Ù\210 Ù\205Ø¹Ø•Ù\212',
        sector: 'Consumer Discretionary Distribution & Retail',
        website: 'www.abomoati.com.sa'
    },
    {
        symbol: '4008:TADAWUL',
        tickerSymbol: '4008',
        name: 'SACO',
        name_ar: 'Ø³Ø§Ù\203Ù\210',
        sector: 'Consumer Discretionary Distribution & Retail',
        website: 'www.saco.com.sa'
    },
    {
        symbol: '4001:TADAWUL',
        tickerSymbol: '4001',
        name: 'A.Othaim Market',
        name_ar: 'Ø£Ø³Ù\210Ø§Ù\202 Ø§Ù\204Ø¹Ø«Ù\212Ù\205',
        sector: 'Consumer Staples Distribution & Retail',
        website: 'www.othaimmarkets.com'
    },
    {
        symbol: '4161:TADAWUL',
        tickerSymbol: '4161',
        name: 'Bindawood',
        name_ar: 'Ø¨Ù\206Ø¯Ø©',
        sector: 'Consumer Staples Distribution & Retail',
        website: 'www.bindawood.com'
    },
    {
        symbol: '4162:TADAWUL',
        tickerSymbol: '4162',
        name: 'Almunajem',
        name_ar: 'Ø§Ù\204Ù\205Ù\206Ø¬Ù\205',
        sector: 'Consumer Staples Distribution & Retail',
        website: 'www.almunajem.com'
    },
    {
        symbol: '4164:TADAWUL',
```

```
        tickerSymbol: '4164',
        name: 'Nahdi',
        name_ar: 'Ø§Ù\204Ù\206Ù\207Ø¯Ù\212',
        sector: 'Consumer Staples Distribution & Retail',
        website: 'www.nahdi.sa'
    },
    {
        symbol: '4163:TADAWUL',
        tickerSymbol: '4163',
        name: 'Aldawaa',
        name_ar: 'Ø§Ù\204Ø¯Ù\210Ø§Ø¡',
        sector: 'Consumer Staples Distribution & Retail',
        website: 'www.aldawaa.com.sa'
    },
    {
        symbol: '4160:TADAWUL',
        tickerSymbol: '4160',
        name: 'Thimar',
        name_ar: 'Ø«Ù\205Ø§Ø±',
        sector: 'Consumer Staples Distribution & Retail',
        website: 'www.thimar.com.sa'
    },
    {
        symbol: '4006:TADAWUL',
        tickerSymbol: '4006',
        name: 'Farm Superstores',
        name_ar: 'Ø£Ø³Ù\210Ø§Ù\202 Ø§Ù\204Ù\205Ø²Ø±Ø¹Ø©',
        sector: 'Consumer Staples Distribution & Retail',
        website: 'www.farm.com.sa'
    },
    {
        symbol: '4061:TADAWUL',
        tickerSymbol: '4061',
        name: 'Anaam Holding',
        name_ar: 'Ø£Ù\206Ø¹Ø§Ù\205',
        sector: 'Consumer Staples Distribution & Retail',
        website: 'www.anaam.com.sa'
    },
    {
        symbol: '6001:TADAWUL',
        tickerSymbol: '6001',
        name: 'HB',
        name_ar: 'Ù\207Ø¨',
        sector: 'Food & Beverages',
        website: 'www.hb.com.sa'
    },
    {
        symbol: '2282:TADAWUL',
        tickerSymbol: '2282',
        name: 'Naqi',
        name_ar: 'Ù\206Ù\202Ù\212',
        sector: 'Food & Beverages',
        website: 'www.naqi.com.sa'
    },
    {
        symbol: '2283:TADAWUL',
        tickerSymbol: '2283',
        name: 'First Mills',
        name_ar: 'Ø§Ù\204Ù\205Ø·Ø§ØÙ\206 Ø§Ù\204Ø£Ù\210Ù\204Ù\211',
        sector: 'Food & Beverages',
        website: 'www.firstmills.com.sa'
    },
    {
        symbol: '2284:TADAWUL',
        tickerSymbol: '2284',
        name: 'Modern Mills',
        name_ar: 'Ø§Ù\204Ù\205Ø·Ø§ØÙ\206 Ø§Ù\204ØØ¯Ù\212Ø«Ø©',
        sector: 'Food & Beverages',
        website: 'www.modernmills.com.sa'
    },
    {
        symbol: '2285:TADAWUL',
        tickerSymbol: '2285',
        name: 'Arabian Mills',
        name_ar: 'Ø§Ù\204Ù\205Ø·Ø§ØÙ\206 Ø§Ù\204Ø¹Ø±Ø¨Ù\212Ø©',
        sector: 'Food & Beverages',
        website: 'www.arabianmills.com.sa'
    },
    {
        symbol: '2286:TADAWUL',
        tickerSymbol: '2286',
        name: 'Fourth Milling',
        name_ar: 'Ø§Ù\204Ù\205Ø·Ø§ØÙ\206 Ø§Ù\204Ø±Ø§Ø¨Ø¹Ø©',
        sector: 'Food & Beverages',
        website: 'www.fourthmilling.com.sa'
    },
```

```
    {
        symbol: '6010:TADAWUL',
        tickerSymbol: '6010',
        name: 'NADEC',
        name_ar: 'Ù\206Ø§Ø¯Ù\203',
        sector: 'Food & Beverages',
        website: 'www.nadec.com.sa'
    },
    {
        symbol: '6020:TADAWUL',
        tickerSymbol: '6020',
        name: 'GACO',
        name_ar: 'Ø¬Ù\203Ù\210',
        sector: 'Food & Beverages',
        website: 'www.gaco.com.sa'
    },
    {
        symbol: '6040:TADAWUL',
        tickerSymbol: '6040',
        name: 'TADCO',
        name_ar: 'Ø²Ø§Ø¯Ù\203Ù\210',
        sector: 'Food & Beverages',
        website: 'www.tadco.com.sa'
    },
    {
        symbol: '6050:TADAWUL',
        tickerSymbol: '6050',
        name: 'SFICO',
        name_ar: 'ØµÙ\201Ù\203Ù\210',
        sector: 'Food & Beverages',
        website: 'www.sfico.com.sa'
    },
    {
        symbol: '6060:TADAWUL',
        tickerSymbol: '6060',
        name: 'Sharqiyah Dev',
        name_ar: 'Ø§Ù\204Ø²Ù\206Ù\205Ù\212Ø© Ø§Ù\204Ø´Ø±Ù\202Ù\212Ø©',
        sector: 'Food & Beverages',
        website: 'www.sharqiyahdev.com.sa'
    },
    {
        symbol: '6070:TADAWUL',
        tickerSymbol: '6070',
        name: 'Aljouf',
        name_ar: 'Ø§Ù\204Ø¬Ù\210Ù\201',
        sector: 'Food & Beverages',
        website: 'www.aljouf.com.sa'
    },
    {
        symbol: '6090:TADAWUL',
        tickerSymbol: '6090',
        name: 'Jazadco',
        name_ar: 'Ø¬Ø§Ø²Ø§Ø¯Ù\203Ù\210',
        sector: 'Food & Beverages',
        website: 'www.jazadco.com'
    },
    {
        symbol: '2281:TADAWUL',
        tickerSymbol: '2281',
        name: 'Tanmiah',
        name_ar: 'Ø§Ù\206Ù\205Ù\212Ø©',
        sector: 'Food & Beverages',
        website: 'www.tanmiah.com'
    },
    {
        symbol: '2050:TADAWUL',
        tickerSymbol: '2050',
        name: 'Savola Group',
        name_ar: 'Ù\205Ø¬Ù\205Ù\210Ø¹Ø© ØµØ§Ù\201Ù\210Ù\204Ø§',
        sector: 'Food & Beverages',
        website: 'www.savola.com'
    },
    {
        symbol: '2100:TADAWUL',
        tickerSymbol: '2100',
        name: 'Wafrah',
        name_ar: 'Ù\210Ù\201Ø±Ø©',
        sector: 'Food & Beverages',
        website: 'www.wafrah.com.sa'
    },
    {
        symbol: '2270:TADAWUL',
        tickerSymbol: '2270',
        name: 'SADAFCO',
        name_ar: 'Ø³Ø¯Ø§Ù\201Ù\203Ù\210',
        sector: 'Food & Beverages',
```

```typescript
        website: 'www.sadafco.com'
    },
    {
        symbol: '2280:TADAWUL',
        tickerSymbol: '2280',
        name: 'Almarai',
        name_ar: 'اÙ\204Ù\205Ø±Ø§Ù\212',
        sector: 'Food & Beverages',
        website: 'www.almarai.com'
    },
    {
        symbol: '4080:TADAWUL',
        tickerSymbol: '4080',
        name: 'Sinad Holding',
        name_ar: 'Ø³Ù\206اد اÙ\204Ù\202ابضة',
        sector: 'Food & Beverages',
        website: 'www.sinad.com.sa'
    },
    {
        symbol: '4014:TADAWUL',
        tickerSymbol: '4014',
        name: 'Equipment House',
        name_ar: ' بÙ\212Øª اÙ\204Ù\205عداØª',
        sector: 'Health Care Equipment & Services',
        website: 'www.equipmenthouse.com.sa'
    },
    {
        symbol: '4017:TADAWUL',
        tickerSymbol: '4017',
        name: 'Fakeeh Care',
        name_ar: 'Ù\201Ù\202Ù\212Ù\207 Ù\203Ù\212Ø±',
        sector: 'Health Care Equipment & Services',
        website: 'www.fakeeh.care'
    },
    {
        symbol: '2140:TADAWUL',
        tickerSymbol: '2140',
        name: 'Ayyan',
        name_ar: 'Ø¹Ù\212اÙ\206',
        sector: 'Health Care Equipment & Services',
        website: 'www.ayyan.com.sa'
    },
    {
        symbol: '4013:TADAWUL',
        tickerSymbol: '4013',
        name: 'Sulaiman Alhabib',
        name_ar: 'Ø³Ù\204Ù\212Ù\205اÙ\206 اÙ\204ØبÙ\212Ø¨',
        sector: 'Health Care Equipment & Services',
        website: 'www.hmg.com.sa'
    },
    {
        symbol: '4005:TADAWUL',
        tickerSymbol: '4005',
        name: 'Care',
        name_ar: 'Ø±Ø¹اÙ\212Ø©',
        sector: 'Health Care Equipment & Services',
        website: 'www.care.com.sa'
    },
    {
        symbol: '4002:TADAWUL',
        tickerSymbol: '4002',
        name: 'Mouwasat',
        name_ar: 'اÙ\205Ù\210اساة',
        sector: 'Health Care Equipment & Services',
        website: 'www.mouwasat.com'
    },
    {
        symbol: '4004:TADAWUL',
        tickerSymbol: '4004',
        name: 'Dallah Health',
        name_ar: 'دÙ\204Ù\207 اÙ\204صØÙ\212Ø©',
        sector: 'Health Care Equipment & Services',
        website: 'www.dallah.com.sa'
    },
    {
        symbol: '4007:TADAWUL',
        tickerSymbol: '4007',
        name: 'Alhammadi',
        name_ar: 'اÙ\204ØÙ\205Ø¯Ù\212',
        sector: 'Health Care Equipment & Services',
        website: 'www.alhammadi.com.sa'
    },
    {
        symbol: '4009:TADAWUL',
        tickerSymbol: '4009',
        name: 'Saudi German Health',
```

```typescript
        name_ar: 'Ø§Ù\204Ø³Ø¹Ù\210Ø¯Ù\212 Ø§Ù\204Ø£Ù\204Ù\205Ø§Ù\206Ù\212 Ø§Ù\204ØµØÙ\212Ø©',
        sector: 'Health Care Equipment & Services',
        website: 'www.sgh.com.sa'
    },
    {
        symbol: '2230:TADAWUL',
        tickerSymbol: '2230',
        name: 'Chemical',
        name_ar: 'Ø§Ù\204Ù\203Ù\212Ù\205Ù\212ØØ¦Ù\212Ø©',
        sector: 'Health Care Equipment & Services',
        website: 'www.chemical.com.sa'
    },
    {
        symbol: '2070:TADAWUL',
        tickerSymbol: '2070',
        name: 'SPIMACO',
        name_ar: 'Ø³Ø¨Ù\212Ù\205Ø§Ù\203Ù\210',
        sector: 'Pharma, Biotech & Life Sciences',
        website: 'www.spimaco.com.sa'
    },
    {
        symbol: '4015:TADAWUL',
        tickerSymbol: '4015',
        name: 'Jamjoom Pharma',
        name_ar: 'Ø¬Ù\205Ø¬Ù\210Ù\205 Ù\201Ø§Ø±Ù\205Ø§',
        sector: 'Pharma, Biotech & Life Sciences',
        website: 'www.jamjoompharma.com'
    },
    {
        symbol: '4016:TADAWUL',
        tickerSymbol: '4016',
        name: 'Avalon Pharma',
        name_ar: 'Ø£Ù\201Ø§Ù\204Ù\210Ù\206 Ù\201Ø§Ø±Ù\205Ø§',
        sector: 'Pharma, Biotech & Life Sciences',
        website: 'www.avalonpharma.com'
    },
    {
        symbol: '1010:TADAWUL',
        tickerSymbol: '1010',
        name: 'Riyad Bank',
        name_ar: 'Ø¨Ù\206Ù\203 Ø§Ù\204Ø±Ù\212Ø§Ø¶',
        sector: 'Banks',
        website: 'www.riyadbank.com'
    },
    {
        symbol: '1020:TADAWUL',
        tickerSymbol: '1020',
        name: 'BJAZ',
        name_ar: 'Ø¨Ù\206Ù\203 Ø§Ù\204Ø¬Ø²Ù\212Ø±Ø©',
        sector: 'Banks',
        website: 'www.bjaz.com.sa'
    },
    {
        symbol: '1030:TADAWUL',
        tickerSymbol: '1030',
        name: 'SAIB',
        name_ar: 'Ø§Ù\204Ø¨Ù\206Ù\203 Ø§Ù\204Ø³Ø¹Ù\210Ø¯Ù\212 Ù\204Ù\204Ø§Ø³ØªØ«Ù\205Ø§Ø±',
        sector: 'Banks',
        website: 'www.saib.com.sa'
    },
    {
        symbol: '1050:TADAWUL',
        tickerSymbol: '1050',
        name: 'BSF',
        name_ar: 'Ø§Ù\204Ø¨Ù\206Ù\203 Ø§Ù\204Ø³Ø¹Ù\210Ø¯Ù\212 Ø§Ù\204Ù\201Ø±Ù\206Ø³Ù\212',
        sector: 'Banks',
        website: 'www.bsf.com.sa'
    },
    {
        symbol: '1060:TADAWUL',
        tickerSymbol: '1060',
        name: 'SAB',
        name_ar: 'Ø§Ù\204Ø¨Ù\206Ù\203 Ø§Ù\204Ø³Ø¹Ù\210Ø¯Ù\212 Ø§Ù\204Ø¨Ø±Ù\212Ø·Ø§Ù\206Ù\212',
        sector: 'Banks',
        website: 'www.sabb.com'
    },
    {
        symbol: '1080:TADAWUL',
        tickerSymbol: '1080',
        name: 'ANB',
        name_ar: 'Ø§Ù\204Ø¨Ù\206Ù\203 Ø§Ù\204Ø¹Ø±Ø¨Ù\212 Ø§Ù\204Ù\210Ø·Ù\206Ù\212',
        sector: 'Banks',
        website: 'www.anb.com.sa'
    },
    {
        symbol: '1120:TADAWUL',
```

```
        tickerSymbol: '1120',
        name: 'Alrajhi',
        name_ar: 'Ù\205ØµØ±Ù\201 Ø§Ù\204Ø±Ø§Ø¬ØÙ\212',
        sector: 'Banks',
        website: 'www.alrajhibank.com.sa'
    },
    {
        symbol: '1140:TADAWUL',
        tickerSymbol: '1140',
        name: 'Albilad',
        name_ar: 'Ø¨Ù\206Ù\203 Ø§Ù\204Ø¨Ù\204Ø§Ø¯',
        sector: 'Banks',
        website: 'www.bankalbilad.com'
    },
    {
        symbol: '1150:TADAWUL',
        tickerSymbol: '1150',
        name: 'Alinma',
        name_ar: 'Ø¨Ù\206Ù\203 Ø§Ù\204Ø¥Ù\206Ù\205Ø§¡',
        sector: 'Banks',
        website: 'www.alinma.com'
    },
    {
        symbol: '1180:TADAWUL',
        tickerSymbol: '1180',
        name: 'SNB',
        name_ar: 'Ø§Ù\204Ø¨Ù\206Ù\203 Ø§Ù\204Ø£Ù\207Ù\204Ù\212 Ø§Ù\204Ø³Ø¹Ù\210Ø¯Ù\212',
        sector: 'Banks',
        website: 'www.alahli.com'
    },
    {
        symbol: '1183:TADAWUL',
        tickerSymbol: '1183',
        name: 'SHL',
        name_ar: 'Ø´Ù\204Ù\207',
        sector: 'Financial Services',
        website: 'www.shl.com.sa'
    },
    {
        symbol: '1182:TADAWUL',
        tickerSymbol: '1182',
        name: 'Amlak',
        name_ar: 'Ø£Ù\205Ù\204Ø§Ù\203',
        sector: 'Financial Services',
        website: 'www.amlak.com.sa'
    },
    {
        symbol: '4081:TADAWUL',
        tickerSymbol: '4081',
        name: 'Nayifat',
        name_ar: 'Ù\206Ø§Ù\212Ù\201Ø§Øª',
        sector: 'Financial Services',
        website: 'www.nayifat.com.sa'
    },
    {
        symbol: '1111:TADAWUL',
        tickerSymbol: '1111',
        name: 'Tadawul Group',
        name_ar: 'Ù\205Ø¬Ù\205Ù\210Ø¹© ØªØ¯Ø§Ù\210Ù\204',
        sector: 'Financial Services',
        website: 'www.tadawul.com.sa'
    },
    {
        symbol: '4082:TADAWUL',
        tickerSymbol: '4082',
        name: 'MRNA',
        name_ar: 'Ù\205Ø±Ù\206Ø©',
        sector: 'Financial Services',
        website: 'www.mrna.com.sa'
    },
    {
        symbol: '4130:TADAWUL',
        tickerSymbol: '4130',
        name: 'Albaha',
        name_ar: 'Ø§Ù\204Ø¨Ø§Ø©',
        sector: 'Financial Services',
        website: 'www.albaha.com.sa'
    },
    {
        symbol: '2120:TADAWUL',
        tickerSymbol: '2120',
        name: 'SAIC',
        name_ar: 'Ø³Ù\212Ø³Ø§Ù\203Ù\210',
        sector: 'Financial Services',
        website: 'www.saic.com.sa'
    },
```

```ts
  {
      symbol: '4280:TADAWUL',
      tickerSymbol: '4280',
      name: 'Kingdom',
      name_ar: 'اÙ\204Ù\205Ù\205Ù\204Ù\203Ø©',
      sector: 'Financial Services',
      website: 'www.kingdom.com.sa'
  },
  {
      symbol: '8313:TADAWUL',
      tickerSymbol: '8313',
      name: 'Rasan',
      name_ar: 'رØ³Ø§Ù\206',
      sector: 'Insurance',
      website: 'www.rasan.com.sa'
  },
  {
      symbol: '8010:TADAWUL',
      tickerSymbol: '8010',
      name: 'Tawuniya',
      name_ar: 'اÙ\204تØ¹اÙ\210Ù\206Ù\212Ø©',
      sector: 'Insurance',
      website: 'www.tawuniya.com.sa'
  },
  {
      symbol: '8020:TADAWUL',
      tickerSymbol: '8020',
      name: 'Malath Insurance',
      name_ar: 'Ù\205Ù\204اذ Ù\204Ù\204تأÙ\205Ù\212Ù\206',
      sector: 'Insurance',
      website: 'www.malath.com.sa'
  },
  {
      symbol: '8030:TADAWUL',
      tickerSymbol: '8030',
      name: 'Medgulf',
      name_ar: 'Ù\205Ù\212Ø¯Ø°Ù\204Ù\201',
      sector: 'Insurance',
      website: 'www.medgulf.com.sa'
  },
  {
      symbol: '8060:TADAWUL',
      tickerSymbol: '8060',
      name: 'Walaa',
      name_ar: 'Ù\210Ù\204اØ¡',
      sector: 'Insurance',
      website: 'www.walaa.com.sa'
  },
  {
      symbol: '8040:TADAWUL',
      tickerSymbol: '8040',
      name: 'Allianz SF',
      name_ar: 'أÙ\204Ù\212اÙ\206Ø² Ø¥Ø³ Ø¥Ù\201',
      sector: 'Insurance',
      website: 'www.allianz.com.sa'
  },
  {
      symbol: '8070:TADAWUL',
      tickerSymbol: '8070',
      name: 'Arabian Shield',
      name_ar: 'اÙ\204Ø¯Ø±Ø¹ اÙ\204Ø¹Ø±Ø¨Ù\212',
      sector: 'Insurance',
      website: 'www.arabianshield.com.sa'
  },
  {
      symbol: '8050:TADAWUL',
      tickerSymbol: '8050',
      name: 'Salama',
      name_ar: 'Ø³Ù\204اÙ\205Ø©',
      sector: 'Insurance',
      website: 'www.salama.com.sa'
  },
  {
      symbol: '8100:TADAWUL',
      tickerSymbol: '8100',
      name: 'SAICO',
      name_ar: 'Ø³Ø§Ù\212Ù\203Ù\210',
      sector: 'Insurance',
      website: 'www.saico.com.sa'
  },
  {
      symbol: '8012:TADAWUL',
      tickerSymbol: '8012',
      name: 'Jazira Takaful',
      name_ar: 'اÙ\204Ø¬Ø²Ù\212Ø±Ø© ØªÙ\203اÙ\201Ù\204',
      sector: 'Insurance',
```

```
        website: 'www.jaziratakaful.com.sa'
    },
    {
        symbol: '8120:TADAWUL',
        tickerSymbol: '8120',
        name: 'Gulf Union Alahlia',
        name_ar: 'اÙ\204Ø®ÙÙ\204Ù\212Ø¬ اÙ\204Ø£Ù\207ÙÙ\204Ù\212Ø©',
        sector: 'Insurance',
        website: 'www.gulfunion.com.sa'
    },
    {
        symbol: '8150:TADAWUL',
        tickerSymbol: '8150',
        name: 'ACIG',
        name_ar: 'اÙ\204Ù\205Ø¬Ù\205Ù\210Ø¹Ø© اÙ\204Ù\205ØªØØ¯Ø©',
        sector: 'Insurance',
        website: 'www.acig.com.sa'
    },
    {
        symbol: '8160:TADAWUL',
        tickerSymbol: '8160',
        name: 'AICC',
        name_ar: 'Ø£Ù\203Ø³Ø§',
        sector: 'Insurance',
        website: 'www.aicc.com.sa'
    },
    {
        symbol: '8170:TADAWUL',
        tickerSymbol: '8170',
        name: 'Aletihad',
        name_ar: 'اÙ\204اتحاد',
        sector: 'Insurance',
        website: 'www.aletihad.com.sa'
    },
    {
        symbol: '8180:TADAWUL',
        tickerSymbol: '8180',
        name: 'Alsagr Insurance',
        name_ar: 'اÙ\204ØµÙ\202Ø± Ù\204Ù\204ØªØ£Ù\205Ù\212Ù\206',
        sector: 'Insurance',
        website: 'www.alsagr.com.sa'
    },
    {
        symbol: '8190:TADAWUL',
        tickerSymbol: '8190',
        name: 'UCA',
        name_ar: 'اÙ\204Ù\205ØªØØ¯Ø© Ù\204Ù\204ØªØ£Ù\205Ù\212Ù\206',
        sector: 'Insurance',
        website: 'www.uca.com.sa'
    },
    {
        symbol: '8200:TADAWUL',
        tickerSymbol: '8200',
        name: 'Saudi Re',
        name_ar: 'اØ¥Ø¹Ø§Ø¯Ø© اÙ\204Ø³Ø¹Ù\210Ø¯Ù\212Ø©',
        sector: 'Insurance',
        website: 'www.saudire.com'
    },
    {
        symbol: '8210:TADAWUL',
        tickerSymbol: '8210',
        name: 'Bupa Arabia',
        name_ar: 'Ø¨Ù\210Ø¨Ø§ اÙ\204Ø¹Ø±Ø¨Ù\212Ø©',
        sector: 'Insurance',
        website: 'www.bupa.com.sa'
    },
    {
        symbol: '8230:TADAWUL',
        tickerSymbol: '8230',
        name: 'Alrajhi Takaful',
        name_ar: 'ØªÙ\203اÙ\201Ù\204 اÙ\204راجحÙ\212',
        sector: 'Insurance',
        website: 'www.alrajhitakaful.com'
    },
    {
        symbol: '8240:TADAWUL',
        tickerSymbol: '8240',
        name: 'Chubb',
        name_ar: 'ØªØ´Ø¨',
        sector: 'Insurance',
        website: 'www.chubb.com.sa'
    },
    {
        symbol: '8250:TADAWUL',
        tickerSymbol: '8250',
        name: 'GIG',
```

```
            name_ar: 'جي آي جي',
            sector: 'Insurance',
            website: 'www.gig.com.sa'
        },
        {
            symbol: '8260:TADAWUL',
            tickerSymbol: '8260',
            name: 'Gulf General',
            name_ar: 'الخليجية العامة',
            sector: 'Insurance',
            website: 'www.gulfgeneral.com.sa'
        },
        {
            symbol: '8270:TADAWUL',
            tickerSymbol: '8270',
            name: 'Buruj',
            name_ar: 'بروج',
            sector: 'Insurance',
            website: 'www.buruj.com.sa'
        },
        {
            symbol: '8280:TADAWUL',
            tickerSymbol: '8280',
            name: 'Liva',
            name_ar: 'ليفا',
            sector: 'Insurance',
            website: 'www.liva.com.sa'
        },
        {
            symbol: '8300:TADAWUL',
            tickerSymbol: '8300',
            name: 'Wataniya',
            name_ar: 'الوطنية',
            sector: 'Insurance',
            website: 'www.wataniya.com.sa'
        },
        {
            symbol: '8310:TADAWUL',
            tickerSymbol: '8310',
            name: 'Amana Insurance',
            name_ar: 'أمانة للتأمين',
            sector: 'Insurance',
            website: 'www.amana.com.sa'
        },
        {
            symbol: '8311:TADAWUL',
            tickerSymbol: '8311',
            name: 'Enaya',
            name_ar: 'عناية',
            sector: 'Insurance',
            website: 'www.enaya.com.sa'
        },
        {
            symbol: '7010:TADAWUL',
            tickerSymbol: '7010',
            name: 'STC',
            name_ar: 'إس تي سي',
            sector: 'Telecommunication Services',
            website: 'www.stc.com.sa'
        },
        {
            symbol: '7020:TADAWUL',
            tickerSymbol: '7020',
            name: 'Etihad Etisalat',
            name_ar: 'اتحاد اتصالات',
            sector: 'Telecommunication Services',
            website: 'www.mobily.com.sa'
        },
        {
            symbol: '7030:TADAWUL',
            tickerSymbol: '7030',
            name: 'Zain KSA',
            name_ar: 'زين السعودية',
            sector: 'Telecommunication Services',
            website: 'www.sa.zain.com'
        },
        {
            symbol: '7040:TADAWUL',
            tickerSymbol: '7040',
            name: 'Atheeb Telecom',
            name_ar: 'اتصالات عذيب',
            sector: 'Telecommunication Services',
            website: 'www.atheeb.com.sa'
        },
        {
            symbol: '2080:TADAWUL',
```

```
        tickerSymbol: '2080',
        name: 'GASCO',
        name_ar: 'Ø°Ø§Ø²Ù\203Ù\210',
        sector: 'Utilities',
        website: 'www.gasco.com.sa'
    },
    {
        symbol: '5110:TADAWUL',
        tickerSymbol: '5110',
        name: 'Saudi Electricity',
        name_ar: 'Ø§Ù\204Ø³Ø¹Ù\210Ø¯Ù\212Ø© Ù\204Ù\204Ù\203Ù\207Ø±Ø¨Ø§Ø¡',
        sector: 'Utilities',
        website: 'www.se.com.sa'
    },
    {
        symbol: '2084:TADAWUL',
        tickerSymbol: '2084',
        name: 'Miahona',
        name_ar: 'Ù\205Ù\212Ø§Ù\207Ù\206Ø§',
        sector: 'Utilities',
        website: 'www.miahona.com.sa'
    },
    {
        symbol: '2081:TADAWUL',
        tickerSymbol: '2081',
        name: 'AWPT',
        name_ar: 'Ø£Ù\212 Ø¯Ø¨Ù\204Ù\212Ù\210 Ø¨Ù\212 ØªÙ\212',
        sector: 'Utilities',
        website: 'www.awpt.com.sa'
    },
    {
        symbol: '2082:TADAWUL',
        tickerSymbol: '2082',
        name: 'ACWA Power',
        name_ar: 'Ø£Ù\203Ù\210Ø§ Ø¨Ø§Ù\210Ø±',
        sector: 'Utilities',
        website: 'www.acwapower.com'
    },
    {
        symbol: '2083:TADAWUL',
        tickerSymbol: '2083',
        name: 'Marafiq',
        name_ar: 'Ù\205Ø±Ø§Ù\201Ù\202',
        sector: 'Utilities',
        website: 'www.marafiq.com.sa'
    },
    {
        symbol: '4330:TADAWUL',
        tickerSymbol: '4330',
        name: 'Riyad REIT',
        name_ar: 'Ø§Ù\204Ø±Ù\212Ø§Ø¶ Ø±Ù\212Øª',
        sector: 'REITs',
        website: 'www.riyadreit.com'
    },
    {
        symbol: '4331:TADAWUL',
        tickerSymbol: '4331',
        name: 'Aljazira REIT',
        name_ar: 'Ø§Ù\204Ø¬Ø²Ù\212Ø±Ø© Ø±Ù\212Øª',
        sector: 'REITs',
        website: 'www.aljazirareit.com'
    },
    {
        symbol: '4332:TADAWUL',
        tickerSymbol: '4332',
        name: 'Jadwa REIT Alharamain',
        name_ar: 'Ø¬Ø¯Ù\210Ù\211 Ø±Ù\212Øª Ø§Ù\204ØØ±Ù\205Ù\212Ù\206',
        sector: 'REITs',
        website: 'www.jadwareit.com'
    },
    {
        symbol: '4333:TADAWUL',
        tickerSymbol: '4333',
        name: 'Taleem REIT',
        name_ar: 'ØªØ¹Ù\204Ù\212Ù\205 Ø±Ù\212Øª',
        sector: 'REITs',
        website: 'www.taleemreit.com'
    },
    {
        symbol: '4334:TADAWUL',
        tickerSymbol: '4334',
        name: 'Al Maather REIT',
        name_ar: 'Ø§Ù\204Ù\205Ø¹Ø°Ø± Ø±Ù\212Øª',
        sector: 'REITs',
        website: 'www.almaatherreit.com'
    },
```

```
    {
        symbol: '4335:TADAWUL',
        tickerSymbol: '4335',
        name: 'Musharaka REIT',
        name_ar: 'Ù\205Ø´Ø§±Ù\203Ø© Ø±Ù\212Øª',
        sector: 'REITs',
        website: 'www.musharakareit.com'
    },
    {
        symbol: '4336:TADAWUL',
        tickerSymbol: '4336',
        name: 'Mulkia REIT',
        name_ar: 'Ù\205Ù\204Ù\203Ù\212Ø© Ø±Ù\212Øª',
        sector: 'REITs',
        website: 'www.mulkiareit.com'
    },
    {
        symbol: '4337:TADAWUL',
        tickerSymbol: '4337',
        name: 'SICO Saudi REIT',
        name_ar: 'Ø³Ù\212Ù\203Ù\210 Ø§Ù\204Ø³Ù¹Ù\210Ø¯Ù\212Ø© Ø±Ù\212Øª',
        sector: 'REITs',
        website: 'www.sicoreit.com'
    },
    {
        symbol: '4338:TADAWUL',
        tickerSymbol: '4338',
        name: 'Alahli REIT 1',
        name_ar: 'Ø§Ù\204Ø£Ù\207Ù\204Ù\212 Ø±Ù\212Øª 1',
        sector: 'REITs',
        website: 'www.alahlireit.com'
    },
    {
        symbol: '4344:TADAWUL',
        tickerSymbol: '4344',
        name: 'Sedco Capital REIT',
        name_ar: 'Ø³Ø¯Ù\203Ù\210 Ù\203Ø§Ø¨Ù\212ØªØ§Ù\204 Ø±Ù\212Øª',
        sector: 'REITs',
        website: 'www.sedcoreit.com'
    },
    {
        symbol: '4339:TADAWUL',
        tickerSymbol: '4339',
        name: 'Derayah REIT',
        name_ar: 'Ø¯Ø±Ø§Ù\212Ø© Ø±Ù\212Øª',
        sector: 'REITs',
        website: 'www.derayahreit.com'
    },
    {
        symbol: '4340:TADAWUL',
        tickerSymbol: '4340',
        name: 'Al Rajhi REIT',
        name_ar: 'Ø§Ù\204Ø±Ø§Ø¬ØÙ\212 Ø±Ù\212Øª',
        sector: 'REITs',
        website: 'www.alrajhireit.com'
    },
    {
        symbol: '4345:TADAWUL',
        tickerSymbol: '4345',
        name: 'Alinma Retail REIT',
        name_ar: 'Ø§Ù\204Ø¥Ù\206Ù\205ØØ¡ Ø±Ù\212Øª Ù\204Ù\204ØªØ¬Ø²Ø¦Ø©',
        sector: 'REITs',
        website: 'www.alinmareit.com'
    },
    {
        symbol: '4342:TADAWUL',
        tickerSymbol: '4342',
        name: 'Jadwa REIT Saudi',
        name_ar: 'Ø¬Ø¯Ù\210Ù\211 Ø±Ù\212Øª Ø§Ù\204Ø³Ø¹Ù\210Ø¯Ù\212Ø©',
        sector: 'REITs',
        website: 'www.jadwareit.com'
    },
    {
        symbol: '4346:TADAWUL',
        tickerSymbol: '4346',
        name: 'MEFIC REIT',
        name_ar: 'Ù\205Ù\212Ù\201Ù\203 Ø±Ù\212Øª',
        sector: 'REITs',
        website: 'www.meficreit.com'
    },
    {
        symbol: '4347:TADAWUL',
        tickerSymbol: '4347',
        name: 'Bonyan REIT',
        name_ar: 'Ø¨Ù\206Ù\212Ø§Ù\206 Ø±Ù\212Øª',
        sector: 'REITs',
```

```
            website: 'www.bonyanreit.com'
        },
        {
            symbol: '4348:TADAWUL',
            tickerSymbol: '4348',
            name: 'Alkhabeer REIT',
            name_ar: 'Ø§Ù\204Ø®Ø¨Ù\212Ø± Ø±Ù\212Øª',
            sector: 'REITs',
            website: 'www.alkhabeerreit.com'
        },
        {
            symbol: '4349:TADAWUL',
            tickerSymbol: '4349',
            name: 'Alinma Hospitality REIT',
            name_ar: 'Ø§Ù\204Ø¥Ù\206Ù\205Ø¡ Ø±Ù\212Øª Ù\204Ù\204Ø¶Ù\212Ø§Ù\201Ø©',
            sector: 'REITs',
            website: 'www.alinmareit.com'
        },
        {
            symbol: '4350:TADAWUL',
            tickerSymbol: '4350',
            name: 'Alistithmar REIT',
            name_ar: 'Ø§Ù\204Ø§Ø³ØªØ«Ù\205Ø± Ø±Ù\212Øª',
            sector: 'REITs',
            website: 'www.alistithmarreit.com'
        },
        {
            symbol: '4020:TADAWUL',
            tickerSymbol: '4020',
            name: 'Alakaria',
            name_ar: 'Ø§Ù\204Ø¹Ù\202Ø§Ø±Ù\212Ø©',
            sector: 'Real Estate Mgmt & Dev\'t',
            website: 'www.alakaria.com.sa'
        },
        {
            symbol: '4090:TADAWUL',
            tickerSymbol: '4090',
            name: 'Taiba',
            name_ar: 'Ø•Ù\212Ø¨Ø©',
            sector: 'Real Estate Mgmt & Dev\'t',
            website: 'www.taiba.com.sa'
        },
        {
            symbol: '4100:TADAWUL',
            tickerSymbol: '4100',
            name: 'MCDC',
            name_ar: 'Ù\205Ù\203Ø©',
            sector: 'Real Estate Mgmt & Dev\'t',
            website: 'www.mcdc.com.sa'
        },
        {
            symbol: '4150:TADAWUL',
            tickerSymbol: '4150',
            name: 'ARDCO',
            name_ar: 'Ø§Ù\204Ø£Ø±Ø¶',
            sector: 'Real Estate Mgmt & Dev\'t',
            website: 'www.ardco.com.sa'
        },
        {
            symbol: '4220:TADAWUL',
            tickerSymbol: '4220',
            name: 'Emaar EC',
            name_ar: 'Ø¥Ø¹Ù\205Ø§Ø± Ø§Ù\204Ù\205Ø¯Ù\212Ù\206Ø© Ø§Ù\204Ø§Ù\202ØªØµØ§Ø¯Ù\212Ø©',
            sector: 'Real Estate Mgmt & Dev\'t',
            website: 'www.emaar.com.sa'
        },
        {
            symbol: '4250:TADAWUL',
            tickerSymbol: '4250',
            name: 'Jabal Omar',
            name_ar: 'Ø¬Ø¨Ù\204 Ø¹Ù\205Ø±',
            sector: 'Real Estate Mgmt & Dev\'t',
            website: 'www.jabalomar.com.sa'
        },
        {
            symbol: '4300:TADAWUL',
            tickerSymbol: '4300',
            name: 'Dar Alarkan',
            name_ar: 'Ø¯Ø§Ø± Ø§Ù\204Ø£Ø±Ù\203Ø§Ù\206',
            sector: 'Real Estate Mgmt & Dev\'t',
            website: 'www.daralarkan.com'
        },
        {
            symbol: '4310:TADAWUL',
            tickerSymbol: '4310',
            name: 'KEC',
```

```
        name_ar: 'Ù\203Ù\212Ù\203',
        sector: 'Real Estate Mgmt & Dev\'t',
        website: 'www.kec.com.sa'
    },
    {
        symbol: '4320:TADAWUL',
        tickerSymbol: '4320',
        name: 'Alandalus',
        name_ar: 'Ø§Ù\204Ø£Ù\206دÙ\204Ø³',
        sector: 'Real Estate Mgmt & Dev\'t',
        website: 'www.alandalus.com.sa'
    },
    {
        symbol: '4321:TADAWUL',
        tickerSymbol: '4321',
        name: 'Cenomi Centers',
        name_ar: 'Ø³Ù\212Ù\206Ù\210Ù\205Ù\212 Ø³Ù\206ØªØ±Ø²',
        sector: 'Real Estate Mgmt & Dev\'t',
        website: 'www.cenomicenters.com'
    },
    {
        symbol: '4322:TADAWUL',
        tickerSymbol: '4322',
        name: 'Retal',
        name_ar: 'Ø±ØªØ§Ù\204',
        sector: 'Real Estate Mgmt & Dev\'t',
        website: 'www.retal.com.sa'
    },
    {
        symbol: '4323:TADAWUL',
        tickerSymbol: '4323',
        name: 'Sumou',
        name_ar: 'Ø³Ù\205Ù\210',
        sector: 'Real Estate Mgmt & Dev\'t',
        website: 'www.sumou.com.sa'
    },
    {
        symbol: '4230:TADAWUL',
        tickerSymbol: '4230',
        name: 'Red Sea',
        name_ar: 'Ø§Ù\204بØØ± Ø§Ù\204Ø£ØÙ\205Ø±',
        sector: 'Real Estate Mgmt & Dev\'t',
        website: 'www.redsea.com.sa'
    },
    {
        symbol: '7200:TADAWUL',
        tickerSymbol: '7200',
        name: 'MIS',
        name_ar: 'Ø¥Ù\205 Ø¢Ù\212 Ø¥Ø³',
        sector: 'Software & Services',
        website: 'www.mis.com.sa'
    },
    {
        symbol: '7201:TADAWUL',
        tickerSymbol: '7201',
        name: 'Arab Sea',
        name_ar: 'Ø§Ù\204بØØ± Ø§Ù\204Ø¹Ø±بÙ\212',
        sector: 'Software & Services',
        website: 'www.arabsea.com.sa'
    },
    {
        symbol: '7202:TADAWUL',
        tickerSymbol: '7202',
        name: 'Solutions',
        name_ar: 'ØÙ\204Ù\210Ù\204',
        sector: 'Software & Services',
        website: 'www.solutions.com.sa'
    },
    {
        symbol: '7203:TADAWUL',
        tickerSymbol: '7203',
        name: 'ELM',
        name_ar: 'Ø¹Ù\204Ù\205',
        sector: 'Software & Services',
        website: 'www.elm.com.sa'
    },
    {
        symbol: '7204:TADAWUL',
        tickerSymbol: '7204',
        name: '2P',
        name_ar: 'ØªÙ\210 Ø¨Ù\212',
        sector: 'Software & Services',
        website: 'www.2p.com.sa'
    },
    {
        symbol: '4165:TADAWUL',
```

```
        tickerSymbol: '4165',
        name: 'Almajed Oud',
        name_ar: 'Ø§Ù\204Ù\205Ø¬Ø¯ Ù\204Ù\204Ø¹Ù\210Ø¯',
        sector: 'Household & Personal Products',
        website: 'www.almajedoud.com.sa'
    },
    {
        symbol: '9406:TADAWUL',
        tickerSymbol: '9406',
        name: 'Albilad MSCI',
        name_ar: 'Ø§Ù\204Ø¨Ù\204Ø§Ø¯ Ø¥Ù\205 Ø¥Ø³ Ø³Ù\212 Ø¢Ù\212',
        sector: 'Index Funds',
        website: 'www.albiladmsci.com'
    },
    {
        symbol: '9402:TADAWUL',
        tickerSymbol: '9402',
        name: 'Alawwal MT30',
        name_ar: 'Ø§Ù\204Ø£Ù\210Ù\204 Ø¥Ù\205 ØªÙ\212 30',
        sector: 'Index Funds',
        website: 'www.alawwalmt30.com'
    },
    {
        symbol: '9403:TADAWUL',
        tickerSymbol: '9403',
        name: 'Albilad Sukuk',
        name_ar: 'Ø§Ù\204Ø¨Ù\204Ø§Ø¯ ØµÙ\203Ù\210Ù\203',
        sector: 'Index Funds',
        website: 'www.albiladsukuk.com'
    },
    {
        symbol: '4701:TADAWUL',
        tickerSymbol: '4701',
        name: 'ADITF',
        name_ar: 'Ø£Ø¯Ù\212ØªÙ\201',
        sector: 'Index Funds',
        website: 'www.aditf.com.sa'
    },
    {
        symbol: '9400:TADAWUL',
        tickerSymbol: '9400',
        name: 'Yaqeen 30',
        name_ar: 'Ù\212Ù\202Ù\212Ù\206 30',
        sector: 'Index Funds',
        website: 'www.yaqeen30.com'
    },
    {
        symbol: '9401:TADAWUL',
        tickerSymbol: '9401',
        name: 'Yaqeen Petrochemical',
        name_ar: 'Ù\212Ù\202Ù\212Ù\206 Ù\204Ù\204Ø¨ØªØ±Ù\210Ù\203Ù\212Ù\205Ø§Ù\210Ù\212Ø§Øª',
        sector: 'Index Funds',
        website: 'www.yaqeenpetrochemical.com'
    },
    {
        symbol: '9404:TADAWUL',
        tickerSymbol: '9404',
        name: 'Alinma Sukuk',
        name_ar: 'Ø§Ù\204Ø¥Ù\206Ù\205Ø§¡ ØµÙ\203Ù\210Ù\203',
        sector: 'Index Funds',
        website: 'www.alinmasukuk.com'
    },
    {
        symbol: '9405:TADAWUL',
        tickerSymbol: '9405',
        name: 'Albilad Gold',
        name_ar: 'Ø§Ù\204Ø¨Ù\204Ø§Ø¯ Ø°Ù\207Ø¨',
        sector: 'Index Funds',
        website: 'www.albiladgold.com'
    },
    {
        symbol: '4700:TADAWUL',
        tickerSymbol: '4700',
        name: 'Alkhabeer Income',
        name_ar: 'Ø§Ù\204Ø®Ø¨Ù\212Ø± Ø¯Ø®Ù\204',
        sector: 'Index Funds',
        website: 'www.alkhabeerincome.com'
    },
    {
        symbol: '9407:TADAWUL',
        tickerSymbol: '9407',
        name: 'Albilad US Tech',
        name_ar: 'Ø§Ù\204Ø¨Ù\204Ø§Ø¯ Ø§Ù\204ØªÙ\203Ù\206Ù\210Ù\204Ù\210Ø¬Ù\212Ø§ Ø§Ù\204Ø£Ù\205Ø±Ù\212Ù\203
\212Ø©',
        sector: 'Index Funds',
        website: 'www.albiladustech.com'
```

```ts
        },
        {
            symbol: '9408:TADAWUL',
            tickerSymbol: '9408',
            name: 'Albilad Saudi Growth',
            name_ar: 'اﻟﺒﻼد اﻟﻨﻤﻮ اﻟﺴﻌﻮدي',
            sector: 'Index Funds',
            website: 'www.albiladsaudigrowth.com'
        },
        {
            symbol: '4702:TADAWUL',
            tickerSymbol: '4702',
            name: 'Alkhabeer Income 2030',
            name_ar: 'اﻟﺨﺒﻴﺮ دﺧﻞ 2030',
            sector: 'Index Funds',
            website: 'www.alkhabeerincome2030.com'
        },
        {
            symbol: '4703:TADAWUL',
            tickerSymbol: '4703',
            name: 'Sedco Multi Asset',
            name_ar: 'سدكو ﻣﺘﻌﺪد اﻷﺻﻮل',
            sector: 'Index Funds',
            website: 'www.sedcomultiasset.com'
        },
        {
            symbol: '9410:TADAWUL',
            tickerSymbol: '9410',
            name: 'Albilad Hong Kong China',
            name_ar: 'اﻟﺒﻼد ﻫﻮﻧﻎ ﻛﻮﻧﻎ اﻟﺼﻴﻦ',
            sector: 'Index Funds',
            website: 'www.albiladhongkongchina.com'
        },
        {
            symbol: '9411:TADAWUL',
            tickerSymbol: '9411',
            name: 'SAB HK',
            name_ar: 'ساب ﻫﻮﻧﻎ ﻛﻮﻧﻎ',
            sector: 'Index Funds',
            website: 'www.sabhk.com'
        }
    ]
    // Helper functions
export function findSaudiSymbol(query: string) {
    const normalizedQuery = query.toLowerCase();

    return SAUDI_SYMBOLS.find(stock =>
        stock.symbol.toLowerCase().includes(normalizedQuery) ||
        stock.tickerSymbol.toLowerCase().includes(normalizedQuery) ||
        stock.name.toLowerCase().includes(normalizedQuery) ||
        stock.name_ar.includes(query)
    );
}

export function getSectorCompanies(sector: string) {
    return SAUDI_SYMBOLS.filter(stock =>
        stock.sector.toLowerCase() === sector.toLowerCase()
    );
}

export function formatSaudiSymbol(symbol: string | number) {
    const stock = SAUDI_SYMBOLS.find(s =>
        s.tickerSymbol === symbol.toString() ||
        s.symbol === symbol.toString()
    );

    if (!stock) return `${symbol}:TADAWUL`;
    return stock.symbol;
}

export function searchStocks(query: string) {
    const normalizedQuery = query.toLowerCase();
    return SAUDI_SYMBOLS.filter(stock =>
        stock.symbol.toLowerCase().includes(normalizedQuery) ||
        stock.tickerSymbol.toLowerCase().includes(normalizedQuery) ||
        stock.name.toLowerCase().includes(normalizedQuery) ||
        stock.name_ar.includes(query) ||
        stock.sector.toLowerCase().includes(normalizedQuery) ||
        stock.website.toLowerCase().includes(normalizedQuery)
    );
}


export function getAllSymbols() {
    return SAUDI_SYMBOLS.map(stock => stock.symbol);
}
```

```typescript
// lib/utils/format.ts
export const formatCurrency = (value: number | null | undefined, compact = false): string => {
  if (value === null || value === undefined) return 'N/A';

  try {
    if (compact) {
      if (Math.abs(value) >= 1e12) {
        return `SAR ${(value / 1e12).toFixed(2)}T`;
      }
      if (Math.abs(value) >= 1e9) {
        return `SAR ${(value / 1e9).toFixed(2)}B`;
      }
      if (Math.abs(value) >= 1e6) {
        return `SAR ${(value / 1e6).toFixed(2)}M`;
      }
      if (Math.abs(value) >= 1e3) {
        return `SAR ${(value / 1e3).toFixed(2)}K`;
      }
    }

    return new Intl.NumberFormat('en-US', {
      style: 'currency',
      currency: 'SAR',
      minimumFractionDigits: 2,
      maximumFractionDigits: 2
    }).format(value);
  } catch (error) {
    return `SAR ${value.toFixed(2)}`;
  }
};

export const formatNumber = (value: number | null | undefined, decimals = 2): string => {
  if (value === null || value === undefined) return 'N/A';
  // return value.toFixed(decimals);
};

export const formatPercent = (value: number | null | undefined, decimals = 2): string => {
  if (value === null || value === undefined) return 'N/A';
  return `${value.toFixed(decimals)}%`;
};

export const formatCompactNumber = (value: number | null | undefined): string => {
  if (value === null || value === undefined) return 'N/A';

  if (Math.abs(value) >= 1e9) {
    return `${(value / 1e9).toFixed(2)}B`;
  }
  if (Math.abs(value) >= 1e6) {
    return `${(value / 1e6).toFixed(2)}M`;
  }
  if (Math.abs(value) >= 1e3) {
    return `${(value / 1e3).toFixed(2)}K`;
  }
  return value.toLocaleString();
};

export const formatChange = (value: number | null | undefined): string => {
  if (value === null || value === undefined) return 'N/A';
  const sign = value >= 0 ? '+' : '';
  return `${sign}${value.toFixed(2)}%`;
};
```

```typescript
// // hooks/useMarketSocket.ts
// import { useEffect, useRef, useState } from 'react';
// import { WS_URL, API_KEY } from '@/app/config/market';

// export interface MarketQuoteData {
//   symbol: string;
//   name: string;
//   price: number;
//   change: number;
//   percent_change: number;
//   volume: number;
//   high: number;
//   low: number;
//   open: number;
//   close: number;
//   timestamp: string;
// }

// interface WebSocketMessage {
//   event: string;
//   symbol: string;
//   price: string;
//   change: string;
//   percent_change: string;
//   volume: string;
//   high: string;
//   low: string;
//   open: string;
//   close: string;
//   timestamp: string;
// }

// interface WebSocketSubscribeMessage {
//   action: 'subscribe';
//   params: {
//     symbols: string;
//   };
// }

// export function useMarketSocket(symbols: string[]) {
//   const [quotes, setQuotes] = useState<Record<string, MarketQuoteData>>({});
//   const [status, setStatus] = useState<'connecting' | 'connected' | 'disconnected'>('disconnected');
//   const [error, setError] = useState<string | null>(null);
//   const ws = useRef<WebSocket | null>(null);
//   const reconnectTimeout = useRef<NodeJS.Timeout>();

//   const connect = () => {
//     try {
//       setStatus('connecting');
//       ws.current = new WebSocket('${WS_URL}?apikey=${API_KEY}');

//       ws.current.onopen = () => {
//         console.log('WebSocket Connected');
//         setStatus('connected');
//         setError(null);
//         if (ws.current?.readyState === WebSocket.OPEN && symbols.length > 0) {
//           const message: WebSocketSubscribeMessage = {
//             action: 'subscribe',
//             params: {
//               symbols: symbols.join(',')
//             }
//           };
//           ws.current.send(JSON.stringify(message));
//         }
//       };

//       ws.current.onmessage = (event) => {
//         try {
//           const data: WebSocketMessage = JSON.parse(event.data);

//           if (data.event === 'price') {
//             setQuotes(prev => ({
//               ...prev,
//               [data.symbol]: {
//                 symbol: data.symbol,
//                 name: prev[data.symbol]?.name || data.symbol,
//                 price: parseFloat(data.price) || 0,
//                 change: parseFloat(data.change) || 0,
//                 percent_change: parseFloat(data.percent_change) || 0,
//                 volume: parseInt(data.volume) || 0,
//                 high: parseFloat(data.high) || 0,
//                 low: parseFloat(data.low) || 0,
//                 open: parseFloat(data.open) || 0,
//                 close: parseFloat(data.close) || 0,
//                 timestamp: data.timestamp
//               }
```

```
//                 }));
//             }
//         } catch (error) {
//             console.error('Error parsing WebSocket message:', error);
//         }
//     };

//     ws.current.onclose = () => {
//       console.log('WebSocket Disconnected');
//       setStatus('disconnected');
//       // Attempt to reconnect after 5 seconds
//       reconnectTimeout.current = setTimeout(() => {
//         connect();
//       }, 5000);
//     };

//     ws.current.onerror = (error) => {
//       console.error('WebSocket error:', error);
//       setError('WebSocket connection error');
//       setStatus('disconnected');
//     };
//   } catch (error) {
//     console.error('Error creating WebSocket:', error);
//     setError('Failed to create WebSocket connection');
//     setStatus('disconnected');
//   }
// };

//   useEffect(() => {
//     if (symbols.length > 0) {
//       connect();
//     }

//     return () => {
//       if (ws.current) {
//         ws.current.close();
//       }
//       if (reconnectTimeout.current) {
//         clearTimeout(reconnectTimeout.current);
//       }
//     };
//   }, [symbols]);

//   // Resubscribe when symbols change and we're already connected
//   useEffect(() => {
//     if (ws.current?.readyState === WebSocket.OPEN && symbols.length > 0) {
//       const message: WebSocketSubscribeMessage = {
//         action: 'subscribe',
//         params: {
//           symbols: symbols.join(',')
//         }
//       };
//       ws.current.send(JSON.stringify(message));
//     }
//   }, [symbols]);

//   return {
//     quotes,
//     status,
//     error,
//     isConnected: status === 'connected'
//   };
// }

// // Helper function to format quote data for display
// export function formatQuoteData(quote: MarketQuoteData) {
//   return {
//     symbol: quote.symbol,
//     name: quote.name,
//     price: quote.price.toFixed(2),
//     change: quote.change.toFixed(2),
//     percentChange: quote.percent_change.toFixed(2),
//     volume: quote.volume.toLocaleString(),
//     isPositive: quote.change >= 0,
//     timestamp: new Date(quote.timestamp).toLocaleString()
//   };
// }
```

```
// // app/hooks/useStockData.ts
// import { useState, useEffect } from 'react';
// import { API_KEY, API_URL, formatSymbol } from '@/app/config/market';

// interface UseStockDataResult {
//   data: StockData | null;
//   loading: boolean;
//   error: string | null;
// }

// export function useStockData(tickerSymbol: string): UseStockDataResult {
//   const [data, setData] = useState<StockData | null>(null);
//   const [loading, setLoading] = useState(true);
//   const [error, setError] = useState<string | null>(null);

//   useEffect(() => {
//     const fetchStockData = async () => {
//       if (!tickerSymbol) {
//         setLoading(false);
//         return;
//       }

//       try {
//         setLoading(true);
//         const symbol = '${tickerSymbol}:TADAWUL';

//         // Fetch all data in parallel
//         const [statsRes, quoteRes, timeSeriesRes, profileRes] = await Promise.all([
//           fetch('${API_URL}/statistics?symbol=${symbol}&apikey=${API_KEY}'),
//           fetch('${API_URL}/quote?symbol=${symbol}&apikey=${API_KEY}'),
//           fetch('${API_URL}/time_series?symbol=${symbol}&interval=1day&outputsize=90&apikey=${API_KEY}'),
//           fetch('${API_URL}/profile?symbol=${symbol}&apikey=${API_KEY}')
//         ]);

//         // Check responses individually
//         const responses = {
//           stats: statsRes.ok,
//           quote: quoteRes.ok,
//           timeSeries: timeSeriesRes.ok,
//           profile: profileRes.ok
//         };

//         console.log('API Response Status:', responses);

//         // Get the JSON data, handling potential errors for each request
//         const [statsData, quoteData, timeSeriesData, profileData] = await Promise.all([
//           statsRes.json().catch(e => ({ statistics: {} })),
//           quoteRes.json().catch(e => ({})),
//           timeSeriesRes.json().catch(e => ({ values: [] })),
//           profileRes.json().catch(e => null)
//         ]);

//         // Log raw data for debugging
//         console.log('Raw Profile Data:', profileData);

//         const stockData: StockData = {
//           quote: {
//             symbol: symbol,
//             name: quoteData.name || '',
//             close: String(quoteData.close || quoteData.price || '0'),
//             change: String(quoteData.change || '0'),
//             percent_change: String(quoteData.percent_change || '0'),
//             high: String(quoteData.high || '0'),
//             low: String(quoteData.low || '0'),
//             open: String(quoteData.open || '0'),
//             volume: String(quoteData.volume || '0')
//           },
//           timeSeries: {
//             values: timeSeriesData.values?.map((item: any) => ({
//               datetime: item.datetime,
//               open: String(item.open || '0'),
//               high: String(item.high || '0'),
//               low: String(item.low || '0'),
//               close: String(item.close || '0'),
//               volume: String(item.volume || '0')
//             })) || []
//           },
//           statistics: statsData.statistics,
//           profile: profileData ? {
//             name: profileData.name || '',
//             exchange: profileData.exchange || 'TADAWUL',
//             mic_code: profileData.mic_code || '',
//             sector: profileData.sector || '',
//             industry: profileData.industry || '',
//             employees: profileData.employees || 0,
//             website: profileData.website || '',
```

```
//                description: profileData.description || '',
//                type: profileData.type || 'Common Stock',
//                CEO: profileData.CEO || '',
//                address: profileData.address || '',
//                address2: profileData.address2 || '',
//                city: profileData.city || '',
//                zip: profileData.zip || '',
//                state: profileData.state || '',
//                country: profileData.country || '',
//                phone: profileData.phone || ''
//            } : undefined
//        };

//        setData(stockData);
//        setError(null);
//      } catch (err) {
//        console.error('Error fetching stock data:', err);
//        setError(err instanceof Error ? err.message : 'Failed to load stock data');
//        setData(null);
//      } finally {
//        setLoading(false);
//      }
//    };

//    fetchStockData();
//    // Refresh data every minute
//    const interval = setInterval(fetchStockData, 60000);
//    return () => clearInterval(interval);
//  }, [tickerSymbol]);

//   return { data, loading, error };
// }
```

```
// // hooks/useSaudiMarket.ts
// import { useState, useEffect } from 'react';

// interface SaudiStock {
//    symbol: string;
//    name: string;
//    currency: string;
//    exchange: string;
//    mic_code: string;
//    country: string;
//    type: string;
// }

// interface StockQuote {
//    symbol: string;
//    price: string;
//    change: string;
//    percent_change: string;
//    volume: string;
//    timestamp: string;
// }

// interface RealTimeData {
//    price: number;
//    change: number;
//    percentChange: number;
//    volume: number;
//    timestamp: string;
// }

// interface StockData {
//    stock: SaudiStock;
//    quote?: StockQuote;
//    realTimeData?: RealTimeData;
// }

// interface WebSocketMessage {
//    event: string;
//    symbol: string;
//    price: string;
//    change: string;
//    percent_change: string;
//    volume: string;
//    timestamp: string;
// }

// export function useSaudiMarket() {
//    const [stocks, setStocks] = useState<StockData[]>([]);
//    const [loading, setLoading] = useState(true);
//    const [error, setError] = useState<string | null>(null);
//    const [selectedTimeframe, setSelectedTimeframe] = useState('1day');

//    const fetchSaudiMarketData = async () => {
//      try {
//        setLoading(true);
//        // Fetch Saudi stocks list
//        const stocksResponse = await fetch(
//          `https://api.twelvedata.com/stocks?country=Saudi Arabia&apikey=${process.env.NEXT_PUBLIC_TWELVE_DATA
_API_KEY}`
//        );

//        if (!stocksResponse.ok) {
//          throw new Error('Failed to fetch stocks list');
//        }

//        const stocksData = await stocksResponse.json();
//        const stocksList: SaudiStock[] = stocksData.data || stocksData;

//        // Take first 20 stocks to avoid rate limits
//        const stocksToFetch = stocksList.slice(0, 20);
//        const symbols = stocksToFetch.map(stock => stock.symbol).join(',');

//        // Fetch quotes for all stocks at once
//        const quotesResponse = await fetch(
//          `https://api.twelvedata.com/quote?symbol=${symbols}&apikey=${process.env.NEXT_PUBLIC_TWELVE_DATA_API
_KEY}`
//        );

//        if (!quotesResponse.ok) {
//          throw new Error('Failed to fetch quotes');
//        }

//        const quotesData = await quotesResponse.json();
//        let quotesArray: StockQuote[];

//        if (Array.isArray(quotesData)) {
```

```
//          quotesArray = quotesData;
//        } else if (quotesData.data) {
//          quotesArray = Array.isArray(quotesData.data) ? quotesData.data : [quotesData.data];
//        } else {
//          quotesArray = [quotesData];
//        }

//        // Combine stocks with their quotes
//        const combinedData = stocksToFetch.map(stock => {
//          const quote = quotesArray.find(q => q.symbol === stock.symbol);
//          return {
//            stock,
//            quote,
//            realTimeData: quote ? {
//              price: parseFloat(quote.price),
//              change: parseFloat(quote.change),
//              percentChange: parseFloat(quote.percent_change),
//              volume: parseInt(quote.volume),
//              timestamp: quote.timestamp
//            } : undefined
//          };
//        });

//        setStocks(combinedData);
//        setupWebSocket(symbols.split(','));
//        setError(null);
//      } catch (err) {
//        console.error('Error fetching market data:', err);
//        setError(err instanceof Error ? err.message : 'An error occurred');
//      } finally {
//        setLoading(false);
//      }
//    };

//    const setupWebSocket = (symbols: string[]) => {
//      const ws = new WebSocket(
//        `wss://ws.twelvedata.com/v1/quotes/price?apikey=${process.env.NEXT_PUBLIC_TWELVE_DATA_API_KEY}`
//      );

//      ws.onopen = () => {
//        ws.send(JSON.stringify({
//          action: 'subscribe',
//          params: {
//            symbols: symbols.join(',')
//          }
//        }));
//      };

//      ws.onmessage = (event: MessageEvent) => {
//        try {
//          const data = JSON.parse(event.data) as WebSocketMessage;
//          if (data.event === 'price') {
//            setStocks(prev => prev.map(stock => {
//              if (stock.stock.symbol === data.symbol) {
//                return {
//                  ...stock,
//                  realTimeData: {
//                    price: parseFloat(data.price),
//                    change: parseFloat(data.change),
//                    percentChange: parseFloat(data.percent_change),
//                    volume: parseInt(data.volume),
//                    timestamp: data.timestamp
//                  }
//                };
//              }
//              return stock;
//            }));
//          }
//        } catch (error) {
//          console.error('WebSocket message parsing error:', error);
//        }
//      };

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

//      return () => {
//        if (ws.readyState === WebSocket.OPEN) {
//          ws.close();
//        }
//      };
//    };

//    useEffect(() => {
//      fetchSaudiMarketData();
```

```
//     // Refresh data every minute
//     const interval = setInterval(fetchSaudiMarketData, 60000);
//     return () => clearInterval(interval);
//   }, [selectedTimeframe]);

//   const changeTimeframe = (timeframe: string) => {
//     setSelectedTimeframe(timeframe);
//   };

//   return {
//     stocks,
//     loading,
//     error,
//     selectedTimeframe,
//     changeTimeframe,
//     refreshData: fetchSaudiMarketData
//   };
// }
```

```
//     // Refresh data every minute
//     const interval = setInterval(fetchSaudiMarketData, 60000);
//     return () => clearInterval(interval);
//   }, [selectedTimeframe]);

//   const changeTimeframe = (timeframe: string) => {
//     setSelectedTimeframe(timeframe);
//   };

//   return {
//     stocks,
//     loading,
//     error,
//     selectedTimeframe,
//     changeTimeframe,
//     refreshData: fetchSaudiMarketData
//   };
// }
```

```typescript
// /app/lib/context/financeContextUtils.ts

import { FinancialMetadata } from './types';

export class FinanceContextAnalyzer {
  private static readonly FINANCIAL_INSTRUMENTS = [
    'stock', 'bond', 'etf', 'option', 'future', 'forex', 'crypto',
    'dividend', 'mutual fund', 'index', 'commodity'
  ];

  private static readonly TECHNICAL_INDICATORS = [
    'moving average', 'MACD', 'RSI', 'bollinger', 'volume', 'momentum',
    'support', 'resistance', 'trend', 'volatility'
  ];

  private static readonly FUNDAMENTAL_FACTORS = [
    'earnings', 'revenue', 'profit', 'margin', 'PE ratio', 'market cap',
    'dividend yield', 'cash flow', 'debt', 'assets'
  ];

  private static readonly MARKET_KEYWORDS = [
    'bull', 'bear', 'volatile', 'correction', 'crash', 'rally',
    'uptick', 'downturn', 'market', 'trade', 'buy', 'sell'
  ];

  public static analyzeFinancialMessage(message: string): FinancialMetadata {
    const words = message.toLowerCase().split(/\s+/);

    return {
      instruments: this.extractFinancialInstruments(message),
      companies: this.extractCompanies(message),
      markets: this.extractMarkets(message),
      metrics: this.extractMetrics(message),
      timeframes: this.extractTimeframes(message),
      sentiment: this.analyzeMarketSentiment(message),
      confidenceScore: this.calculateConfidenceScore(message),
      riskLevel: this.assessRiskLevel(message),
      technicalIndicators: this.extractTechnicalIndicators(message),
      fundamentalFactors: this.extractFundamentalFactors(message)
    };
  }

  private static extractFinancialInstruments(message: string): string[] {
    const text = message.toLowerCase();
    return this.FINANCIAL_INSTRUMENTS.filter(instrument =>
      text.includes(instrument)
    );
  }

  private static extractCompanies(message: string): string[] {
    // Look for potential company names (capitalized words)
    const companies = new Set<string>();
    const words = message.split(/\s+/);

    for (let i = 0; i < words.length; i++) {
      const word = words[i];
      if (word.length > 1 &&
          word[0] === word[0].toUpperCase() &&
          word[1] === word[1].toLowerCase()) {
        companies.add(word);
      }
      // Check for stock symbols (uppercase letters)
      if (word.match(/^\$?[A-Z]{1,5}$/)) {
        companies.add(word);
      }
    }

    return Array.from(companies);
  }

  private static extractMarkets(message: string): string[] {
    const commonMarkets = [
      'NYSE', 'NASDAQ', 'DOW', 'S&P', 'FTSE', 'market',
      'exchange', 'trading'
    ];

    return commonMarkets.filter(market =>
      message.toLowerCase().includes(market.toLowerCase())
    );
  }

  private static extractMetrics(message: string): string[] {
    const commonMetrics = [
      'price', 'volume', 'volatility', 'return', 'yield',
      'ratio', 'growth', 'loss', 'gain', 'percentage'
    ];
```

```typescript
    return commonMetrics.filter(metric =>
      message.toLowerCase().includes(metric)
    );
  }

  private static extractTimeframes(message: string): string[] {
    const timeframes = [
      'day', 'week', 'month', 'year', 'quarter',
      'short-term', 'long-term', 'intraday'
    ];

    return timeframes.filter(timeframe =>
      message.toLowerCase().includes(timeframe)
    );
  }

  private static analyzeMarketSentiment(message: string): 'bullish' | 'bearish' | 'neutral' {
    const text = message.toLowerCase();

    const bullishWords = ['buy', 'bull', 'up', 'growth', 'positive', 'increase'];
    const bearishWords = ['sell', 'bear', 'down', 'decline', 'negative', 'decrease'];

    let sentiment = 0;

    bullishWords.forEach(word => {
      if (text.includes(word)) sentiment++;
    });

    bearishWords.forEach(word => {
      if (text.includes(word)) sentiment--;
    });

    if (sentiment > 0) return 'bullish';
    if (sentiment < 0) return 'bearish';
    return 'neutral';
  }

  private static calculateConfidenceScore(message: string): number {
    let score = 0.5; // Base confidence

    // Increase confidence based on presence of specific data
    if (this.extractMetrics(message).length > 0) score += 0.1;
    if (this.extractTechnicalIndicators(message).length > 0) score += 0.1;
    if (this.extractFundamentalFactors(message).length > 0) score += 0.1;

    // Cap the score at 1.0
    return Math.min(score, 1.0);
  }

  private static assessRiskLevel(message: string): 'low' | 'medium' | 'high' {
    const text = message.toLowerCase();

    const highRiskWords = ['volatile', 'risky', 'speculation', 'aggressive'];
    const lowRiskWords = ['safe', 'conservative', 'stable', 'defensive'];

    let riskScore = 0;

    highRiskWords.forEach(word => {
      if (text.includes(word)) riskScore++;
    });

    lowRiskWords.forEach(word => {
      if (text.includes(word)) riskScore--;
    });

    if (riskScore > 0) return 'high';
    if (riskScore < 0) return 'low';
    return 'medium';
  }

  private static extractTechnicalIndicators(message: string): string[] {
    return this.TECHNICAL_INDICATORS.filter(indicator =>
      message.toLowerCase().includes(indicator)
    );
  }

  private static extractFundamentalFactors(message: string): string[] {
    return this.FUNDAMENTAL_FACTORS.filter(factor =>
      message.toLowerCase().includes(factor)
    );
  }
}
```

```typescript
// /app/lib/context/types.ts

// Base metadata interfaces
export interface FinancialMetrics {
  pe_ratio?: number;
  profit_margin?: number;
  revenue_growth?: number;
  dividend_yield?: number;
  market_cap?: number;
  volume?: number;
  [key: string]: number | undefined;
}

export interface TechnicalIndicators {
  rsi?: number;
  macd?: {
    value: number;
    signal: number;
    histogram: number;
  };
  moving_averages?: {
    sma_50?: number;
    sma_200?: number;
    ema_20?: number;
  };
  support_levels?: number[];
  resistance_levels?: number[];
}



// Enhanced company context
export interface CompanyContext {
  symbol: string;
  name: string;
  name_ar?: string;
  sector?: string;
  lastMentioned: number;
  metadata: {
    recentTopics: string[];
    financialMetrics?: FinancialMetrics;
    technicalIndicators?: TechnicalIndicators;
    fundamentalFactors: string[];
    marketMetrics: {
      price: number;
      change: number;
      changePercent: number;
      volume: number;
      timestamp: number;
    };
    analysisContext: {
      mentionCount: number;
      sentiment: 'bullish' | 'bearish' | 'neutral';
      confidence: number;
    };
  };
}

// Enhanced market context
export interface MarketContext {
  currentMarketHours: boolean;
  marketConditions: 'bull' | 'bear' | 'neutral';
  volatilityLevel: 'low' | 'medium' | 'high';
  timestamp: number;
  majorIndices: {
    [key: string]: {
      value: number;
      change: number;
      changePercent: number;
      volume: number;
    };
  };
  sectorPerformance: {
    [sector: string]: {
      change: number;
      volume: number;
      topMovers: Array<{
        symbol: string;
        change: number;
      }>;
    };
  };
  marketBreadth: {
    advancers: number;
    decliners: number;
    unchanged: number;
```

```typescript
    totalVolume: number;
  };
}

// Enhanced message interface
export interface Message {
  id: string;
  role: 'user' | 'assistant';
  content: string;
  timestamp: number;
  metadata: {
    instruments: string[];
    companies: string[];
    markets: string[];
    metrics: string[];
    timeframes: string[];
    sentiment: 'bullish' | 'bearish' | 'neutral';
    confidence: number;
    riskLevel: 'low' | 'medium' | 'high';
    technicalIndicators: string[];
    fundamentalFactors: string[];
    analysisType: 'technical' | 'fundamental' | 'mixed' | 'general';
    priority: 'critical' | 'high' | 'medium' | 'low';
    language?: 'en' | 'ar';
  };
  contextScore?: ContextScore;
}

// Enhanced conversation state
export interface ConversationState {
  id: string;
  currentCompany?: CompanyContext;
  recentCompanies: CompanyContext[];
  messageHistory: Array<Message>;
  analysisContext: {
    activeTechnicalAnalysis: boolean;
    fundamentalAnalysisActive: boolean;
    lastMarketUpdate: number;
    confidence: number;
    contextQuality: 'excellent' | 'good' | 'limited' | 'poor';
  };
  userPreferences: {
    preferredMarkets: string[];
    riskTolerance: 'conservative' | 'moderate' | 'aggressive';
    preferredTimeframe: 'short' | 'medium' | 'long';
    technicalAnalysisLevel: 'basic' | 'intermediate' | 'advanced';
    language: 'en' | 'ar';
  };
  marketContext?: MarketContext;
  lastActivity: number;
  conversationStart: number;
}

// Context window management
export interface ContextWindow {
  maxSize: number;
  currentSize: number;
  priorityThreshold: number;
  recentMessages: Message[];
  marketSnapshots: MarketContext[];
  keyInsights: Array<{
    content: string;
    timestamp: number;
    priority: 'critical' | 'high' | 'medium' | 'low';
    category: 'market' | 'company' | 'sector' | 'general';
  }>;
}

export interface ContextScore {
  messageId: string;
  marketRelevance: number;
  timeRelevance: number;
  instrumentRelevance: number;
  userPriorityScore: number;
  finalScore: number;
  timestamp: number;
}

export interface AnalysisResponse {
  companies: CompanyContext[];
  sectors: Set<string>;
  confidence: number;
  analysisTimestamp: number;
  marketSnapshot?: MarketContext;
}
```

```typescript
// Response schema for API
export interface ChatResponse {
  response: string;
  thinking: string;
  user_mood: 'positive' | 'neutral' | 'negative' | 'curious' | 'frustrated' | 'confused';
  suggested_questions: string[];
  financial_context: {
    instruments_mentioned: string[];
    market_sentiment: 'bullish' | 'bearish' | 'neutral';
    risk_level: 'low' | 'medium' | 'high';
    confidence_score: number;
    technical_indicators: string[];
    fundamental_factors: string[];
    sectors: string[];
    economic_indicators: string[];
    analysis_type: 'technical' | 'fundamental' | 'mixed' | 'general';
    market_data: any | null;
    key_insights: string[];
  };
  debug: {
    context_used: boolean;
    market_data_used: boolean;
    context_confidence: number;
    analysis_quality: string;
  };
}
```

```typescript
// /app/lib/context/financeContextManager.ts

import {
  Message,
  ConversationContext,
  ContextWindow,
  ContextScore,
  ContextPriority,
  MarketContext,
  FinancialMetadata
} from './types';

export class FinanceContextManager {
  private readonly maxWindowSize: number;
  private contexts: Map<string, ConversationContext>;
  private activeWindow: ContextWindow;
  private marketContext: MarketContext;

  constructor(maxWindowSize: number = 4096) {
    this.maxWindowSize = maxWindowSize;
    this.contexts = new Map();
    this.activeWindow = {
      maxSize: maxWindowSize,
      currentSize: 0,
      priorityThreshold: 0.7,
      recentMessages: [],
      marketSnapshots: [],
      keyInsights: []
    };
    this.marketContext = {
      currentMarketHours: false,
      marketConditions: 'neutral',
      volatilityLevel: 'low',
      majorIndices: {}
    };
  }

  private calculateFinancialContextScore(message: Message, currentTime: number): ContextScore {
    const timeRelevance = Math.exp(-(currentTime - message.timestamp) / (1000 * 60 * 60));
    const metadata = message.metadata;

    let marketRelevance = 0.5;
    if (metadata) {
      if (metadata.instruments?.length) marketRelevance += 0.1;
      if (metadata.markets?.length) marketRelevance += 0.1;
      if (metadata.technicalIndicators?.length) marketRelevance += 0.1;
      if (metadata.fundamentalFactors?.length) marketRelevance += 0.1;
    }

    let instrumentRelevance = 0.5;
    if (metadata?.instruments?.length) {
      instrumentRelevance = 0.7;
      if (this.isActiveInstrument(metadata.instruments[0])) {
        instrumentRelevance = 0.9;
      }
    }

    let userPriorityScore = 0.5;
    if (metadata) {
      if (metadata.riskLevel === 'high') userPriorityScore += 0.2;
      if (metadata.confidenceScore && metadata.confidenceScore > 0.8) userPriorityScore += 0.1;
    }

    return {
      messageId: message.id,
      marketRelevance,
      timeRelevance,
      instrumentRelevance,
      userPriorityScore,
      finalScore: (
        marketRelevance * 0.3 +
        timeRelevance * 0.2 +
        instrumentRelevance * 0.3 +
        userPriorityScore * 0.2
      )
    };
  }

  private isActiveInstrument(instrument: string): boolean {
    const contexts = Array.from(this.contexts.values());
    return contexts.some(context =>
      context.activeInstruments.includes(instrument)
    );
  }

  public addMessage(conversationId: string, message: Message): void {
```

```typescript
    let context = this.contexts.get(conversationId);

    if (!context) {
      context = {
        conversationId,
        messages: [],
        activeInstruments: [],
        activeTechnicalAnalysis: false,
        fundamentalAnalysisActive: false,
        lastUpdated: Date.now(),
        userPreferences: {
          preferredMarkets: [],
          riskTolerance: 'moderate',
          preferredTimeframe: 'medium',
          technicalAnalysisLevel: 'basic'
        }
      };
      this.contexts.set(conversationId, context);
    }

    if (message.metadata?.instruments) {
      context.activeInstruments = Array.from(new Set([
        ...context.activeInstruments,
        ...message.metadata.instruments
      ]));
    }

    if (message.metadata?.technicalIndicators?.length) {
      context.activeTechnicalAnalysis = true;
    }
    if (message.metadata?.fundamentalFactors?.length) {
      context.fundamentalAnalysisActive = true;
    }

    context.messages.push(message);
    context.lastUpdated = Date.now();
    this.updateActiveWindow(context);
  }

  private updateActiveWindow(context: ConversationContext): void {
    const currentTime = Date.now();

    const scoredMessages = context.messages
      .map(msg => ({
        message: msg,
        score: this.calculateFinancialContextScore(msg, currentTime)
      }))
      .sort((a, b) => b.score.finalScore - a.score.finalScore);

    this.activeWindow.recentMessages = scoredMessages
      .filter(scored => scored.score.finalScore > this.activeWindow.priorityThreshold)
      .map(scored => scored.message)
      .slice(0, Math.floor(this.maxWindowSize / 2));

    this.activeWindow.keyInsights = this.extractFinancialInsights(
      scoredMessages.slice(0, 5).map(scored => scored.message)
    );

    if (this.shouldTakeMarketSnapshot(context)) {
      this.activeWindow.marketSnapshots.push({ ...this.marketContext });
    }
  }

  private shouldTakeMarketSnapshot(context: ConversationContext): boolean {
    const latestMessage = context.messages[context.messages.length - 1];
    return Boolean(
      latestMessage.metadata?.markets?.length ||
      (this.marketContext.volatilityLevel === 'high' && Math.random() > 0.7)
    );
  }

  private extractFinancialInsights(messages: Message[]): string[] {
    const insights = new Set<string>();

    messages.forEach(message => {
      const metadata = message.metadata;
      if (metadata) {
        if (metadata.technicalIndicators?.length) {
          insights.add(`Technical Analysis: ${metadata.technicalIndicators.join(', ')}`);
        }

        if (metadata.fundamentalFactors?.length) {
          insights.add(`Fundamental Factors: ${metadata.fundamentalFactors.join(', ')}`);
        }

        if (metadata.sentiment) {
```

```typescript
          insights.add(`Market Sentiment: ${metadata.sentiment}`);
        }

        if (metadata.riskLevel) {
          insights.add(`Risk Level: ${metadata.riskLevel}`);
        }
      }
    });

    return Array.from(insights);
  }

  public getContextForPrompt(conversationId: string): string {
    const context = this.contexts.get(conversationId);
    if (!context) return '';

    const relevantMessages = this.activeWindow.recentMessages
      .map(msg => {
        const metadata = msg.metadata;
        return `${msg.role}: ${msg.content}
        ${metadata ? `
        Instruments: ${metadata.instruments?.join(', ') || 'none'}
        Technical Indicators: ${metadata.technicalIndicators?.join(', ') || 'none'}
        Sentiment: ${metadata.sentiment || 'neutral'}
        Risk Level: ${metadata.riskLevel || 'medium'}` : ''}`;
      })
      .join('\n\n');

    const marketContext = this.activeWindow.marketSnapshots.length > 0
      ? `\nCurrent Market Context:
        Market Hours: ${this.marketContext.currentMarketHours ? 'Open' : 'Closed'}
        Conditions: ${this.marketContext.marketConditions}
        Volatility: ${this.marketContext.volatilityLevel}`
      : '';

    const insights = this.activeWindow.keyInsights.length > 0
      ? `\nKey Insights:\n${this.activeWindow.keyInsights.join('\n')}`
      : '';

    return `${marketContext}\n${insights}\n\nRecent Discussion:\n${relevantMessages}`;
  }

  public updateMarketContext(newContext: Partial<MarketContext>): void {
    this.marketContext = { ...this.marketContext, ...newContext };
  }

  public getActiveInstruments(conversationId: string): string[] {
    return this.contexts.get(conversationId)?.activeInstruments || [];
  }

  public getCurrentMarketContext(): MarketContext {
    return this.marketContext;
  }

  public getKeyInsights(conversationId: string): string[] {
    return this.activeWindow.keyInsights;
  }
}
```

```typescript
// /app/lib/context/financeAnalyzer.ts

import { SAUDI_SYMBOLS } from '@/app/config/market';
import { CompanyContext } from './types';

interface CompanyMatch {
  symbol: string;
  name: string;
  sector?: string;
  isContinuedDiscussion: boolean;
  confidence: number;
}
interface MatchConfidence {
  symbolMatch: boolean;
  nameMatch: boolean;
  arabicMatch: boolean;
  partialMatch: boolean;
  contextualRelevance: boolean;
}
interface StockQuote {
  symbol: string;
  price: number;
  change: number;
  percentChange: number;
  volume: number;
  lastUpdated: string;
}
interface MarketData {
  [symbol: string]: {
    values: {
      close: number;
      open: number;
      high: number;
      low: number;
      volume: number;
      datetime: string;
    }[];
  };
}



export class FinanceAnalyzer {
  // Known financial instruments and companies
  private static readonly SAUDI_MARKET_TERMS = new Set([
    'tadawul', 'saudi stock', 'saudi market', 'tasi', 'nomu',
    'parallel market', 'main market', 'saudi exchange', 'cma'
  ]);

  private static readonly TECHNICAL_INDICATORS = new Set([
    'RSI', 'MACD', 'moving average', 'bollinger bands', 'support', 'resistance',
    'volume', 'momentum', 'trend', 'breakout', 'consolidation', 'divergence',
    'fibonacci', 'chart pattern', 'candlestick', 'oscillator', 'volatility',
    'EMA', 'SMA', 'price action', 'trend line', 'head and shoulders',
    'double top', 'double bottom', 'triangle pattern', 'flag pattern'
  ]);

  private static readonly FUNDAMENTAL_FACTORS = new Set([
    'PE ratio', 'P/E', 'EPS', 'revenue', 'earnings', 'profit margin', 'cash flow',
    'debt', 'assets', 'liabilities', 'market cap', 'dividend', 'book value',
    'ROE', 'ROI', 'EBITDA', 'gross margin', 'operating margin', 'net margin',
    'PEG ratio', 'debt to equity', 'current ratio', 'quick ratio',
    'dividend yield', 'payout ratio', 'price to book', 'price to sales'
  ]);

  private static readonly SECTORS = new Set(
    Array.from(new Set(SAUDI_SYMBOLS.map(stock => stock.sector)))
  );

  private static readonly RISK_TERMS = {
    high: ['volatile', 'risky', 'speculative', 'aggressive', 'unstable', 'uncertain'],
    medium: ['moderate', 'balanced', 'mixed', 'neutral'],
    low: ['safe', 'stable', 'conservative', 'defensive', 'reliable']
  };

  private static readonly SENTIMENT_TERMS = {
    bullish: ['up', 'rise', 'grow', 'positive', 'bullish', 'outperform', 'buy', 'strong',
              'increase', 'higher', 'upside', 'potential', 'gain', 'profit'],
    bearish: ['down', 'fall', 'decline', 'negative', 'bearish', 'underperform', 'sell', 'weak',
              'decrease', 'lower', 'downside', 'risk', 'loss', 'concern'],
    neutral: ['stable', 'steady', 'hold', 'neutral', 'balanced', 'maintain', 'unchanged']
  };

  private static readonly COMPANY_RELATED_TERMS = new Set([
    'revenue', 'profit', 'earnings', 'growth', 'performance', 'management',
    'strategy', 'market share', 'competition', 'expansion', 'dividend',
```

```typescript
    'announcement', 'report', 'guidance', 'forecast', 'outlook',
    'CEO', 'executive', 'board', 'shareholders', 'stock', 'shares',
    'price', 'valuation', 'target', 'rating', 'upgrade', 'downgrade'
  ]);

  private static calculateMatchConfidence(matches: {
    symbolMatch: boolean;
    nameMatch: boolean;
    arabicMatch: boolean;
    partialMatch: boolean;
    contextualRelevance: boolean;
  }): number {
    let confidence = 0;

    if (matches.symbolMatch) confidence += 0.4;
    if (matches.nameMatch) confidence += 0.3;
    if (matches.arabicMatch) confidence += 0.3;
    if (matches.partialMatch) confidence += 0.2;
    if (matches.contextualRelevance) confidence += 0.1;
    // Normalize confidence to be between 0 and 1
    return Math.min(1, confidence);
  }


  public static analyzeFinancialContext(
    message: string,
    currentCompany?: CompanyContext
  ): {
    confidence: number;
    companies: Array<{
      symbol: string;
      name: string;
      sector?: string;
      isContinuedDiscussion: boolean;
      confidence: number;
    }>;
    technicalIndicators: string[];
    fundamentalFactors: string[];
    sectors: string[];
    riskLevel: 'low' | 'medium' | 'high';
    sentiment: 'bullish' | 'bearish' | 'neutral';
    analysisType: 'technical' | 'fundamental' | 'mixed' | 'general';
    marketContext: {
      isSaudiMarket: boolean;
      mentionedMarket?: string;
    };
    companyContext: {
      hasExplicitMention: boolean;
      continuedDiscussion: boolean;
      discussionTopics: string[];
    };
  } {
    const text = message.toLowerCase();

    // Analyze company context first with enhanced detection
    const companiesAnalysis = this.analyzeCompanyContext(text, currentCompany);

    // Extract all indicators and factors
    const technical = this.extractTechnicalIndicators(text);
    const fundamental = this.extractFundamentalFactors(text);
    const sectors = this.extractSectors(text);

    // Determine market context
    const isSaudiMarket = this.checkSaudiMarketContext(text);

    // Analyze sentiment with enhanced company context
    const sentiment = companiesAnalysis.companies.length > 0
      ? this.analyzeSentimentWithContext(text, companiesAnalysis.companies[0])
      : this.determineSentiment(text);

    // Determine analysis type with enhanced logic
    const analysisType = this.determineAnalysisType(
      technical.length,
      fundamental.length,
      companiesAnalysis.discussionTopics
    );

    // Calculate confidence with enhanced factors
    const confidence = this.calculateConfidence({
      technical: technical.length,
      fundamental: fundamental.length,
      sectors: sectors.length,
      hasExplicitCompany: companiesAnalysis.hasExplicitMention,
      isContinuedDiscussion: companiesAnalysis.continuedDiscussion,
      messageLength: text.split(/\s+/).length,
      hasNumbers: /\d+/.test(text),
```

```typescript
      isQuestion: text.includes('?'),
      isSaudiMarket,
      companyConfidence: companiesAnalysis.companies[0]?.confidence || 0
    });

    return {
      confidence,
      companies: companiesAnalysis.companies,
      technicalIndicators: technical,
      fundamentalFactors: fundamental,
      sectors,
      riskLevel: this.assessRiskLevel(text),
      sentiment,
      analysisType,
      marketContext: {
        isSaudiMarket,
        mentionedMarket: isSaudiMarket ? 'Tadawul' : undefined
      },
      companyContext: {
        hasExplicitMention: companiesAnalysis.hasExplicitMention,
        continuedDiscussion: companiesAnalysis.continuedDiscussion,
        discussionTopics: companiesAnalysis.discussionTopics
      }
    };
  }

  private static analyzeCompanyContext(
    text: string,
    currentCompany?: CompanyContext
  ): {
    companies: CompanyMatch[];
    hasExplicitMention: boolean;
    continuedDiscussion: boolean;
    discussionTopics: string[];
  } {
    const companies: CompanyMatch[] = [];
    const discussionTopics: string[] = [];
    let hasExplicitMention = false;
    let continuedDiscussion = false;
    // Safely convert text to lowercase
    const lowerText = (text || '').toLowerCase();
    // First check for explicit company mentions
    SAUDI_SYMBOLS.forEach(stock => {
      const matches: MatchConfidence = {
        symbolMatch: Boolean(stock.tickerSymbol && lowerText.includes(stock.tickerSymbol.toLowerCase())),
        nameMatch: Boolean(stock.name && lowerText.includes(stock.name.toLowerCase())),
        arabicMatch: Boolean(stock.name_ar && lowerText.includes(stock.name_ar.toLowerCase())),
        partialMatch: false,
        contextualRelevance: false
      };
      // Check for partial matches if no exact match found
      if (!matches.symbolMatch && !matches.nameMatch && !matches.arabicMatch) {
        matches.partialMatch = this.checkPartialMatch(lowerText, stock);
      }
      // Check contextual relevance
      matches.contextualRelevance = this.checkContextualRelevance(lowerText, stock);
      // Calculate confidence score
      const confidence = this.calculateMatchConfidence(matches);
      if (confidence > 0.3) { // Threshold for considering a match
        companies.push({
          symbol: stock.symbol,
          name: stock.name,
          sector: stock.sector,
          isContinuedDiscussion: false,
          confidence
        });
        hasExplicitMention = true;
      }
    });
    // If no explicit mentions but we have a current company
    if (companies.length === 0 && currentCompany) {
      const hasRelatedTerms = Array.from(this.COMPANY_RELATED_TERMS)
        .some(term => lowerText.includes(term.toLowerCase()));

      if (hasRelatedTerms) {
        companies.push({
          symbol: currentCompany.symbol,
          name: currentCompany.name,
          sector: currentCompany.sector,
          isContinuedDiscussion: true,
          confidence: 0.4 // Default confidence for continued discussion
        });
        continuedDiscussion = true;
      }
    }
    // Extract discussion topics
```

```typescript
    if (companies.length > 0) {
      discussionTopics.push(
        ...this.extractTechnicalIndicators(text),
        ...this.extractFundamentalFactors(text),
        ...Array.from(this.COMPANY_RELATED_TERMS)
          .filter(term => lowerText.includes(term.toLowerCase()))
      );
    }
    return {
      companies,
      hasExplicitMention,
      continuedDiscussion,
      discussionTopics: Array.from(new Set(discussionTopics))
    };
  }

  private static checkPartialMatch(text: string, stock: any): boolean {
    const words = text.split(/\s+/);
    const stockNameWords = stock.name?.toLowerCase().split(/\s+/) || [];

    return stockNameWords.some((word: string) =>
      word.length > 3 && words.some((textWord: string) =>
        textWord.includes(word) || word.includes(textWord)
      )
    );
  }




  private static checkContextualRelevance(text: string, stock: any): boolean {
    const sectorTerms = this.getSectorTerms(stock.sector);
    return sectorTerms.some((term: string) => text.includes(term.toLowerCase()));
  }

  private static getSectorTerms(sector?: string): string[] {
    // Add sector-specific terms mapping
    const sectorTermsMap: { [key: string]: string[] } = {
      'Technology': ['tech', 'software', 'digital', 'IT'],
      'Banking': ['bank', 'finance', 'loan', 'deposit'],
      // Add more sectors as needed
    };

    return sector ? (sectorTermsMap[sector] || []) : [];
  }

  private static checkSaudiMarketContext(text: string): boolean {
    return Array.from(this.SAUDI_MARKET_TERMS)
      .some(term => text.includes(term.toLowerCase()));
  }

  private static extractTechnicalIndicators(text: string): string[] {
    return Array.from(this.TECHNICAL_INDICATORS)
      .filter(indicator => text.includes(indicator.toLowerCase()));
  }

  private static extractFundamentalFactors(text: string): string[] {
    return Array.from(this.FUNDAMENTAL_FACTORS)
      .filter(factor => text.includes(factor.toLowerCase()));
  }

  private static extractSectors(text: string): string[] {
    return Array.from(this.SECTORS)
      .filter(sector => text.includes(sector.toLowerCase()));
  }

  private static analyzeSentimentWithContext(
    text: string,
    company: { symbol: string; name: string }
  ): 'bullish' | 'bearish' | 'neutral' {
    const words = text.split(/\s+/);
    const companyIndex = words.findIndex(word =>
      word.includes(company.name.toLowerCase()) ||
      word.includes(company.symbol.toLowerCase())
    );

    if (companyIndex !== -1) {
      const start = Math.max(0, companyIndex - 5);
      const end = Math.min(words.length, companyIndex + 6);
      const contextText = words.slice(start, end).join(' ');
      return this.determineSentiment(contextText);
    }

    return this.determineSentiment(text);
  }
```

```typescript
  private static determineSentiment(text: string): 'bullish' | 'bearish' | 'neutral' {
    let score = 0;

    for (const term of this.SENTIMENT_TERMS.bullish) {
      if (text.includes(term)) score += 1;
    }
    for (const term of this.SENTIMENT_TERMS.bearish) {
      if (text.includes(term)) score -= 1;
    }

    if (score > 1) return 'bullish';
    if (score < -1) return 'bearish';
    return 'neutral';
  }

  private static assessRiskLevel(text: string): 'low' | 'medium' | 'high' {
    let score = 0;

    for (const term of this.RISK_TERMS.high) {
      if (text.includes(term)) score += 2;
    }
    for (const term of this.RISK_TERMS.low) {
      if (text.includes(term)) score -= 2;
    }

    if (text.includes('hedge') || text.includes('diversif')) score -= 1;
    if (text.includes('leverage') || text.includes('margin')) score += 1;

    if (score > 2) return 'high';
    if (score < -1) return 'low';
    return 'medium';
  }

  private static determineAnalysisType(
    technicalCount: number,
    fundamentalCount: number,
    topics: string[]
  ): 'technical' | 'fundamental' | 'mixed' | 'general' {
    const hasCompanySpecific = topics.some(topic =>
      this.COMPANY_RELATED_TERMS.has(topic)
    );

    if (technicalCount > 0 && fundamentalCount > 0) {
      return 'mixed';
    }
    if (technicalCount > 0) {
      return 'technical';
    }
    if (fundamentalCount > 0 || hasCompanySpecific) {
      return 'fundamental';
    }
    return 'general';
  }

  private static calculateConfidence(factors: {
    technical: number;
    fundamental: number;
    sectors: number;
    hasExplicitCompany: boolean;
    isContinuedDiscussion: boolean;
    messageLength: number;
    hasNumbers: boolean;
    isQuestion: boolean;
    isSaudiMarket: boolean;
    companyConfidence: number;
  }): number {
    let confidence = 0.5; // Base confidence

    // Company context factors
    if (factors.hasExplicitCompany) confidence += 0.2;
    if (factors.isContinuedDiscussion) confidence += 0.1;
    confidence += factors.companyConfidence * 0.2;

    // Analysis depth factors
    if (factors.technical > 0) confidence += 0.1;
    if (factors.fundamental > 0) confidence += 0.1;
    if (factors.sectors > 0) confidence += 0.05;

    // Message quality factors
    if (factors.hasNumbers) confidence += 0.05;
    if (factors.isQuestion) confidence += 0.05;
    if (factors.isSaudiMarket) confidence += 0.1;

    // Penalize very short messages
    if (factors.messageLength < 3) confidence -= 0.2;
```

```
    // Ensure confidence stays within bounds
    return Math.max(0.1, Math.min(0.95, confidence));
  }
}
```

```typescript
// app/lib/pdf-utils.ts
import { encode } from 'base64-js';

export async function convertPDFToBase64(file: File): Promise<string> {
  return new Promise((resolve, reject) => {
    const reader = new FileReader();
    reader.onload = () => {
      const arrayBuffer = reader.result as ArrayBuffer;
      const uint8Array = new Uint8Array(arrayBuffer);
      const base64String = encode(uint8Array);
      resolve(base64String);
    };
    reader.onerror = reject;
    reader.readAsArrayBuffer(file);
  });
}

export interface PDFMessage {
  role: 'user';
  content: {
    type: 'document' | 'text';
    source?: {
      type: 'base64';
      media_type: 'application/pdf';
      data: string;
    };
    text?: string;
  }[];
}

export function createPDFMessage(base64Data: string, query: string): PDFMessage {
  return {
    role: 'user',
    content: [
      {
        type: 'document',
        source: {
          type: 'base64',
          media_type: 'application/pdf',
          data: base64Data
        }
      },
      {
        type: 'text',
        text: query
      }
    ]
  };
}
```

```typescript
// lib/utils.ts
import { type ClassValue, clsx } from "clsx";
import { twMerge } from "tailwind-merge";

export function cn(...inputs: ClassValue[]) {
  return twMerge(clsx(inputs));
}

export function formatNumber(
  value: number | undefined | null,
  currency?: string,
  compact = false
): string {
  if (value === undefined || value === null || isNaN(value)) {
    return "N/A";
  }

  try {
    if (currency === "%") {
      return `${value.toFixed(2)}%`;
    }

    if (compact) {
      const units = ["", "K", "M", "B", "T"];
      let unitIndex = 0;
      let scaledValue = value;

      while (scaledValue >= 1000 && unitIndex < units.length - 1) {
        scaledValue /= 1000;
        unitIndex += 1;
      }

      return `${currency ? currency + " " : ""}${scaledValue.toFixed(1)}${units[unitIndex]}`;
    }

    if (currency) {
      return `${currency} ${value.toLocaleString(undefined, {
        minimumFractionDigits: 2,
        maximumFractionDigits: 2,
      })}`;
    }

    return value.toLocaleString(undefined, {
      minimumFractionDigits: 2,
      maximumFractionDigits: 2,
    });
  } catch (error) {
    // Fallback formatting if Intl is not available
    return `${currency ? currency + " " : ""}${value.toFixed(2)}`;
  }
}
```

```
// // lib/api/market.ts

// const TWELVE_DATA_API_KEY = process.env.NEXT_PUBLIC_TWELVE_DATA_API_KEY;
// const BASE_URL = 'https://api.twelvedata.com';

// export const DEFAULT_SYMBOLS = [
//   'AAPL', 'AMZN', 'MSFT', 'NVDA', 'TSLA',
//   'GOOGL', 'V', 'TSM', 'BRK.A', 'WMT'
// ];

// export async function getQuotes(symbols: string[] = DEFAULT_SYMBOLS) {
//   try {
//     const response = await fetch(
//       `${BASE_URL}/quote?symbol=${symbols.join(',')}&apikey=${TWELVE_DATA_API_KEY}`
//     );
//     const data = await response.json();
//     return Array.isArray(data) ? data : [data];
//   } catch (error) {
//     console.error('Error fetching quotes:', error);
//     return [];
//   }
// }

// export async function getTimeSeries(
//   symbol: string,
//   interval: string = '1day',
//   outputsize: number = 252 // One year of trading days
// ) {
//   try {
//     const response = await fetch(
//       `${BASE_URL}/time_series?symbol=${symbol}&interval=${interval}&outputsize=${outputsize}&apikey=${TWELVE_DATA_API_KEY}`
//     );
//     const data = await response.json();
//     return data.values || [];
//   } catch (error) {
//     console.error('Error fetching time series:', error);
//     return [];
//   }
// }
```

```typescript
// // services/saudi-market.ts

// import { API_KEY, API_URL } from '@/app/config/market';

// export interface SaudiSymbol {
//   symbol: string;
//   name: string;
//   currency: string;
//   exchange: string;
//   mic_code: string;
//   type: string;
//   logo_url?: string; // Will be added from another API or local storage
// }

// export async function fetchSaudiSymbols(): Promise<SaudiSymbol[]> {
//   try {
//     // Fetch all stocks with Saudi Arabia filter
//     const response = await fetch(
//       `${API_URL}/stocks?country=Saudi Arabia&apikey=${API_KEY}`
//     );

//     if (!response.ok) {
//       throw new Error('Failed to fetch Saudi stocks');
//     }

//     const data = await response.json();
//     const stocks = Array.isArray(data) ? data : data.data || [];

//     // Map and add logo URLs
//     // We'll need to either:
//     // 1. Use another API to get logos
//     // 2. Store logos locally
//     // 3. Generate placeholder logos
//     return stocks.map(stock => ({
//       ...stock,
//       logo_url: `/logos/${stock.symbol}.png` // This path will need to be adjusted
//     }));

//   } catch (error) {
//     console.error('Error fetching Saudi symbols:', error);
//     throw error;
//   }
// }

// // Function to get logo URL for a symbol
// export async function getCompanyLogo(symbol: string): Promise<string> {
//   // We can implement this in several ways:
//   // 1. Use a financial API that provides logos
//   // 2. Scrape from Tadawul website
//   // 3. Store logos locally
//   // 4. Generate placeholder logos

//   // For now, return a placeholder
//   return `https://ui-avatars.com/api/?name=${encodeURIComponent(symbol)}&background=random`;
// }

// // Function to get company details
// export interface CompanyDetails {
//   symbol: string;
//   name: string;
//   sector: string;
//   industry: string;
//   website?: string;
//   description?: string;
//   market_cap?: number;
//   employees?: number;
//   ceo?: string;
//   headquarters?: string;
//   founded?: string;
// }

// export async function getCompanyDetails(symbol: string): Promise<CompanyDetails | null> {
//   try {
//     // This would need to be implemented with actual data source
//     // Could be from Tadawul API or another financial data provider
//     return null;
//   } catch (error) {
//     console.error(`Error fetching details for ${symbol}:`, error);
//     return null;
//   }
// }
```

```typescript
import { API_KEY, API_URL, SAUDI_SYMBOLS } from '@/app/config/market';

interface FundamentalData {
  symbol: string;
  name: string;
  sector: string;
  valuation_metrics: {
    market_capitalization?: number;
    enterprise_value?: number;
    trailing_pe?: number;
    forward_pe?: number;
    peg_ratio?: number;
    price_to_sales_ttm?: number;
    price_to_book_mrq?: number;
    enterprise_to_revenue?: number;
    enterprise_to_ebitda?: number;
  };
  financials: {
    gross_margin?: number;
    profit_margin?: number;
    operating_margin?: number;
    return_on_assets_ttm?: number;
    return_on_equity_ttm?: number;
    income_statement?: {
      revenue_ttm?: number;
      revenue_per_share_ttm?: number;
      quarterly_revenue_growth?: number;
      gross_profit_ttm?: number;
      ebitda?: number;
      net_income_to_common_ttm?: number;
      diluted_eps_ttm?: number;
      quarterly_earnings_growth_yoy?: number;
    };
    balance_sheet?: {
      total_cash_mrq?: number;
      total_cash_per_share_mrq?: number;
      total_debt_mrq?: number;
      total_debt_to_equity_mrq?: number;
      current_ratio_mrq?: number;
      book_value_per_share_mrq?: number;
    };
    cash_flow?: {
      operating_cash_flow_ttm?: number;
      levered_free_cash_flow_ttm?: number;
    };
  };
  dividends?: {
    forward_annual_dividend_rate?: number;
    forward_annual_dividend_yield?: number;
    trailing_annual_dividend_rate?: number;
    trailing_annual_dividend_yield?: number;
    five_year_average_dividend_yield?: number;
    payout_ratio?: number;
    dividend_date?: string;
    ex_dividend_date?: string;
  };
  stock_statistics?: {
    shares_outstanding?: number;
    float_shares?: number;
    avg_volume_10d?: number;
    avg_volume_90d?: number;
    short_ratio?: number;
    percent_held_by_insiders?: number;
    percent_held_by_institutions?: number;
  };
}

interface BalanceSheetResponse {
  meta: {
    symbol: string;
    name: string;
    currency: string;
    exchange: string;
    mic_code: string;
    exchange_timezone: string;
    period: string;
  };
  balance_sheet: Array<{
    fiscal_date: string;
    assets: {
      current_assets: {
        cash: number;
        cash_equivalents: number;
        cash_and_cash_equivalents: number;
        other_short_term_investments: number;
        accounts_receivable: number;
```

```typescript
        other_receivables: number;
        inventory: number;
        prepaid_assets: number | null;
        restricted_cash: number | null;
        assets_held_for_sale: number | null;
        hedging_assets: number | null;
        other_current_assets: number;
        total_current_assets: number;
      };
      non_current_assets: {
        properties: number;
        land_and_improvements: number;
        machinery_furniture_equipment: number;
        construction_in_progress: number | null;
        leases: number;
        accumulated_depreciation: number;
        goodwill: number | null;
        investment_properties: number | null;
        financial_assets: number | null;
        intangible_assets: number | null;
        investments_and_advances: number;
        other_non_current_assets: number;
        total_non_current_assets: number;
      };
      total_assets: number;
    };
    liabilities: {
      current_liabilities: {
        accounts_payable: number;
        accrued_expenses: number | null;
        short_term_debt: number;
        deferred_revenue: number;
        tax_payable: number | null;
        pensions: number | null;
        other_current_liabilities: number;
        total_current_liabilities: number;
      };
      non_current_liabilities: {
        long_term_provisions: number | null;
        long_term_debt: number;
        provision_for_risks_and_charges: number;
        deferred_liabilities: number | null;
        derivative_product_liabilities: number | null;
        other_non_current_liabilities: number;
        total_non_current_liabilities: number;
      };
      total_liabilities: number;
    };
    shareholders_equity: {
      common_stock: number;
      retained_earnings: number;
      other_shareholders_equity: number;
      total_shareholders_equity: number;
      additional_paid_in_capital: number | null;
      treasury_stock: number | null;
      minority_interest: number | null;
    };
  }>;
}

interface CashFlowResponse {
  meta: {
    symbol: string;
    name: string;
    currency: string;
    exchange: string;
    mic_code: string;
    exchange_timezone: string;
    period: string;
  };
  cash_flow: Array<{
    fiscal_date: string;
    quarter?: number;
    operating_activities: {
      net_income: number;
      depreciation: number;
      deferred_taxes: number;
      stock_based_compensation: number;
      other_non_cash_items: number;
      accounts_receivable: number;
      accounts_payable: number;
      other_assets_liabilities: number;
      operating_cash_flow: number;
    };
    investing_activities: {
      capital_expenditures: number;
```

```typescript
      net_intangibles: number | null;
      net_acquisitions: number | null;
      purchase_of_investments: number;
      sale_of_investments: number;
      other_investing_activity: number;
      investing_cash_flow: number;
    };
    financing_activities: {
      long_term_debt_issuance: number | null;
      long_term_debt_payments: number;
      short_term_debt_issuance: number;
      common_stock_issuance: number | null;
      common_stock_repurchase: number;
      common_dividends: number;
      other_financing_charges: number;
      financing_cash_flow: number;
    };
    end_cash_position: number;
    income_tax_paid: number;
    interest_paid: number;
    free_cash_flow: number;
  }>;
}


interface IncomeStatementResponse {
  meta: {
    symbol: string;
    name: string;
    currency: string;
    exchange: string;
    mic_code: string;
    exchange_timezone: string;
    period: string;
  };
  income_statement: Array<{
    fiscal_date: string;
    quarter?: number;
    year?: number;
    sales: number;
    cost_of_goods: number;
    gross_profit: number;
    operating_expense: {
      research_and_development: number;
      selling_general_and_administrative: number;
      other_operating_expenses: number | null;
    };
    operating_income: number;
    non_operating_interest: {
      income: number;
      expense: number;
    };
    other_income_expense: number;
    pretax_income: number;
    income_tax: number;
    net_income: number;
    eps_basic: number;
    eps_diluted: number;
    basic_shares_outstanding: number;
    diluted_shares_outstanding: number;
    ebitda: number;
    net_income_continuous_operations: number | null;
    minority_interests: number | null;
    preferred_stock_dividends: number | null;
  }>;
}

export class SaudiFundamentals {
  private static readonly CACHE_DURATION = 300000; // 5 minutes
  private static cache: Map<string, { data: FundamentalData; timestamp: number }> = new Map();

  private static async fetchWithCache(url: string): Promise<any> {
    const cached = this.cache.get(url);
    if (cached && Date.now() - cached.timestamp < this.CACHE_DURATION) {
      return cached.data;
    }

    try {
      const response = await fetch(url);
      if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`);
      }
      const data = await response.json();
      this.cache.set(url, { data, timestamp: Date.now() });
      return data;
```

```typescript
    } catch (error) {
      console.error('Error fetching data:', error);
      throw error;
    }
  }

  static async getFundamental(symbol: string): Promise<FundamentalData | null> {
    try {
      const stockInfo = SAUDI_SYMBOLS.find(s => s.symbol === symbol);
      if (!stockInfo) return null;

      const formattedSymbol = stockInfo.tickerSymbol;

      // Fetch statistics from Twelve Data API
      const url = `${API_URL}/statistics?symbol=${formattedSymbol}&apikey=${API_KEY}`;
      const response = await this.fetchWithCache(url);

      if (!response || !response.statistics) {
        throw new Error('Invalid response from statistics endpoint');
      }

      const stats = response.statistics;

      // Transform API response into our FundamentalData structure
      const fundamentalData: FundamentalData = {
        symbol: stockInfo.symbol,
        name: stockInfo.name,
        sector: stockInfo.sector,
        valuation_metrics: {
          market_capitalization: stats.valuations_metrics?.market_capitalization,
          enterprise_value: stats.valuations_metrics?.enterprise_value,
          trailing_pe: stats.valuations_metrics?.trailing_pe,
          forward_pe: stats.valuations_metrics?.forward_pe,
          peg_ratio: stats.valuations_metrics?.peg_ratio,
          price_to_sales_ttm: stats.valuations_metrics?.price_to_sales_ttm,
          price_to_book_mrq: stats.valuations_metrics?.price_to_book_mrq,
          enterprise_to_revenue: stats.valuations_metrics?.enterprise_to_revenue,
          enterprise_to_ebitda: stats.valuations_metrics?.enterprise_to_ebitda
        },
        financials: {
          gross_margin: stats.financials?.gross_margin,
          profit_margin: stats.financials?.profit_margin,
          operating_margin: stats.financials?.operating_margin,
          return_on_assets_ttm: stats.financials?.return_on_assets_ttm,
          return_on_equity_ttm: stats.financials?.return_on_equity_ttm,
          income_statement: {
            revenue_ttm: stats.financials?.income_statement?.revenue_ttm,
            revenue_per_share_ttm: stats.financials?.income_statement?.revenue_per_share_ttm,
            quarterly_revenue_growth: stats.financials?.income_statement?.quarterly_revenue_growth,
            gross_profit_ttm: stats.financials?.income_statement?.gross_profit_ttm,
            ebitda: stats.financials?.income_statement?.ebitda,
            net_income_to_common_ttm: stats.financials?.income_statement?.net_income_to_common_ttm,
            diluted_eps_ttm: stats.financials?.income_statement?.diluted_eps_ttm,
            quarterly_earnings_growth_yoy: stats.financials?.income_statement?.quarterly_earnings_growth_yoy
          },
          balance_sheet: {
            total_cash_mrq: stats.financials?.balance_sheet?.total_cash_mrq,
            total_cash_per_share_mrq: stats.financials?.balance_sheet?.total_cash_per_share_mrq,
            total_debt_mrq: stats.financials?.balance_sheet?.total_debt_mrq,
            total_debt_to_equity_mrq: stats.financials?.balance_sheet?.total_debt_to_equity_mrq,
            current_ratio_mrq: stats.financials?.balance_sheet?.current_ratio_mrq,
            book_value_per_share_mrq: stats.financials?.balance_sheet?.book_value_per_share_mrq
          },
          cash_flow: {
            operating_cash_flow_ttm: stats.financials?.cash_flow?.operating_cash_flow_ttm,
            levered_free_cash_flow_ttm: stats.financials?.cash_flow?.levered_free_cash_flow_ttm
          }
        },
        dividends: {
          forward_annual_dividend_rate: stats.dividends_and_splits?.forward_annual_dividend_rate,
          forward_annual_dividend_yield: stats.dividends_and_splits?.forward_annual_dividend_yield,
          trailing_annual_dividend_rate: stats.dividends_and_splits?.trailing_annual_dividend_rate,
          trailing_annual_dividend_yield: stats.dividends_and_splits?.trailing_annual_dividend_yield,
          five_year_average_dividend_yield: stats.dividends_and_splits?.["5_year_average_dividend_yield"],
          payout_ratio: stats.dividends_and_splits?.payout_ratio,
          dividend_date: stats.dividends_and_splits?.dividend_date,
          ex_dividend_date: stats.dividends_and_splits?.ex_dividend_date
        },
        stock_statistics: {
          shares_outstanding: stats.stock_statistics?.shares_outstanding,
          float_shares: stats.stock_statistics?.float_shares,
          avg_volume_10d: stats.stock_statistics?.avg_10_volume,
          avg_volume_90d: stats.stock_statistics?.avg_90_volume,
          short_ratio: stats.stock_statistics?.short_ratio,
          percent_held_by_insiders: stats.stock_statistics?.percent_held_by_insiders,
          percent_held_by_institutions: stats.stock_statistics?.percent_held_by_institutions
```

```
      }
    };

    return fundamentalData;

  } catch (error) {
    console.error(`Error fetching fundamental data for ${symbol}:`, error);
    return null;
  }
}

static async getSectorAverages(sector: string): Promise<{
  avg_pe?: number;
  avg_market_cap?: number;
  avg_dividend_yield?: number;
  avg_revenue_growth?: number;
  best_performer?: { symbol: string; name: string; growth: number };
  worst_performer?: { symbol: string; name: string; growth: number };
}> {
  try {
    const sectorStocks = SAUDI_SYMBOLS.filter(stock => stock.sector === sector);
    const fundamentalsPromises = sectorStocks.map(stock => this.getFundamental(stock.symbol));
    const results = await Promise.all(fundamentalsPromises);

    const validResults = results.filter((result): result is FundamentalData => result !== null);

    if (validResults.length === 0) {
      return {};
    }

    const sectorData = {
      avg_pe: this.calculateAverage(validResults.map(r => r.valuation_metrics?.trailing_pe)),
      avg_market_cap: this.calculateAverage(validResults.map(r => r.valuation_metrics?.market_capitalization)
),
      avg_dividend_yield: this.calculateAverage(validResults.map(r => r.dividends?.trailing_annual_dividend_y
ield)),
      avg_revenue_growth: this.calculateAverage(validResults.map(r => r.financials?.income_statement?.quarter
ly_revenue_growth))
    };

    // Find best and worst performers based on revenue growth
    const performanceData = validResults
      .map(result => ({
        symbol: result.symbol,
        name: result.name,
        growth: result.financials?.income_statement?.quarterly_revenue_growth || 0
      }))
      .sort((a, b) => b.growth - a.growth);

    return {
      ...sectorData,
      best_performer: performanceData[0],
      worst_performer: performanceData[performanceData.length - 1]
    };

  } catch (error) {
    console.error(`Error calculating sector averages for ${sector}:`, error);
    return {};
  }
}

private static calculateAverage(numbers: (number | undefined)[]): number | undefined {
  const validNumbers = numbers.filter((n): n is number => typeof n === 'number' && !isNaN(n));
  if (validNumbers.length === 0) return undefined;
  return validNumbers.reduce((sum, n) => sum + n, 0) / validNumbers.length;
}
}
```

```typescript
import Anthropic from "@anthropic-ai/sdk";
import { z } from "zod";
import { SaudiMarketHandler } from "@/app/market/saudiMarketHandler";
import { SaudiFundamentals } from "@/app/lib/services/saudiFundamentals";

const anthropic = new Anthropic({
  apiKey: process.env.ANTHROPIC_API_KEY!,
});

// Core response schema
const responseSchema = z.object({
  response: z.string(),
  thinking: z.string(),
  user_mood: z.enum(["positive", "neutral", "negative", "curious", "frustrated", "confused"]),
  suggested_questions: z.array(z.string()),
  financial_context: z.object({
    companies: z.array(z.object({
      symbol: z.string(),
      name: z.string(),
      sector: z.string().optional(),
      price: z.number().optional(),
      change: z.number().optional(),
      analysis: z.any().optional()
    })).default([]),
    market_sentiment: z.enum(["bullish", "bearish", "neutral"]).default("neutral"),
    risk_level: z.enum(["low", "medium", "high"]).default("medium"),
    confidence_score: z.number().default(0.5),
    key_insights: z.array(z.string()).default([]),
    analysis_details: z.any().optional()
  }),
  debug: z.object({
    context_used: z.boolean().default(false),
    market_data_used: z.boolean().default(false),
    analysis_quality: z.string().default("limited")
  })
});

export async function POST(req: Request) {
  try {
    const { messages, documentContext } = await req.json();
    const latestMessage = messages[messages.length - 1].content;

    // Step 1: Market Analysis
    const marketContext = await analyzeMarket(latestMessage);

    // Step 2: Get Claude's Response
    const enhancedPrompt = buildPrompt(latestMessage, marketContext, documentContext);
    const claudeResponse = await anthropic.messages.create({
      model: "claude-3-5-sonnet-20241022",
      max_tokens: 4096,
      messages: [{ role: 'user', content: enhancedPrompt }],
    });

    // Step 3: Build Enhanced Response
    const response = await buildResponse(claudeResponse.content[0].text, marketContext);

    // Validate and return
    const validatedResponse = responseSchema.parse(response);
    return new Response(JSON.stringify(validatedResponse), {
      status: 200,
      headers: { "Content-Type": "application/json" }
    });

  } catch (error) {
    console.error("Error in chat handler:", error);
    return handleError(error);
  }
}

async function analyzeMarket(message: string) {
  // Get market data
  const marketAnalysis = await SaudiMarketHandler.analyzeMentionedCompanies(message);

  // Get fundamentals for mentioned companies
  const fundamentals = await Promise.all(
    marketAnalysis.companies.map(company =>
      SaudiFundamentals.getFundamental(company.symbol)
    )
  );

  // Analyze sentiment
  const sentiment = analyzeSentiment(message, marketAnalysis.companies);

  return {
    companies: marketAnalysis.companies,
    sectors: Array.from(marketAnalysis.sectors),
```

```typescript
    fundamentals: fundamentals.filter(f => f !== null),
    sentiment,
    timestamp: new Date().toISOString()
  };
}

function buildPrompt(message: string, marketContext: any, documentContext?: any) {
  const contextParts = [
    "You are a highly knowledgeable Saudi market financial advisor and analyst.",
    "Provide clear, concise analysis and recommendations.",

    // Market Context
    marketContext.companies.length > 0 && `
    Analyzing companies: ${marketContext.companies.map(c => c.name).join(', ')}
    Sectors involved: ${marketContext.sectors.join(', ')}
    Market sentiment: ${marketContext.sentiment.sentiment}
    `,

    // Document Context (if available)
    documentContext && `Reference materials available: ${documentContext}`,

    // User Query
    `User query: ${message}`
  ].filter(Boolean);

  return contextParts.join("\n\n");
}

async function buildResponse(claudeResponse: string, marketContext: any) {
  const suggestedQuestions = generateQuestions(marketContext);
  const insights = generateInsights(marketContext);

  return {
    response: claudeResponse,
    thinking: "Analyzing Saudi market data",
    user_mood: determineUserMood(claudeResponse),
    suggested_questions: suggestedQuestions,
    financial_context: {
      companies: marketContext.companies.map(enrichCompanyData),
      market_sentiment: marketContext.sentiment.sentiment,
      risk_level: assessRiskLevel(marketContext),
      confidence_score: calculateConfidence(marketContext),
      key_insights: insights,
      analysis_details: {
        fundamentals: marketContext.fundamentals,
        sentiment: marketContext.sentiment
      }
    },
    debug: {
      context_used: true,
      market_data_used: marketContext.companies.length > 0,
      analysis_quality: determineQuality(marketContext)
    }
  };
}

// Helper Functions
function analyzeSentiment(message: string, companies: any[]) {
  const bullishTerms = ['up', 'gain', 'grow', 'positive', 'bullish'];
  const bearishTerms = ['down', 'loss', 'decline', 'negative', 'bearish'];

  let score = 0;
  const text = message.toLowerCase();

  bullishTerms.forEach(term => text.includes(term) && score++);
  bearishTerms.forEach(term => text.includes(term) && score--);

  if (companies.length > 0) {
    const avgChange = companies.reduce((sum, c) => sum + (c.percent_change || 0), 0) / companies.length;
    score += Math.sign(avgChange);
  }

  return {
    sentiment: score > 0 ? 'bullish' : score < 0 ? 'bearish' : 'neutral',
    confidence: Math.min(Math.abs(score) / 3, 0.9)
  };
}

function enrichCompanyData(company: any) {
  return {
    symbol: company.symbol,
    name: company.name,
    sector: company.sector,
    price: company.price,
    change: company.percent_change,
    analysis: {
```

```typescript
        momentum: company.percent_change > 0 ? 'positive' : 'negative',
        activity: company.volume > (company.avg_volume || 0) ? 'high' : 'normal'
      }
    };
  }
}

function generateQuestions(context: any): string[] {
  const questions = new Set<string>();

  if (context.companies.length > 0) {
    const company = context.companies[0];
    questions.add(`What's the outlook for ${company.name}?`);
  }

  if (context.sectors.length > 0) {
    questions.add(`How is the ${context.sectors[0]} sector performing?`);
  }

  questions.add("What are the key market trends today?");

  return Array.from(questions).slice(0, 3);
}

function generateInsights(context: any): string[] {
  const insights = [];

  if (context.companies.length > 0) {
    const movers = context.companies.filter(c => Math.abs(c.percent_change) > 2);
    if (movers.length > 0) {
      insights.push(
        `Significant price movements in: ${movers.map(c => c.name).join(', ')}`
      );
    }
  }

  if (context.fundamentals?.length > 0) {
    const strongFundamentals = context.fundamentals.filter(f =>
      f?.financials?.profit_margin > 0.15
    );
    if (strongFundamentals.length > 0) {
      insights.push("Companies showing strong profitability metrics");
    }
  }

  return insights;
}

function determineUserMood(response: string): "positive" | "neutral" | "negative" | "curious" | "frustrated" |
"confused" {
  const text = response.toLowerCase();
  if (text.includes('help') || text.includes('?')) return 'curious';
  if (text.includes('thank') || text.includes('great')) return 'positive';
  if (text.includes('error') || text.includes('wrong')) return 'frustrated';
  if (text.includes('not sure')) return 'confused';
  if (text.includes('bad')) return 'negative';
  return 'neutral';
}

function assessRiskLevel(context: any): "low" | "medium" | "high" {
  const volatility = context.companies.some(c => Math.abs(c.percent_change) > 5);
  const fundamentalRisk = context.fundamentals.some(f =>
    f?.valuation_metrics?.trailing_pe > 30
  );

  if (volatility && fundamentalRisk) return "high";
  if (volatility || fundamentalRisk) return "medium";
  return "low";
}

function calculateConfidence(context: any): number {
  let score = 0.5;
  if (context.companies.length > 0) score += 0.2;
  if (context.fundamentals.length > 0) score += 0.2;
  if (context.sentiment.confidence > 0.7) score += 0.1;
  return Math.min(score, 0.95);
}

function determineQuality(context: any): string {
  const hasCompanies = context.companies.length > 0;
  const hasFundamentals = context.fundamentals.length > 0;

  if (hasCompanies && hasFundamentals) return 'comprehensive';
  if (hasCompanies || hasFundamentals) return 'good';
  return 'limited';
}
```

```ts
function handleError(error: unknown) {
  return new Response(
    JSON.stringify({
      response: "An error occurred while processing your request.",
      thinking: "Error occurred",
      user_mood: "frustrated",
      suggested_questions: [
        "Would you like to try a different query?",
        "Should we look at the overall market instead?",
        "Would you like to focus on a specific sector?"
      ],
      financial_context: {
        companies: [],
        market_sentiment: "neutral",
        risk_level: "medium",
        confidence_score: 0,
        key_insights: [],
      },
      debug: {
        context_used: false,
        market_data_used: false,
        analysis_quality: "error"
      }
    }),
    { status: 500 }
  );
}
```

```ts
// app/api/market/quotes/route.ts
import { NextResponse } from "next/server";
import { API_KEY, API_URL } from "@/app/config/market";
import { MarketDataError } from '@/app/lib/errors';

export async function GET(request: Request) {
  const { searchParams } = new URL(request.url);
  const symbols = searchParams.get("symbols");

  if (!symbols) {
    throw new MarketDataError(
      'VALIDATION_ERROR',
      'Symbol is required',
      'error',
      { timestamp: new Date().toISOString() }
    );
  } try {
    const response = await fetch(
      `${API_URL}/quote?symbol=${symbols}&apikey=${API_KEY}`
    );

    if (!response.ok) {
      throw new Error("Failed to fetch quotes");
    }

    const data = await response.json();
    const quotes: { [key: string]: any } = {};

    // Handle both single quote and multiple quotes responses
    if (Array.isArray(data)) {
      data.forEach((quote) => {
        quotes[quote.symbol] = {
          price: parseFloat(quote.close || quote.price || "0"),
          change: parseFloat(quote.change || "0"),
          changePercent: parseFloat(quote.percent_change || "0"),
          volume: parseInt(quote.volume || "0"),
        };
      });
    } else if (data.symbol) {
      quotes[data.symbol] = {
        price: parseFloat(data.close || data.price || "0"),
        change: parseFloat(data.change || "0"),
        changePercent: parseFloat(data.percent_change || "0"),
        volume: parseInt(data.volume || "0"),
      };
    }

    // If API fails to return data, generate sample data for demo
    if (Object.keys(quotes).length === 0) {
      symbols.split(",").forEach((symbol) => {
        quotes[symbol] = {
          price: Math.random() * 100 + 50,
          change: (Math.random() - 0.5) * 5,
          changePercent: (Math.random() - 0.5) * 10,
          volume: Math.floor(Math.random() * 1000000),
        };
      });
    }

    return NextResponse.json(quotes);
  } catch (error) {
    if (error instanceof MarketDataError) {
      return new Response(
        JSON.stringify({
          error: error.message,
          code: error.code,
          severity: error.severity,
          metadata: error.metadata
        }),
        {
          status: error.severity === 'critical' ? 500 : 400,
          headers: { 'Content-Type': 'application/json' }
        }
      );
    }
    // Return sample data in case of error
    const sampleData = symbols.split(",").reduce((acc: any, symbol) => {
      acc[symbol] = {
        price: Math.random() * 100 + 50,
        change: (Math.random() - 0.5) * 5,
        changePercent: (Math.random() - 0.5) * 10,
        volume: Math.floor(Math.random() * 1000000),
      };
      return acc;
    }, {});
```

```
    return NextResponse.json(sampleData);

  }

}
```

```
    return NextResponse.json(sampleData);

  }

}
```

```typescript
// app/api/market/stock/[symbol]/route.ts
import { NextResponse } from "next/server";
import { API_KEY, API_URL } from "@/app/config/market";

export async function GET(
  request: Request,
  { params }: { params: { symbol: string } }
) {
  try {
    // Fetch data from multiple endpoints
    const [quoteRes, statsRes, timeSeriesRes] = await Promise.all([
      fetch(`${API_URL}/quote?symbol=${params.symbol}&apikey=${API_KEY}`),
      fetch(`${API_URL}/statistics?symbol=${params.symbol}&apikey=${API_KEY}`),
      fetch(`${API_URL}/time_series?symbol=${params.symbol}&interval=1day&outputsize=90&apikey=${API_KEY}`)
    ]);

    // Check responses
    if (!quoteRes.ok || !statsRes.ok || !timeSeriesRes.ok) {
      throw new Error('Failed to fetch data from API');
    }

    const [quote, statistics, timeSeries] = await Promise.all([
      quoteRes.json(),
      statsRes.json(),
      timeSeriesRes.json()
    ]);

    // Extract statistics data for easier access
    const stats = statistics.statistics || {};
    const valuations = stats.valuations_metrics || {};
    const financials = stats.financials || {};
    const income = financials.income_statement || {};
    const returns = financials.returns || {};
    const growth = financials.growth || {};
    const dividends = stats.dividends_and_splits || {};

    // Transform and combine data
    const stockData = {
      quote: {
        symbol: params.symbol,
        name: quote.name || '',
        close: quote.close || quote.price?.toString() || '0',
        change: quote.change?.toString() || '0',
        percent_change: quote.percent_change?.toString() || '0',
        high: quote.high?.toString() || '0',
        low: quote.low?.toString() || '0',
        open: quote.open?.toString() || '0',
        volume: quote.volume?.toString() || '0'
      },
      stats: {
        // Profile
        market_cap: valuations.market_capitalization || 0,
        enterprise_value: valuations.enterprise_value || 0,
        shares_outstanding: stats.stock_statistics?.shares_outstanding || 0,
        revenue: income.revenue_ttm || 0,
        employees: stats.employees || 0,

        // Margins
        gross_margin: financials.gross_margin || 0,
        ebitda_margin: financials.ebitda_margin || 0,
        operating_margin: financials.operating_margin || 0,
        pretax_margin: financials.pretax_margin || 0,
        net_margin: financials.profit_margin || 0,
        fcf_margin: financials.fcf_margin || 0,

        // Returns (5Yr Avg)
        roa: returns.return_on_assets_5yr || 0,
        rota: returns.return_on_total_assets_5yr || 0,
        roe: returns.return_on_equity_5yr || 0,
        roce: returns.return_on_capital_employed_5yr || 0,
        roic: returns.return_on_invested_capital_5yr || 0,

        // Valuation (TTM)
        pe_ratio: valuations.trailing_pe || 0,
        pb_ratio: valuations.price_to_book_mrq || 0,
        ev_sales: valuations.enterprise_to_revenue || 0,
        ev_ebitda: valuations.enterprise_to_ebitda || 0,
        pfcf_ratio: valuations.price_to_fcf || 0,
        ev_gross_profit: valuations.enterprise_to_gross_profit || 0,

        // Valuation (NTM)
        price_target: valuations.analyst_target_price || 0,
        forward_pe: valuations.forward_pe || 0,
        peg_ratio: valuations.peg_ratio || 0,
        forward_ev_sales: valuations.forward_ev_to_revenue || 0,
        forward_ev_ebitda: valuations.forward_ev_to_ebitda || 0,
```

```typescript
        forward_pfcf: valuations.forward_price_to_fcf || 0,

        // Financial Health
        cash: financials.total_cash || 0,
        net_debt: financials.net_debt || 0,
        debt_to_equity: financials.debt_to_equity || 0,
        interest_coverage: financials.interest_coverage || 0,

        // Growth (CAGR)
        revenue_growth_3y: growth.revenue_growth_3y || 0,
        revenue_growth_5y: growth.revenue_growth_5y || 0,
        revenue_growth_10y: growth.revenue_growth_10y || 0,
        eps_growth_3y: growth.eps_growth_3y || 0,
        eps_growth_5y: growth.eps_growth_5y || 0,
        eps_growth_10y: growth.eps_growth_10y || 0,
        revenue_growth_fwd_2y: growth.revenue_growth_fwd_2y || 0,
        ebitda_growth_fwd_2y: growth.ebitda_growth_fwd_2y || 0,
        eps_growth_fwd_2y: growth.eps_growth_fwd_2y || 0,
        eps_growth_lt: growth.eps_growth_lt || 0,

        // Dividends
        dividend_yield: dividends.trailing_annual_dividend_yield || 0,
        payout_ratio: dividends.payout_ratio || 0,
        dps: dividends.trailing_annual_dividend_rate || 0,
        dps_growth_3y: dividends.dividend_growth_3y || 0,
        dps_growth_5y: dividends.dividend_growth_5y || 0,
        dps_growth_10y: dividends.dividend_growth_10y || 0,
        dps_growth_fwd_2y: dividends.dividend_growth_fwd_2y || 0,
      },
      timeSeries: {
        values: timeSeries.values?.map((item: any) => ({
          datetime: item.datetime,
          open: item.open?.toString() || '0',
          high: item.high?.toString() || '0',
          low: item.low?.toString() || '0',
          close: item.close?.toString() || '0',
          volume: item.volume?.toString() || '0'
        })) || []
      }
    };

    return NextResponse.json(stockData);
  } catch (error) {
    console.error('Error fetching stock data:', error);
    return NextResponse.json(
      { error: 'Failed to fetch stock data' },
      { status: 500 }
    );
  }
}
```

```typescript
// app/api/market/company/route.ts
import { NextResponse } from "next/server";
import { API_KEY, API_URL } from "@/app/config/market";

export async function GET(request: Request) {
  const { searchParams } = new URL(request.url);
  const symbol = searchParams.get("symbol");

  if (!symbol) {
    return NextResponse.json({ error: "Symbol is required" }, { status: 400 });
  }

  try {
    // Fetch fundamental data
    const [statsResponse, incomeResponse, balanceResponse] = await Promise.all([
      fetch(`${API_URL}/statistics?symbol=${symbol}&apikey=${API_KEY}`),
      fetch(`${API_URL}/income_statement?symbol=${symbol}&apikey=${API_KEY}`),
      fetch(`${API_URL}/balance_sheet?symbol=${symbol}&apikey=${API_KEY}`)
    ]);

    const [stats, income, balance] = await Promise.all([
      statsResponse.json(),
      incomeResponse.json(),
      balanceResponse.json()
    ]);

    // Calculate key metrics
    const latestIncome = income.income_statement?.[0] || {};
    const latestBalance = balance.balance_sheet?.[0] || {};

    const revenue = latestIncome.revenue || 0;
    const ebitda = latestIncome.ebitda || 0;
    const netIncome = latestIncome.net_income || 0;
    const totalAssets = latestBalance.total_assets || 0;
    const totalEquity = latestBalance.total_shareholders_equity || 0;

    const response = {
      // Market data
      market_cap: stats.market_cap || 0,
      enterprise_value: stats.enterprise_value || 0,
      shares_outstanding: stats.shares_outstanding || 0,

      // Valuation metrics
      pe_ratio: stats.pe_ratio || 0,
      pb_ratio: stats.price_to_book || 0,
      ev_to_sales: stats.enterprise_value_to_revenue || 0,
      ev_to_ebitda: stats.enterprise_value_to_ebitda || 0,

      // Financial metrics
      revenue: revenue,
      ebitda: ebitda,
      net_income: netIncome,
      total_assets: totalAssets,
      total_equity: totalEquity,

      // Operating metrics
      employees: stats.full_time_employees || 0,

      // Margins
      gross_margin: latestIncome.gross_margin || 0,
      ebitda_margin: ebitda / revenue || 0,
      operating_margin: latestIncome.operating_margin || 0,
      net_margin: netIncome / revenue || 0,
      fcf_margin: (latestIncome.operating_cash_flow - latestIncome.capital_expenditure) / revenue || 0,

      // Growth metrics
      revenue_growth: stats.quarterly_revenue_growth || 0,
      earnings_growth: stats.quarterly_earnings_growth || 0
    };

    return NextResponse.json(response);
  } catch (error) {
    console.error('Error fetching company data:', error);
    return NextResponse.json(
      { error: "Failed to fetch company data" },
      { status: 500 }
    );
  }
}
```

```typescript
// /app/market/saudiMarketHandler.ts

import { API_KEY, API_URL, SAUDI_SYMBOLS, SAUDI_MARKET } from '@/app/config/market';
import { MarketValidation } from './services/marketValidation';
import { MarketQuality } from './services/marketQuality';
import { MarketDataQuality } from './types/marketTypes';
export class SaudiMarketHandler {
  private static readonly ENDPOINTS = {
    QUOTE: (symbol: string) => `/quote?symbol=${symbol}&apikey=${API_KEY}`,
    STATISTICS: (symbol: string) => `/statistics?symbol=${symbol}&apikey=${API_KEY}`,
    INCOME: (symbol: string) => `/income_statement?symbol=${symbol}&period=quarterly&apikey=${API_KEY}`,
    BALANCE: (symbol: string) => `/balance_sheet?symbol=${symbol}&period=quarterly&apikey=${API_KEY}`,
    CASH_FLOW: (symbol: string) => `/cash_flow?symbol=${symbol}&period=quarterly&apikey=${API_KEY}`
  };

  private static readonly CACHE_DURATION = 300000; // 5 minutes
  private static cache = new Map<string, { data: any; timestamp: number }>();

  private static async fetchWithCache(endpoint: string): Promise<any> {
    try {
      const url = `${API_URL}${endpoint}`;
      const cached = this.cache.get(url);

      if (cached && Date.now() - cached.timestamp < this.CACHE_DURATION) {
        return cached.data;
      }

      console.log('Fetching URL:', url);
      const response = await fetch(url);

      if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`);
      }

      const data = await response.json();

      if (data.status === 'error') {
        throw new Error(data.message || 'API Error');
      }

      this.cache.set(url, {
        data,
        timestamp: Date.now()
      });

      return data;
    } catch (error) {
      console.error('Fetch error:', error);
      const cached = this.cache.get(endpoint);
      if (cached) {
        console.log('Using cached data due to fetch error');
        return cached.data;
      }
      return null;
    }
  }

  private static async getFundamentals(symbol: string): Promise<any> {
    try {
      const [statistics, incomeStatement, balanceSheet, cashFlow] = await Promise.all([
        this.fetchWithCache(this.ENDPOINTS.STATISTICS(symbol)),
        this.fetchWithCache(this.ENDPOINTS.INCOME(symbol)),
        this.fetchWithCache(this.ENDPOINTS.BALANCE(symbol)),
        this.fetchWithCache(this.ENDPOINTS.CASH_FLOW(symbol))
      ]);

      return {
        statistics,
        incomeStatement,
        balanceSheet,
        cashFlow
      };
    } catch (error) {
      console.error(`Error fetching fundamentals for ${symbol}:`, error);
      return null;
    }
  }

  static async getQuoteData(symbol: string): Promise<any> {
    try {
      const formattedSymbol = this.formatTadawulSymbol(symbol);
      const stockInfo = SAUDI_SYMBOLS.find(s =>
        s.symbol === formattedSymbol ||
        s.tickerSymbol === symbol
      );
```

```typescript
      if (!stockInfo) {
        return null;
      }

      const quoteData = await this.fetchWithCache(this.ENDPOINTS.QUOTE(formattedSymbol));
      if (!quoteData) {
        return null;
      }

      // Make fundamentals optional and don't let it block the main response
      const fundamentals = await this.getFundamentals(formattedSymbol)
        .catch(() => null);

      return {
        symbol: stockInfo.tickerSymbol,
        name: stockInfo.name,
        name_ar: stockInfo.name_ar,
        price: this.safeParseNumber(quoteData.close || quoteData.price),
        change: this.safeParseNumber(quoteData.change),
        percent_change: this.safeParseNumber(quoteData.percent_change),
        volume: this.safeParseNumber(quoteData.volume),
        timestamp: this.formatTimestamp(quoteData.timestamp),
        sector: stockInfo.sector,
        trading_info: {
          open: this.safeParseNumber(quoteData.open),
          high: this.safeParseNumber(quoteData.high),
          low: this.safeParseNumber(quoteData.low),
          previous_close: this.safeParseNumber(quoteData.previous_close),
          is_market_open: quoteData.is_market_open || false,
          fifty_two_week: quoteData.fifty_two_week
        },
        fundamentals: fundamentals
      };
    } catch (error) {
      console.error(`Error fetching quote for ${symbol}:`, error);
      return null;
    }
  }
}

static async analyzeMentionedCompanies(text: string): Promise<{
  companies: any[];
  sectors: Set<string>;
}> {
  try {
    console.log('Analyzing text for company mentions:', text);

    const mentionedStocks = SAUDI_SYMBOLS.filter(stock => {
      const nameMatch = stock.name && text.toLowerCase().includes(stock.name.toLowerCase());
      const tickerMatch = stock.tickerSymbol && text.includes(stock.tickerSymbol);
      const arabicMatch = stock.name_ar && text.includes(stock.name_ar);
      return nameMatch || tickerMatch || arabicMatch;
    });

    console.log('Found mentioned stocks:', mentionedStocks.length);

    if (mentionedStocks.length === 0) {
      return {
        companies: [],
        sectors: new Set<string>()
      };
    }

    const quotesPromises = mentionedStocks.map(async (stock) => {
      try {
        return await this.getQuoteData(stock.tickerSymbol);
      } catch (error) {
        console.error(`Error fetching data for ${stock.tickerSymbol}:`, error);
        return null;
      }
    });

    const quotes = (await Promise.all(quotesPromises))
      .filter(quote => quote !== null);

    console.log('Successfully fetched data for', quotes.length, 'companies');

    return {
      companies: quotes,
      sectors: new Set(mentionedStocks.map(stock => stock.sector))
    };
  } catch (error) {
    console.error('Error analyzing mentioned companies:', error);
    return {
      companies: [],
      sectors: new Set<string>()
    };
```

```typescript
    }
  }

  private static formatTadawulSymbol(symbol: string): string {
    const cleanSymbol = symbol.replace(/\.(SAU|TADAWUL)$/, '');
    return `${cleanSymbol}:TADAWUL`;
  }

  private static safeParseNumber(value: any): number {
    if (value === null || value === undefined) return 0;
    const parsed = parseFloat(value);
    return isNaN(parsed) ? 0 : parsed;
  }

  private static formatTimestamp(timestamp?: number | string): string {
    if (!timestamp) return new Date().toISOString();
    if (typeof timestamp === 'number') {
      return new Date(timestamp * 1000).toISOString();
    }
    return timestamp;
  }
}
```

```ts
// /app/market/types/marketTypes.ts

export interface ValidationResult {
    isValid: boolean;
    score: number;
    timestamp: string;
    issues: string[];
    warnings: string[];
}

export interface MarketDataQuality {
    freshness: {
        lastUpdate: string;
        isStale: boolean;
        age: number;
    };
    validation: {
        price: ValidationResult;
        volume: ValidationResult;
        fundamentals?: ValidationResult;
    };
    reliability: {
        dataCompleteness: number;
        sourceQuality: number;
        overallScore: number;
    };
}

export interface MarketError {
    code: string;
    message: string;
    severity: 'critical' | 'warning' | 'info';
    timestamp: string;
    context?: any;
}
```

```ts
// /app/types/chat.ts

export interface FinancialMetadata {
    instruments?: string[];
    sentiment?: 'bullish' | 'bearish' | 'neutral';
    riskLevel?: 'low' | 'medium' | 'high';
    confidenceScore?: number;
    technicalIndicators?: string[];
    fundamentalFactors?: string[];
  }

  export interface Message {
    id: string;
    role: string;
    content: string;
    metadata?: FinancialMetadata;
  }

  export interface ChatResponse {
    id: string;
    response: string;
    thinking: string;
    user_mood: string;
    suggested_questions: string[];
    financial_context: {
      instruments_mentioned: string[];
      market_sentiment: 'bullish' | 'bearish' | 'neutral';
      risk_level: 'low' | 'medium' | 'high';
      confidence_score: number;
      technical_indicators: string[];
      fundamental_factors: string[];
    };
    debug: {
      context_used: boolean;
      market_data_used: boolean;
    };
  }
```

```typescript
// /app/market/services/marketValidation.ts

import { ValidationResult } from '../types//marketTypes';

export class MarketValidation {
  private static readonly RULES = {
    PRICE: {
      MAX_DAILY_CHANGE: 0.20, // 20%
      MIN_PRICE: 0.01,
      MAX_PRICE: 10000
    },
    VOLUME: {
      MIN_DAILY_VOLUME: 100,
      MAX_SPIKE: 1000 // 1000% increase from average
    },
    FUNDAMENTALS: {
      MAX_PE_RATIO: 200,
      MIN_MARKET_CAP: 1000000,
      MAX_DEBT_EQUITY: 5
    }
  };

  static validatePrice(
    currentPrice: number,
    previousClose: number,
    lastUpdate: string
  ): ValidationResult {
    const issues: string[] = [];
    const warnings: string[] = [];
    let score = 100;

    // Check price range
    if (currentPrice < this.RULES.PRICE.MIN_PRICE) {
      issues.push('Price too low: ${currentPrice}');
      score -= 30;
    }
    if (currentPrice > this.RULES.PRICE.MAX_PRICE) {
      issues.push('Price too high: ${currentPrice}');
      score -= 30;
    }

    // Check price change
    if (previousClose > 0) {
      const priceChange = Math.abs(currentPrice - previousClose) / previousClose;
      if (priceChange > this.RULES.PRICE.MAX_DAILY_CHANGE) {
        warnings.push('Large price change: ${(priceChange * 100).toFixed(2)}%');
        score -= 20;
      }
    }

    return {
      isValid: issues.length === 0,
      score: Math.max(0, score),
      timestamp: new Date().toISOString(),
      issues,
      warnings
    };
  }

  static validateVolume(
    currentVolume: number,
    averageVolume: number
  ): ValidationResult {
    const issues: string[] = [];
    const warnings: string[] = [];
    let score = 100;

    if (currentVolume < this.RULES.VOLUME.MIN_DAILY_VOLUME) {
      issues.push('Volume too low: ${currentVolume}');
      score -= 30;
    }

    if (averageVolume > 0) {
      const volumeSpike = currentVolume / averageVolume;
      if (volumeSpike > this.RULES.VOLUME.MAX_SPIKE) {
        warnings.push('Unusual volume spike: ${volumeSpike.toFixed(2)}x average');
        score -= 20;
      }
    }

    return {
      isValid: issues.length === 0,
      score: Math.max(0, score),
      timestamp: new Date().toISOString(),
      issues,
      warnings
```

```typescript
    };
  }

  static validateFundamentals(data: {
    peRatio?: number;
    marketCap?: number;
    debtEquity?: number;
  }): ValidationResult {
    const issues: string[] = [];
    const warnings: string[] = [];
    let score = 100;

    if (data.peRatio && data.peRatio > this.RULES.FUNDAMENTALS.MAX_PE_RATIO) {
      warnings.push(`High P/E ratio: ${data.peRatio}`);
      score -= 15;
    }

    if (data.marketCap && data.marketCap < this.RULES.FUNDAMENTALS.MIN_MARKET_CAP) {
      issues.push(`Low market cap: ${data.marketCap}`);
      score -= 25;
    }

    if (data.debtEquity && data.debtEquity > this.RULES.FUNDAMENTALS.MAX_DEBT_EQUITY) {
      warnings.push(`High debt/equity ratio: ${data.debtEquity}`);
      score -= 15;
    }

    return {
      isValid: issues.length === 0,
      score: Math.max(0, score),
      timestamp: new Date().toISOString(),
      issues,
      warnings
    };
  }
}
```

```typescript
// /app/market/services/marketQuality.ts

import { MarketDataQuality, ValidationResult } from '../types/marketTypes';

export class MarketQuality {
  private static readonly MAX_DATA_AGE = 300000; // 5 minutes

  static assessDataQuality(
    priceValidation: ValidationResult,
    volumeValidation: ValidationResult,
    fundamentalsValidation: ValidationResult | undefined,
    lastUpdate: string
  ): MarketDataQuality {
    // Calculate data freshness
    const freshness = this.calculateFreshness(lastUpdate);

    // Calculate reliability scores
    const reliability = this.calculateReliability(
      priceValidation,
      volumeValidation,
      fundamentalsValidation,
      freshness
    );

    return {
      freshness,
      validation: {
        price: priceValidation,
        volume: volumeValidation,
        fundamentals: fundamentalsValidation
      },
      reliability
    };
  }

  private static calculateFreshness(lastUpdate: string): {
    lastUpdate: string;
    isStale: boolean;
    age: number;
  } {
    const now = Date.now();
    const updateTime = new Date(lastUpdate).getTime();
    const age = now - updateTime;

    return {
      lastUpdate,
      isStale: age > this.MAX_DATA_AGE,
      age
    };
  }

  private static calculateReliability(
    priceValidation: ValidationResult,
    volumeValidation: ValidationResult,
    fundamentalsValidation: ValidationResult | undefined,
    freshness: { isStale: boolean; age: number }
  ): {
    dataCompleteness: number;
    sourceQuality: number;
    overallScore: number;
  } {
    let dataCompleteness = 100;

    // Reduce completeness score for missing or invalid data
    if (!priceValidation.isValid) dataCompleteness -= 40;
    if (!volumeValidation.isValid) dataCompleteness -= 30;
    if (!fundamentalsValidation) dataCompleteness -= 20;

    // Calculate source quality based on validation scores
    const sourceQuality = (
      priceValidation.score * 0.4 +
      volumeValidation.score * 0.3 +
      (fundamentalsValidation?.score || 0) * 0.3
    );

    // Calculate overall score
    let overallScore = (
      dataCompleteness * 0.3 +
      sourceQuality * 0.4 +
      (freshness.isStale ? 50 : 100) * 0.3
    );

    // Ensure scores are within bounds
    return {
      dataCompleteness: Math.max(0, Math.min(100, dataCompleteness)),
      sourceQuality: Math.max(0, Math.min(100, sourceQuality)),
```

```
      overallScore: Math.max(0, Math.min(100, overallScore))
    };
  }
}
```

```
      overallScore: Math.max(0, Math.min(100, overallScore))
    };
  }
}
```

```typescript
export type DocumentCategory =
  | 'regulation'
  | 'profile'
  | 'research'
  | 'educational'
  | 'market-update';

export type DocumentLanguage = 'en' | 'ar';

export interface DocumentMetadata {
  id: string;
  title: string;
  category: DocumentCategory;
  tags: string[];
  language: DocumentLanguage;
  lastUpdated: string;
  source: string;
  version: string;
}

export interface FinancialDocument {
  metadata: DocumentMetadata;
  content: string;
  path: string;
}

export interface DocumentSearchResult {
  document: FinancialDocument;
  relevanceScore: number;
  matchedSegments: string[];
}

export interface DocumentContext {
  relevantDocuments: DocumentSearchResult[];
  confidence: number;
  context: string;
}

export interface DocumentStore {
  documents: Map<string, FinancialDocument>;
  indexes: {
    byCategory: Map<DocumentCategory, string[]>;
    byTag: Map<string, string[]>;
    byLanguage: Map<DocumentLanguage, string[]>;
  };
}

export interface DocumentQuery {
  searchTerm?: string;
  category?: DocumentCategory;
  tags?: string[];
  language?: DocumentLanguage;
  limit?: number;
}
```

```typescript
import { FinancialDocument, DocumentCategory, DocumentMetadata } from '../types/document';
import FinancialDocumentStore from '../store/documentStore';

export class DocumentLoader {
  private store: FinancialDocumentStore;

  constructor() {
    this.store = FinancialDocumentStore.getInstance();
  }

  async loadDocument(file: File): Promise<FinancialDocument> {
    try {
      // Read file content
      const content = await this.readFileContent(file);

      // Generate metadata
      const metadata = await this.generateMetadata(file);

      // Create document
      const document: FinancialDocument = {
        metadata,
        content,
        path: file.name,
      };

      // Add to store
      await this.store.addDocument(document);

      return document;
    } catch (error) {
      console.error('Error loading document:', error);
      throw error;
    }
  }

  private async readFileContent(file: File): Promise<string> {
    return new Promise((resolve, reject) => {
      const reader = new FileReader();

      reader.onload = (e) => {
        resolve(e.target?.result as string);
      };

      reader.onerror = (e) => {
        reject(new Error('Error reading file'));
      };

      reader.readAsText(file);
    });
  }

  private async generateMetadata(file: File): Promise<DocumentMetadata> {
    // Extract category from file path/name
    const category = this.determineCategory(file.name);

    // Generate unique ID
    const id = this.generateDocumentId(file.name);

    return {
      id,
      title: file.name.replace(/\.[^/.]+$/, ""), // Remove extension
      category,
      tags: this.generateTags(file.name, category),
      language: this.determineLanguage(file.name),
      lastUpdated: new Date().toISOString(),
      source: 'local',
      version: '1.0'
    };
  }

  private determineCategory(filename: string): DocumentCategory {
    // Implement logic to determine category based on filename or path
    if (filename.includes('regulation')) return 'regulation';
    if (filename.includes('profile')) return 'profile';
    if (filename.includes('research')) return 'research';
    if (filename.includes('educational')) return 'educational';
    return 'market-update';
  }

  private generateDocumentId(filename: string): string {
    return `doc_${Date.now()}_${Math.random().toString(36).substr(2, 9)}`;
  }

  private generateTags(filename: string, category: DocumentCategory): string[] {
    const tags = new Set<string>();
```

```typescript
      // Add category as tag
      tags.add(category);

      // Add filename-based tags
      const words = filename
        .toLowerCase()
        .replace(/[^a-z0-9\s]/g, ' ')
        .split(' ')
        .filter(word => word.length > 3);

      words.forEach(word => tags.add(word));

      return Array.from(tags);
    }

    private determineLanguage(filename: string): 'en' | 'ar' {
      // Simple language detection based on filename
      // Enhance this based on your needs
      return filename.includes('_ar') ? 'ar' : 'en';
    }
  }

export const documentLoader = new DocumentLoader();
```

```typescript
import { useState, useCallback } from 'react';
import { useDocuments } from '../context/DocumentContext';
import { DocumentSearchResult, DocumentQuery } from '../types/document';

interface ChatDocumentContext {
  relevantDocuments: DocumentSearchResult[];
  documentContext: string;
  confidence: number;
}

export function useChatDocuments() {
  const { searchDocuments } = useDocuments();
  const [lastContext, setLastContext] = useState<ChatDocumentContext | null>(null);

  const getDocumentContext = useCallback(async (message: string): Promise<ChatDocumentContext> => {
    // Extract key terms and topics from the message
    const query: DocumentQuery = {
      searchTerm: message,
      limit: 5 // Limit to most relevant documents
    };

    try {
      // Search for relevant documents
      const relevantDocs = await searchDocuments(query);

      // Build context from relevant documents
      const context = buildDocumentContext(relevantDocs);

      // Calculate confidence based on relevance scores
      const confidence = calculateContextConfidence(relevantDocs);

      const result = {
        relevantDocuments: relevantDocs,
        documentContext: context,
        confidence
      };

      setLastContext(result);
      return result;

    } catch (error) {
      console.error('Error getting document context:', error);
      return {
        relevantDocuments: [],
        documentContext: '',
        confidence: 0
      };
    }
  }, [searchDocuments]);

  const buildDocumentContext = (documents: DocumentSearchResult[]): string => {
    let context = '';

    documents.forEach((doc, index) => {
      // Add document metadata
      context += `[Document ${index + 1}: ${doc.document.metadata.title}]\n`;

      // Add matched segments if available
      if (doc.matchedSegments.length > 0) {
        context += 'Relevant excerpts:\n';
        doc.matchedSegments.forEach(segment => {
          context += `- ${segment}\n`;
        });
      } else {
        // If no specific matches, add document summary
        const summary = summarizeDocument(doc.document.content);
        context += `Summary: ${summary}\n`;
      }

      context += '\n';
    });

    return context;
  };

  const summarizeDocument = (content: string): string => {
    // Simple summarization - take first few sentences
    const sentences = content.split(/[.!?]+/).filter(Boolean);
    return sentences.slice(0, 2).join('. ') + '.';
  };

  const calculateContextConfidence = (documents: DocumentSearchResult[]): number => {
    if (documents.length === 0) return 0;

    // Average relevance scores
    const avgRelevance = documents.reduce((sum, doc) =>
```

```
      sum + doc.relevanceScore, 0) / documents.length;

    // Consider number of documents found
    const coverageScore = Math.min(documents.length / 5, 1); // Max score with 5 docs

    return (avgRelevance * 0.7 + coverageScore * 0.3);
  };

  return {
    getDocumentContext,
    lastContext
  };
}
```

```
      sum + doc.relevanceScore, 0) / documents.length;


    // Consider number of documents found
    const coverageScore = Math.min(documents.length / 5, 1); // Max score with 5 docs


    return (avgRelevance * 0.7 + coverageScore * 0.3);
  };


  return {
    getDocumentContext,
    lastContext
  };
}
```

```typescript
import {
    DocumentStore,
    FinancialDocument,
    DocumentCategory,
    DocumentQuery,
    DocumentSearchResult
} from '../types/document';

export class FinancialDocumentStore {
    private store: DocumentStore;
    private static instance: FinancialDocumentStore;

    private constructor() {
        this.store = {
            documents: new Map(),
            indexes: {
                byCategory: new Map(),
                byTag: new Map(),
                byLanguage: new Map()
            }
        };
    }

    static getInstance(): FinancialDocumentStore {
        if (!FinancialDocumentStore.instance) {
            FinancialDocumentStore.instance = new FinancialDocumentStore();
        }
        return FinancialDocumentStore.instance;
    }

    async addDocument(document: FinancialDocument) {
        // Add to main store
        this.store.documents.set(document.metadata.id, document);

        // Update indexes
        this.updateIndexes(document);
    }

    private updateIndexes(document: FinancialDocument) {
        // Update category index
        const categoryDocs = this.store.indexes.byCategory.get(document.metadata.category) || [];
        this.store.indexes.byCategory.set(
            document.metadata.category,
            [...categoryDocs, document.metadata.id]
        );

        // Update tag index
        document.metadata.tags.forEach(tag => {
            const tagDocs = this.store.indexes.byTag.get(tag) || [];
            this.store.indexes.byTag.set(tag, [...tagDocs, document.metadata.id]);
        });

        // Update language index
        const langDocs = this.store.indexes.byLanguage.get(document.metadata.language) || [];
        this.store.indexes.byLanguage.set(
            document.metadata.language,
            [...langDocs, document.metadata.id]
        );
    }

    async searchDocuments(query: DocumentQuery): Promise<DocumentSearchResult[]> {
        const results: DocumentSearchResult[] = [];

        // Get initial document set based on filters
        let documentSet = this.getFilteredDocuments(query);

        // If search term exists, perform text search
        if (query.searchTerm) {
            documentSet = this.performTextSearch(documentSet, query.searchTerm);
        }

        // Convert to search results with relevance scores
        documentSet.forEach(doc => {
            results.push({
                document: doc,
                relevanceScore: this.calculateRelevanceScore(doc, query),
                matchedSegments: this.findMatchedSegments(doc, query.searchTerm)
            });
        });

        // Sort by relevance and apply limit
        return results
            .sort((a, b) => b.relevanceScore - a.relevanceScore)
            .slice(0, query.limit || results.length);
    }
```

```typescript
  private getFilteredDocuments(query: DocumentQuery): FinancialDocument[] {
    let documents = Array.from(this.store.documents.values());

    if (query.category) {
      documents = documents.filter(doc =>
        doc.metadata.category === query.category
      );
    }

    if (query.tags?.length) {
      documents = documents.filter(doc =>
        query.tags!.some(tag => doc.metadata.tags.includes(tag))
      );
    }

    if (query.language) {
      documents = documents.filter(doc =>
        doc.metadata.language === query.language
      );
    }

    return documents;
  }

  private performTextSearch(
    documents: FinancialDocument[],
    searchTerm: string
  ): FinancialDocument[] {
    const normalizedSearch = searchTerm.toLowerCase();
    return documents.filter(doc => {
      // Search in content
      if (doc.content.toLowerCase().includes(normalizedSearch)) {
        return true;
      }
      // Search in metadata
      if (doc.metadata.title.toLowerCase().includes(normalizedSearch)) {
        return true;
      }
      // Search in tags
      if (doc.metadata.tags.some(tag =>
        tag.toLowerCase().includes(normalizedSearch))
      ) {
        return true;
      }
      return false;
    });
  }

  private calculateRelevanceScore(
    document: FinancialDocument,
    query: DocumentQuery
  ): number {
    let score = 1;

    if (query.searchTerm) {
      // Calculate text match relevance
      const termFrequency = this.calculateTermFrequency(
        document.content,
        query.searchTerm
      );
      score *= (1 + termFrequency);
    }

    // Boost score based on recency
    const docAge = Date.now() - new Date(document.metadata.lastUpdated).getTime();
    const recencyBoost = 1 + (1 / (1 + docAge / (1000 * 60 * 60 * 24))); // Decay over days
    score *= recencyBoost;

    return score;
  }

  private calculateTermFrequency(text: string, term: string): number {
    const regex = new RegExp(term, 'gi');
    const matches = text.match(regex);
    return matches ? matches.length / text.length : 0;
  }

  private findMatchedSegments(
    document: FinancialDocument,
    searchTerm?: string
  ): string[] {
    if (!searchTerm) return [];

    const segments: string[] = [];
    const regex = new RegExp(`(.{0,50}${searchTerm}.{0,50})`, 'gi');
    let match;
```

```
      while ((match = regex.exec(document.content)) !== null) {
        segments.push(match[1]);
      }

      return segments;
    }

    getDocumentById(id: string): FinancialDocument | undefined {
      return this.store.documents.get(id);
    }

    getDocumentsByCategory(category: DocumentCategory): FinancialDocument[] {
      const docIds = this.store.indexes.byCategory.get(category) || [];
      return docIds
        .map(id => this.store.documents.get(id))
        .filter((doc): doc is FinancialDocument => doc !== undefined);
    }
  }

  export default FinancialDocumentStore;
```

```css
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer base {
    :root {
        --background: 0 0% 100%;
        --foreground: 0 0% 3.9%;
        --card: 0 0% 100%;
        --card-foreground: 0 0% 3.9%;
        --popover: 0 0% 100%;
        --popover-foreground: 0 0% 3.9%;
        --primary: 0 0% 9%;
        --primary-foreground: 0 0% 98%;
        --secondary: 0 0% 96.1%;
        --secondary-foreground: 0 0% 9%;
        --muted: 0 0% 96.1%;
        --muted-foreground: 0 0% 45.1%;
        --accent: 0 0% 96.1%;
        --accent-foreground: 0 0% 9%;
        --destructive: 0 84.2% 60.2%;
        --destructive-foreground: 0 0% 98%;
        --border: 0 0% 89.8%;
        --input: 0 0% 89.8%;
        --ring: 0 0% 3.9%;
        --radius: 0.75rem;
        --chart-1: 12 76% 61%;
        --chart-2: 173 58% 39%;
        --chart-3: 197 37% 24%;
        --chart-4: 43 74% 66%;
        --chart-5: 27 87% 67%;
    }

    .dark {
        --background: 0 0% 3.9%;
        --foreground: 0 0% 98%;
        --card: 0 0% 3.9%;
        --card-foreground: 0 0% 98%;
        --popover: 0 0% 3.9%;
        --popover-foreground: 0 0% 98%;
        --primary: 0 0% 98%;
        --primary-foreground: 0 0% 9%;
        --secondary: 0 0% 14.9%;
        --secondary-foreground: 0 0% 98%;
        --muted: 0 0% 14.9%;
        --muted-foreground: 0 0% 63.9%;
        --accent: 0 0% 14.9%;
        --accent-foreground: 0 0% 98%;
        --destructive: 0 62.8% 30.6%;
        --destructive-foreground: 0 0% 98%;
        --border: 0 0% 14.9%;
        --input: 0 0% 14.9%;
        --ring: 0 0% 83.1%;
        --chart-1: 220 70% 50%;
        --chart-2: 160 60% 45%;
        --chart-3: 30 80% 55%;
        --chart-4: 280 65% 60%;
        --chart-5: 340 75% 55%;
    }
}

/* globals.css */
@layer base {
    .tabular-nums {
      font-variant-numeric: tabular-nums;
    }
  }

  @layer utilities {
    .line-clamp-3 {
      display: -webkit-box;
      -webkit-line-clamp: 3;
      -webkit-box-orient: vertical;
      overflow: hidden;
    }
  }

@layer base {
    * {
        @apply border-border;
    }
    body {
        @apply bg-background text-foreground;
    }
}
```

```bash
#!/bin/bash

# Function to print usage
print_usage() {
    echo "Usage: $0 <output_pdf> <directory1> [<directory2> ...]"
    echo "Example: $0 all_code.pdf /path/to/project1 /path/to/project2"
}

# Check if at least two arguments are provided
if [ $# -lt 2 ]; then
    print_usage
    exit 1
fi

# Output PDF file name
OUTPUT="$1"
shift

# Temporary PS file
TEMP_PS="temp.ps"

# Array of file extensions to include
EXTENSIONS=("c" "cpp" "h" "hpp" "py" "java" "js" "html" "css" "php" "rb" "go" "rs" "swift" "kt" "scala" "pl" "sh" "sql" "js" "ts")

# Function to get the appropriate language for enscript
get_language() {
    case "$1" in
        c|h|cpp|hpp) echo "c" ;;
        py) echo "python" ;;
        java) echo "java" ;;
        js|ts) echo "javascript" ;;
        html|css) echo "html" ;;
        php) echo "php" ;;
        rb) echo "ruby" ;;
        go) echo "go" ;;
        rs) echo "rust" ;;
        swift) echo "swift" ;;
        kt) echo "kotlin" ;;
        scala) echo "scala" ;;
        pl) echo "perl" ;;
        sh) echo "bash" ;;
        sql) echo "sql" ;;
        json) echo "javascript" ;;
        *) echo "text" ;;
    esac
}

# Function to generate the find command for a single directory
generate_find_command() {
    local dir="$1"
    local cmd="find \"$dir\" \( -type d \( -name node_modules -o -name .next \) -prune \) -o \( -type f \("
    for i in "${!EXTENSIONS[@]}"; do
        if [ $i -ne 0 ]; then
            cmd+=" -o"
        fi
        cmd+=" -name \"*.${EXTENSIONS[$i]}\""
    done
    cmd+=" \) -print \)"
    echo "$cmd"
}

# Clear the temporary PS file if it exists
> "$TEMP_PS"

# Process each directory
for dir in "$@"; do
    if [ ! -d "$dir" ]; then
        echo "Warning: $dir is not a valid directory. Skipping."
        continue
    fi

    echo "Processing directory: $dir"

    # Generate and execute the find command
    eval "$(generate_find_command "$dir")" | while read -r file; do
        echo "Converting $file"
        # Get the file extension
        ext="${file##*.}"
        # Get the appropriate language for enscript
        lang=$(get_language "$ext")
        # Use the appropriate syntax highlighting based on the file extension
        enscript -p - --highlight="$lang" --color=1 -fCourier8 --header="$file|Page \$% of \$=" "$file" >> "$TEMP_PS"
    done
done
```

```bash
# Check if any files were processed
if [ ! -s "$TEMP_PS" ]; then
    echo "Error: No files were found or processed. The output PDF will not be created."
    rm "$TEMP_PS"
    exit 1
fi

# Convert PostScript to PDF
ps2pdf "$TEMP_PS" "$OUTPUT"

# Remove temporary PostScript file
rm "$TEMP_PS"

echo "Conversion complete. Output saved as $OUTPUT"
```

```
/// <reference types="next" />
/// <reference types="next/image-types/global" />

// NOTE: This file should not be edited
// see https://nextjs.org/docs/basic-features/typescript for more information.
```

```ts
import type { Config } from "tailwindcss";

const config = {
  darkMode: ["class"],
  content: [
    "./pages/**/*.{ts,tsx}",
    "./components/**/*.{ts,tsx}",
    "./app/**/*.{ts,tsx}",
    "./src/**/*.{ts,tsx}",
  ],
  prefix: "",
  theme: {
    container: {
      center: true,
      padding: "2rem",
      screens: {
        "2xl": "1400px",
      },
    },
    extend: {
      colors: {
        border: "hsl(var(--border))",
        input: "hsl(var(--input))",
        ring: "hsl(var(--ring))",
        background: "hsl(var(--background))",
        foreground: "hsl(var(--foreground))",
        primary: {
          DEFAULT: "hsl(var(--primary))",
          foreground: "hsl(var(--primary-foreground))",
        },
        secondary: {
          DEFAULT: "hsl(var(--secondary))",
          foreground: "hsl(var(--secondary-foreground))",
        },
        destructive: {
          DEFAULT: "hsl(var(--destructive))",
          foreground: "hsl(var(--destructive-foreground))",
        },
        muted: {
          DEFAULT: "hsl(var(--muted))",
          foreground: "hsl(var(--muted-foreground))",
        },
        accent: {
          DEFAULT: "hsl(var(--accent))",
          foreground: "hsl(var(--accent-foreground))",
        },
        popover: {
          DEFAULT: "hsl(var(--popover))",
          foreground: "hsl(var(--popover-foreground))",
        },
        card: {
          DEFAULT: "hsl(var(--card))",
          foreground: "hsl(var(--card-foreground))",
        },
      },
      borderRadius: {
        lg: "var(--radius)",
        md: "calc(var(--radius) - 2px)",
        sm: "calc(var(--radius) - 4px)",
      },
      keyframes: {
        "accordion-down": {
          from: { height: "0" },
          to: { height: "var(--radix-accordion-content-height)" },
        },
        "accordion-up": {
          from: { height: "var(--radix-accordion-content-height)" },
          to: { height: "0" },
        },
        fadeIn: {
          "0%": { opacity: "0" },
          "100%": { opacity: "1" },
        },
        fadeInUp: {
          "0%": { opacity: "0", transform: "translateY(8px)" },
          "100%": { opacity: "1", transform: "translateY(0)" },
        },
        shimmer: {
          "100%": { transform: "translateX(100%)" },
        },
      },
      animation: {
        "accordion-down": "accordion-down 0.2s ease-out",
        "accordion-up": "accordion-up 0.2s ease-out",
        "fade-in": "fadeIn 0.5s ease-out forwards",
        "fade-in-up": "fadeInUp 0.5s ease-out forwards",
```

```
        "fade-in-up-fast": "fadeInUp 0.3s ease-out forwards",
        "fade-in-up-slow":
          "fadeInUp 0.8s cubic-bezier(0.4, 0, 0.2, 1) forwards",
        shimmer: "shimmer 2s linear infinite",
      },
    },
  },
  plugins: [require("tailwindcss-animate")],
} satisfies Config;

export default config;
```

```javascript
// themes.ts

export const themes = {
  neutral: {
    light: {
      background: "0 0% 100%",
      foreground: "0 0% 3.9%",
      card: "0 0% 100%",
      "card-foreground": "0 0% 3.9%",
      popover: "0 0% 100%",
      "popover-foreground": "0 0% 3.9%",
      primary: "0 0% 9%",
      "primary-foreground": "0 0% 98%",
      secondary: "0 0% 96.1%",
      "secondary-foreground": "0 0% 9%",
      muted: "0 0% 96.1%",
      "muted-foreground": "0 0% 45.1%",
      accent: "0 0% 96.1%",
      "accent-foreground": "0 0% 9%",
      destructive: "0 84.2% 60.2%",
      "destructive-foreground": "0 0% 98%",
      border: "0 0% 89.8%",
      input: "0 0% 89.8%",
      ring: "0 0% 3.9%",
      radius: "0.75rem",
      "chart-1": "12 76% 61%",
      "chart-2": "173 58% 39%",
      "chart-3": "197 37% 24%",
      "chart-4": "43 74% 66%",
      "chart-5": "27 87% 67%",
    },
    dark: {
      background: "0 0% 3.9%",
      foreground: "0 0% 98%",
      card: "0 0% 3.9%",
      "card-foreground": "0 0% 98%",
      popover: "0 0% 3.9%",
      "popover-foreground": "0 0% 98%",
      primary: "0 0% 98%",
      "primary-foreground": "0 0% 9%",
      secondary: "0 0% 14.9%",
      "secondary-foreground": "0 0% 98%",
      muted: "0 0% 14.9%",
      "muted-foreground": "0 0% 63.9%",
      accent: "0 0% 14.9%",
      "accent-foreground": "0 0% 98%",
      destructive: "0 62.8% 30.6%",
      "destructive-foreground": "0 0% 98%",
      border: "0 0% 14.9%",
      input: "0 0% 14.9%",
      ring: "0 0% 83.1%",
      radius: "0.75rem",
      "chart-1": "220 70% 50%",
      "chart-2": "160 60% 45%",
      "chart-3": "30 80% 55%",
      "chart-4": "280 65% 60%",
      "chart-5": "340 75% 55%",
    },
  },
  red: {
    light: {
      background: "0 0% 100%",
      foreground: "0 0% 3.9%",
      card: "0 0% 100%",
      "card-foreground": "0 0% 3.9%",
      popover: "0 0% 100%",
      "popover-foreground": "0 0% 3.9%",
      primary: "0 72.2% 50.6%",
      "primary-foreground": "0 85.7% 97.3%",
      secondary: "0 0% 96.1%",
      "secondary-foreground": "0 0% 9%",
      muted: "0 0% 96.1%",
      "muted-foreground": "0 0% 45.1%",
      accent: "0 0% 96.1%",
      "accent-foreground": "0 0% 9%",
      destructive: "0 84.2% 60.2%",
      "destructive-foreground": "0 0% 98%",
      border: "0 0% 89.8%",
      input: "0 0% 89.8%",
      ring: "0 72.2% 50.6%",
      radius: "0.75rem",
      "chart-1": "12 76% 61%",
      "chart-2": "173 58% 39%",
      "chart-3": "197 37% 24%",
      "chart-4": "43 74% 66%",
      "chart-5": "27 87% 67%",
```

```javascript
    },
    dark: {
      background: "0 0% 3.9%",
      foreground: "0 0% 98%",
      card: "0 0% 3.9%",
      "card-foreground": "0 0% 98%",
      popover: "0 0% 3.9%",
      "popover-foreground": "0 0% 98%",
      primary: "0 72.2% 50.6%",
      "primary-foreground": "0 85.7% 97.3%",
      secondary: "0 0% 14.9%",
      "secondary-foreground": "0 0% 98%",
      muted: "0 0% 14.9%",
      "muted-foreground": "0 0% 63.9%",
      accent: "0 0% 14.9%",
      "accent-foreground": "0 0% 98%",
      destructive: "0 62.8% 30.6%",
      "destructive-foreground": "0 0% 98%",
      border: "0 0% 14.9%",
      input: "0 0% 14.9%",
      ring: "0 72.2% 50.6%",
      radius: "0.75rem",
      "chart-1": "220 70% 50%",
      "chart-2": "160 60% 45%",
      "chart-3": "30 80% 55%",
      "chart-4": "280 65% 60%",
      "chart-5": "340 75% 55%",
    },
  },
  violet: {
    light: {
      background: "0 0% 100%",
      foreground: "224 71.4% 4.1%",
      card: "0 0% 100%",
      "card-foreground": "224 71.4% 4.1%",
      popover: "0 0% 100%",
      "popover-foreground": "224 71.4% 4.1%",
      primary: "262.1 83.3% 57.8%",
      "primary-foreground": "210 20% 98%",
      secondary: "220 14.3% 95.9%",
      "secondary-foreground": "220.9 39.3% 11%",
      muted: "220 14.3% 95.9%",
      "muted-foreground": "220 8.9% 46.1%",
      accent: "220 14.3% 95.9%",
      "accent-foreground": "220.9 39.3% 11%",
      destructive: "0 84.2% 60.2%",
      "destructive-foreground": "210 20% 98%",
      border: "220 13% 91%",
      input: "220 13% 91%",
      ring: "262.1 83.3% 57.8%",
      radius: "0.75rem",
      "chart-1": "12 76% 61%",
      "chart-2": "173 58% 39%",
      "chart-3": "197 37% 24%",
      "chart-4": "43 74% 66%",
      "chart-5": "27 87% 67%",
    },
    dark: {
      background: "224 71.4% 4.1%",
      foreground: "210 20% 98%",
      card: "224 71.4% 4.1%",
      "card-foreground": "210 20% 98%",
      popover: "224 71.4% 4.1%",
      "popover-foreground": "210 20% 98%",
      primary: "263.4 70% 50.4%",
      "primary-foreground": "210 20% 98%",
      secondary: "215 27.9% 16.9%",
      "secondary-foreground": "210 20% 98%",
      muted: "215 27.9% 16.9%",
      "muted-foreground": "217.9 10.6% 64.9%",
      accent: "215 27.9% 16.9%",
      "accent-foreground": "210 20% 98%",
      destructive: "0 62.8% 30.6%",
      "destructive-foreground": "210 20% 98%",
      border: "215 27.9% 16.9%",
      input: "215 27.9% 16.9%",
      ring: "263.4 70% 50.4%",
      radius: "0.75rem",
      "chart-1": "220 70% 50%",
      "chart-2": "160 60% 45%",
      "chart-3": "30 80% 55%",
      "chart-4": "280 65% 60%",
      "chart-5": "340 75% 55%",
    },
  },
  blue: {
```

```js
    light: {
      background: "0 0% 100%",
      foreground: "222.2 84% 4.9%",
      card: "0 0% 100%",
      "card-foreground": "222.2 84% 4.9%",
      popover: "0 0% 100%",
      "popover-foreground": "222.2 84% 4.9%",
      primary: "221.2 83.2% 53.3%",
      "primary-foreground": "210 40% 98%",
      secondary: "210 40% 96.1%",
      "secondary-foreground": "222.2 47.4% 11.2%",
      muted: "210 40% 96.1%",
      "muted-foreground": "215.4 16.3% 46.9%",
      accent: "210 40% 96.1%",
      "accent-foreground": "222.2 47.4% 11.2%",
      destructive: "0 84.2% 60.2%",
      "destructive-foreground": "210 40% 98%",
      border: "214.3 31.8% 91.4%",
      input: "214.3 31.8% 91.4%",
      ring: "221.2 83.2% 53.3%",
      radius: "0.75rem",
      "chart-1": "12 76% 61%",
      "chart-2": "173 58% 39%",
      "chart-3": "197 37% 24%",
      "chart-4": "43 74% 66%",
      "chart-5": "27 87% 67%",
    },
    dark: {
      background: "222.2 84% 4.9%",
      foreground: "210 40% 98%",
      card: "222.2 84% 4.9%",
      "card-foreground": "210 40% 98%",
      popover: "222.2 84% 4.9%",
      "popover-foreground": "210 40% 98%",
      primary: "217.2 91.2% 59.8%",
      "primary-foreground": "222.2 47.4% 11.2%",
      secondary: "217.2 32.6% 17.5%",
      "secondary-foreground": "210 40% 98%",
      muted: "217.2 32.6% 17.5%",
      "muted-foreground": "215 20.2% 65.1%",
      accent: "217.2 32.6% 17.5%",
      "accent-foreground": "210 40% 98%",
      destructive: "0 62.8% 30.6%",
      "destructive-foreground": "210 40% 98%",
      border: "217.2 32.6% 17.5%",
      input: "217.2 32.6% 17.5%",
      ring: "224.3 76.3% 48%",
      "chart-1": "220 70% 50%",
      "chart-2": "160 60% 45%",
      "chart-3": "30 80% 55%",
      "chart-4": "280 65% 60%",
      "chart-5": "340 75% 55%",
    },
  },
  tangerine: {
    light: {
      background: "0 0% 100%",
      foreground: "20 14.3% 4.1%",
      card: "0 0% 100%",
      "card-foreground": "20 14.3% 4.1%",
      popover: "0 0% 100%",
      "popover-foreground": "20 14.3% 4.1%",
      primary: "24.6 95% 53.1%",
      "primary-foreground": "60 9.1% 97.8%",
      secondary: "60 4.8% 95.9%",
      "secondary-foreground": "24 9.8% 10%",
      muted: "60 4.8% 95.9%",
      "muted-foreground": "25 5.3% 44.7%",
      accent: "60 4.8% 95.9%",
      "accent-foreground": "24 9.8% 10%",
      destructive: "0 84.2% 60.2%",
      "destructive-foreground": "60 9.1% 97.8%",
      border: "20 5.9% 90%",
      input: "20 5.9% 90%",
      ring: "24.6 95% 53.1%",
      radius: "0.75rem",
      "chart-1": "12 76% 61%",
      "chart-2": "173 58% 39%",
      "chart-3": "197 37% 24%",
      "chart-4": "43 74% 66%",
      "chart-5": "27 87% 67%",
    },
    dark: {
      background: "20 14.3% 4.1%",
      foreground: "60 9.1% 97.8%",
      card: "20 14.3% 4.1%",
```

```
      "card-foreground": "60 9.1% 97.8%",
      popover: "20 14.3% 4.1%",
      "popover-foreground": "60 9.1% 97.8%",
      primary: "20.5 90.2% 48.2%",
      "primary-foreground": "60 9.1% 97.8%",
      secondary: "12 6.5% 15.1%",
      "secondary-foreground": "60 9.1% 97.8%",
      muted: "12 6.5% 15.1%",
      "muted-foreground": "24 5.4% 63.9%",
      accent: "12 6.5% 15.1%",
      "accent-foreground": "60 9.1% 97.8%",
      destructive: "0 72.2% 50.6%",
      "destructive-foreground": "60 9.1% 97.8%",
      border: "12 6.5% 15.1%",
      input: "12 6.5% 15.1%",
      ring: "20.5 90.2% 48.2%",
      "chart-1": "220 70% 50%",
      "chart-2": "160 60% 45%",
      "chart-3": "30 80% 55%",
      "chart-4": "280 65% 60%",
      "chart-5": "340 75% 55%",
    },
  },
  emerald: {
    light: {
      background: "0 0% 100%",
      foreground: "240 10% 3.9%",
      card: "0 0% 100%",
      "card-foreground": "240 10% 3.9%",
      popover: "0 0% 100%",
      "popover-foreground": "240 10% 3.9%",
      primary: "142.1 76.2% 36.3%",
      "primary-foreground": "355.7 100% 97.3%",
      secondary: "240 4.8% 95.9%",
      "secondary-foreground": "240 5.9% 10%",
      muted: "240 4.8% 95.9%",
      "muted-foreground": "240 3.8% 46.1%",
      accent: "240 4.8% 95.9%",
      "accent-foreground": "240 5.9% 10%",
      destructive: "0 84.2% 60.2%",
      "destructive-foreground": "0 0% 98%",
      border: "240 5.9% 90%",
      input: "240 5.9% 90%",
      ring: "142.1 76.2% 36.3%",
      radius: "0.75rem",
      "chart-1": "12 76% 61%",
      "chart-2": "173 58% 39%",
      "chart-3": "197 37% 24%",
      "chart-4": "43 74% 66%",
      "chart-5": "27 87% 67%",
    },
    dark: {
      background: "20 14.3% 4.1%",
      foreground: "0 0% 95%",
      card: "24 9.8% 10%",
      "card-foreground": "0 0% 95%",
      popover: "0 0% 9%",
      "popover-foreground": "0 0% 95%",
      primary: "142.1 70.6% 45.3%",
      "primary-foreground": "144.9 80.4% 10%",
      secondary: "240 3.7% 15.9%",
      "secondary-foreground": "0 0% 98%",
      muted: "0 0% 15%",
      "muted-foreground": "240 5% 64.9%",
      accent: "12 6.5% 15.1%",
      "accent-foreground": "0 0% 98%",
      destructive: "0 62.8% 30.6%",
      "destructive-foreground": "0 85.7% 97.3%",
      border: "240 3.7% 15.9%",
      input: "240 3.7% 15.9%",
      ring: "142.4 71.8% 29.2%",
      "chart-1": "220 70% 50%",
      "chart-2": "160 60% 45%",
      "chart-3": "30 80% 55%",
      "chart-4": "280 65% 60%",
      "chart-5": "340 75% 55%",
    },
  },
  amber: {
    light: {
      background: "0 0% 100%",
      foreground: "20 14.3% 4.1%",
      card: "0 0% 100%",
      "card-foreground": "20 14.3% 4.1%",
      popover: "0 0% 100%",
      "popover-foreground": "20 14.3% 4.1%",
```

```
      primary: "47.9 95.8% 53.1%",
      "primary-foreground": "26 83.3% 14.1%",
      secondary: "60 4.8% 95.9%",
      "secondary-foreground": "24 9.8% 10%",
      muted: "60 4.8% 95.9%",
      "muted-foreground": "25 5.3% 44.7%",
      accent: "60 4.8% 95.9%",
      "accent-foreground": "24 9.8% 10%",
      destructive: "0 84.2% 60.2%",
      "destructive-foreground": "60 9.1% 97.8%",
      border: "20 5.9% 90%",
      input: "20 5.9% 90%",
      ring: "20 14.3% 4.1%",
      radius: "0.75rem",
      "chart-1": "12 76% 61%",
      "chart-2": "173 58% 39%",
      "chart-3": "197 37% 24%",
      "chart-4": "43 74% 66%",
      "chart-5": "27 87% 67%",
    },
    dark: {
      background: "20 14.3% 4.1%",
      foreground: "60 9.1% 97.8%",
      card: "20 14.3% 4.1%",
      "card-foreground": "60 9.1% 97.8%",
      popover: "20 14.3% 4.1%",
      "popover-foreground": "60 9.1% 97.8%",
      primary: "47.9 95.8% 53.1%",
      "primary-foreground": "26 83.3% 14.1%",
      secondary: "12 6.5% 15.1%",
      "secondary-foreground": "60 9.1% 97.8%",
      muted: "12 6.5% 15.1%",
      "muted-foreground": "24 5.4% 63.9%",
      accent: "12 6.5% 15.1%",
      "accent-foreground": "60 9.1% 97.8%",
      destructive: "0 62.8% 30.6%",
      "destructive-foreground": "60 9.1% 97.8%",
      border: "12 6.5% 15.1%",
      input: "12 6.5% 15.1%",
      ring: "35.5 91.7% 32.9%",
      "chart-1": "220 70% 50%",
      "chart-2": "160 60% 45%",
      "chart-3": "30 80% 55%",
      "chart-4": "280 65% 60%",
      "chart-5": "340 75% 55%",
    },
  },
};
```

```typescript
// lib/utils.ts
import { type ClassValue, clsx } from "clsx";
import { twMerge } from "tailwind-merge";

export function cn(...inputs: ClassValue[]) {
  return twMerge(clsx(inputs));
}

export function formatNumber(
  value: number,
  currency?: string,
  compact = false
): string {
  if (typeof value !== "number" || isNaN(value)) {
    return "N/A";
  }

  const formatter = new Intl.NumberFormat("en-US", {
    style: currency ? "currency" : "decimal",
    currency: currency === "SAR" ? "SAR" : undefined,
    notation: compact ? "compact" : "standard",
    maximumFractionDigits: 2,
  });

  const formattedValue = formatter.format(value);

  if (currency === "%") {
    return `${formattedValue}%`;
  }

  return formattedValue;
}
```

```ts
type Config = {
  includeLeftSidebar: boolean;
  includeRightSidebar: boolean;
};

const config: Config = {
  includeLeftSidebar: process.env.NEXT_PUBLIC_INCLUDE_LEFT_SIDEBAR === "true",
  includeRightSidebar: process.env.NEXT_PUBLIC_INCLUDE_RIGHT_SIDEBAR === "true",
};

export default config;
```