```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,
  },
];
```

```
// config/market.ts

export const API_KEY = process.env.NEXT_PUBLIC_TWELVE_DATA_API_KEY || 'demo';
export const API_URL = 'https://api.twelvedata.com';

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



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;

// Trial symbol for testing: 4261 (Theeb Rent A Car Company)
export const SAUDI_SYMBOLS = [
  { symbol: '4261', name: 'Theeb Rent A Car' },
  { symbol: '1180', name: 'Al Rajhi Bank' },
  { symbol: '2222', name: 'Saudi Aramco' },
  { symbol: '2350', name: 'Saudi Telecom' },
  { symbol: '1211', name: "Ma'aden" }
];
```

```
// // hooks/useMarketSocket.ts

// import { useEffect, useRef, useState } from 'react';
// import { WS_URL, API_KEY, MarketQuote } from '@/app/config/market';

// export function useMarketSocket(symbols: string[]) {
//   const [quotes, setQuotes] = useState<Record<string, MarketQuote>>({});
//   const ws = useRef<WebSocket | null>(null);

//   useEffect(() => {
//     if (!symbols.length) return;

//     // Create WebSocket connection
//     ws.current = new WebSocket(`${WS_URL}?apikey=${API_KEY}`);

//     ws.current.onopen = () => {
//       console.log('WebSocket Connected');
//       if (ws.current?.readyState === WebSocket.OPEN) {
//         // Subscribe to symbols
//         ws.current.send(JSON.stringify({
//           action: 'subscribe',
//           params: {
//             symbols: symbols.join(',')
//           }
//         }));
//       }
//     };

//     ws.current.onmessage = (event) => {
//       const data = JSON.parse(event.data);
//       if (data.event === 'price') {
//         setQuotes(prev => ({
//           ...prev,
//           [data.symbol]: {
//             ...prev[data.symbol],
//             symbol: data.symbol,
//             name: prev[data.symbol]?.name || data.symbol,
//             price: parseFloat(data.price),
//             timestamp: data.timestamp,
//             change: parseFloat(data.change) || 0,
//             percentChange: parseFloat(data.percent_change) || 0,
//             volume: parseInt(data.volume) || 0
//           }
//         }));
//       }
//     };

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

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

//   return quotes;
// }
// 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_AP
I_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_KE
Y}`
      );

      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
  };
}
```

```
// 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
      }
    ]
  };
}
```

```
import {
  BedrockAgentRuntimeClient,
  RetrieveCommand,
  RetrieveCommandInput,
} from "@aws-sdk/client-bedrock-agent-runtime";
import { type ClassValue, clsx } from "clsx";
import { twMerge } from "tailwind-merge";

console.log("ð\237\224\221 Have AWS AccessKey?", !!process.env.BAWS_ACCESS_KEY_ID);
console.log("ð\237\224\221 Have AWS Secret?", !!process.env.BAWS_SECRET_ACCESS_KEY);

const bedrockClient = new BedrockAgentRuntimeClient({
  region: "us-east-1", // Make sure this matches your Bedrock region
  credentials: {
    accessKeyId: process.env.BAWS_ACCESS_KEY_ID!,
    secretAccessKey: process.env.BAWS_SECRET_ACCESS_KEY!,
  },
});

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

export interface RAGSource {
  id: string;
  fileName: string;
  snippet: string;
  score: number;
}

export async function retrieveContext(
  query: string,
  knowledgeBaseId: string,
  n: number = 3,
): Promise<{
  context: string;
  isRagWorking: boolean;
  ragSources: RAGSource[];
}> {
  try {
    if (!knowledgeBaseId) {
      console.error("knowledgeBaseId is not provided");
      return {
        context: "",
        isRagWorking: false,
        ragSources: [],
      };
    }

    const input: RetrieveCommandInput = {
      knowledgeBaseId: knowledgeBaseId,
      retrievalQuery: { text: query },
      retrievalConfiguration: {
        vectorSearchConfiguration: { numberOfResults: n },
      },
    };

    const command = new RetrieveCommand(input);
    const response = await bedrockClient.send(command);

    // Parse results
    const rawResults = response?.retrievalResults || [];
    const ragSources: RAGSource[] = rawResults
      .filter((res: any) => res.content && res.content.text)
      .map((result: any, index: number) => {
        const uri = result?.location?.s3Location?.uri || "";
        const fileName = uri.split("/").pop() || `Source-${index}.txt`;

        return {
          id:
            result.metadata?.["x-amz-bedrock-kb-chunk-id"] || `chunk-${index}`,
          fileName: fileName.replace(/_/g, " ").replace(".txt", ""),
          snippet: result.content?.text || "",
          score: result.score || 0,
        };
      })
      .slice(0, 1);

    console.log("ð\237\224\215 Parsed RAG Sources:", ragSources); // Debug log

    const context = rawResults
      .filter((res: any) => res.content && res.content.text)
      .map((res: any) => res.content.text)
      .join("\n\n");

    return {
      context,
      isRagWorking: true,
      ragSources,
    };
  } catch (error) {
    console.error("RAG Error:", error);
    return { context: "", isRagWorking: false, ragSources: [] };
  }
}
```

```
// 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_D
ATA_API_KEY}`
    );
    const data = await response.json();
    return data.values || [];
  } catch (error) {
    console.error('Error fetching time series:', error);
    return [];
  }
}
```

```
// // 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;
//   }
// }
// app/api/chat/route.ts
import Anthropic from "@anthropic-ai/sdk";
import { z } from "zod";
import crypto from "crypto";

const anthropic = new Anthropic({
  apiKey: process.env.ANTHROPIC_API_KEY,
  defaultHeaders: {
    "anthropic-beta": "pdfs-2024-09-25"  // Add beta header here
  }
});

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()),
  debug: z.object({
    context_used: z.boolean(),
  })
});

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

    let messageContent;
    if (pdfData) {
      messageContent = [{
        role: ’user’,
        content: [
          {
            type: ’document’,
            source: {
              type: ’base64’,
              media_type: ’application/pdf’,
              data: pdfData,
            },
          },
          {
            type: ’text’,
            text: latestMessage,
          },
        ],
      }];
    } else {
      messageContent = [{
        role: ’user’,
        content: [{
          type: ’text’,
          text: latestMessage
        }],
      }];
    }

    const response = await anthropic.messages.create({
      model: "claude-3-5-sonnet-20241022",
      max_tokens: 4096,
      messages: messageContent,
    });

    const responseData = {
      response: response.content[0].text,
      thinking: "Processing your request",
      user_mood: "curious",
      suggested_questions: [
        "Can you summarize the main points?",
        "What are the key findings?",
        "Could you explain that in more detail?"
      ],
      debug: { context_used: true }
    };

    const validatedResponse = responseSchema.parse(responseData);

    return new Response(JSON.stringify({
      id: crypto.randomUUID(),
      ...validatedResponse,
    }), {
      status: 200,
      headers: { "Content-Type": "application/json" }
    });

  } catch (error) {
    console.error("Error in chat handler:", error);
    return new Response(
      JSON.stringify({
        response: "An error occurred while processing your request. Please try again.",
        thinking: "Error in processing",
        user_mood: "neutral",
        suggested_questions: [],
        debug: { context_used: false }
      }),
      {
        status: 500,
        headers: { "Content-Type": "application/json" }
      }
    );
  }
}
```

```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%;
    }
}

@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" "s
h" "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 \$=" "$file" >> "$TE
MP_PS"
    done
done
```

```
# 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;
```

```
// 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%",
    },
    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: {
    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%",
    },
  },
};
```

```
import { type ClassValue, clsx } from "clsx";
import { twMerge } from "tailwind-merge";

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

```
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;
```