**🎨 FRONTEND DEVELOPMENT GUIDE FOR KURZORA**

**📋 DOCUMENT STATUS**

* **Status:** ✅ **MASTER FRONTEND IMPLEMENTATION GUIDE**
* **Version:** 1.0
* **Authority:** Single Source of Truth for Next.js Frontend Development
* **Target:** Complete multi-language trading platform frontend

**🎯 FRONTEND ARCHITECTURE OVERVIEW**

**Technology Stack**

* **Framework:** Next.js 14+ with App Router
* **Language:** TypeScript
* **Styling:** Tailwind CSS with RTL support
* **UI Components:** Custom components + shadcn/ui
* **State Management:** React Context + Zustand
* **Internationalization:** next-intl
* **Authentication:** Supabase Auth
* **Charts:** TradingView Lightweight Charts
* **Forms:** react-hook-form + zod validation

**Multi-Language Support**

* **English (EN):** Primary market, LTR layout
* **German (DE):** European market, LTR layout, BaFin compliance
* **Arabic (AR):** MENA market, RTL layout, Islamic finance compliance

**🏗️ PROJECT STRUCTURE SETUP**

**1. Initialize Next.js Project**

# Create Next.js project with TypeScript and Tailwind

npx create-next-app@latest kurzora-platform --typescript --tailwind --app

cd kurzora-platform

# Install core dependencies

npm install @supabase/supabase-js

npm install next-intl

npm install @stripe/stripe-js @stripe/react-stripe-js

npm install react-hook-form @hookform/resolvers zod

npm install zustand

npm install lucide-react

npm install framer-motion

npm install recharts

npm install date-fns

npm install clsx tailwind-merge

npm install @types/lodash lodash

# Install UI dependencies

npm install @radix-ui/react-dialog

npm install @radix-ui/react-dropdown-menu

npm install @radix-ui/react-select

npm install @radix-ui/react-tabs

npm install @radix-ui/react-toast

**2. Project Structure Implementation**

kurzora-platform/

├── app/

│ ├── [locale]/ # Internationalized routes

│ │ ├── layout.tsx # Locale-specific layout

│ │ ├── page.tsx # Landing page

│ │ ├── auth/

│ │ │ ├── signin/page.tsx # Sign in page

│ │ │ └── signup/page.tsx # Sign up page

│ │ ├── dashboard/

│ │ │ ├── page.tsx # Main dashboard

│ │ │ ├── signals/page.tsx # Signals page

│ │ │ ├── portfolio/page.tsx # Portfolio page

│ │ │ └── settings/page.tsx # Settings page

│ │ └── api/ # API routes

│ ├── globals.css # Global styles with RTL support

│ └── layout.tsx # Root layout

├── components/

│ ├── ui/ # Reusable UI components

│ ├── auth/ # Authentication components

│ ├── dashboard/ # Dashboard components

│ ├── signals/ # Signal-related components

│ ├── landing/ # Landing page components

│ └── layout/ # Layout components

├── lib/

│ ├── supabase.ts # Supabase client

│ ├── i18n.ts # Internationalization setup

│ ├── utils.ts # Utility functions

│ └── validation.ts # Form validation schemas

├── hooks/

│ ├── useAuth.ts # Authentication hook

│ ├── useTranslation.ts # Translation hook

│ └── useRTL.ts # RTL layout hook

├── contexts/

│ ├── AuthContext.tsx # Authentication context

│ └── RTLContext.tsx # RTL layout context

├── messages/ # Translation files

│ ├── en.json # English translations

│ ├── de.json # German translations

│ └── ar.json # Arabic translations

└── types/

└── index.ts # TypeScript type definitions

**🌍 INTERNATIONALIZATION SETUP**

**1. Next.js i18n Configuration**

// next.config.js

/\*\* @type {import('next').NextConfig} \*/

const nextConfig = {

experimental: {

appDir: true,

},

async redirects() {

return [

{

source: '/',

destination: '/en',

permanent: false,

},

]

},

}

module.exports = nextConfig

**2. i18n Library Setup**

// lib/i18n.ts

import { getRequestConfig } from 'next-intl/server'

import { notFound } from 'next/navigation'

export const locales = ['en', 'de', 'ar'] as const

export type Locale = typeof locales[number]

export default getRequestConfig(async ({ locale }) => {

// Validate that the incoming `locale` parameter is valid

if (!locales.includes(locale as Locale)) notFound()

return {

messages: (await import(`../messages/${locale}.json`)).default,

timeZone: getTimeZoneForLocale(locale),

formats: {

dateTime: {

short: {

day: 'numeric',

month: 'short',

year: 'numeric'

}

},

number: {

currency: {

style: 'currency',

currency: getCurrencyForLocale(locale)

}

}

}

}

})

function getTimeZoneForLocale(locale: string): string {

switch (locale) {

case 'de': return 'Europe/Berlin'

case 'ar': return 'Asia/Dubai'

default: return 'America/New\_York'

}

}

function getCurrencyForLocale(locale: string): string {

switch (locale) {

case 'de': return 'EUR'

case 'ar': return 'USD' // or 'AED' for UAE

default: return 'USD'

}

}

**3. Translation Files**

// messages/en.json

{

"metadata": {

"title": "Kurzora - Smart Trading Signals",

"description": "AI-powered trading signals for retail investors"

},

"navigation": {

"dashboard": "Dashboard",

"signals": "Signals",

"portfolio": "Portfolio",

"settings": "Settings",

"signOut": "Sign Out"

},

"auth": {

"signIn": "Sign In",

"signUp": "Sign Up",

"email": "Email",

"password": "Password",

"confirmPassword": "Confirm Password",

"name": "Full Name",

"forgotPassword": "Forgot Password?",

"createAccount": "Create Account",

"alreadyHaveAccount": "Already have an account?",

"dontHaveAccount": "Don't have an account?"

},

"landing": {

"heroTitle": "Smart Trading Signals for Retail Investors",

"heroSubtitle": "AI-powered analysis of 500+ stocks with clear 0-100 scoring",

"getStarted": "Get Started",

"learnMore": "Learn More",

"features": {

"transparent": "Transparent Scoring",

"ai": "AI Explanations",

"multiLanguage": "Multi-Language",

"paperTrading": "Paper Trading"

}

},

"dashboard": {

"title": "Dashboard",

"welcomeBack": "Welcome back",

"signalHeatmap": "Signal Heatmap",

"activeSignals": "Active Signals",

"portfolioValue": "Portfolio Value",

"todaysPnL": "Today's P&L",

"winRate": "Win Rate"

},

"signals": {

"title": "Trading Signals",

"score": "Score",

"type": "Type",

"strength": "Strength",

"entryPrice": "Entry Price",

"stopLoss": "Stop Loss",

"takeProfit": "Take Profit",

"confidence": "Confidence",

"bullish": "Bullish",

"bearish": "Bearish",

"strong": "Strong",

"moderate": "Moderate",

"weak": "Weak"

},

"common": {

"loading": "Loading...",

"error": "An error occurred",

"retry": "Retry",

"cancel": "Cancel",

"save": "Save",

"edit": "Edit",

"delete": "Delete",

"confirm": "Confirm",

"search": "Search",

"filter": "Filter",

"sort": "Sort",

"refresh": "Refresh"

}

}

// messages/de.json

{

"metadata": {

"title": "Kurzora - Intelligente Handelssignale",

"description": "KI-gestützte Handelssignale für Privatanleger"

},

"navigation": {

"dashboard": "Dashboard",

"signals": "Signale",

"portfolio": "Portfolio",

"settings": "Einstellungen",

"signOut": "Abmelden"

},

"auth": {

"signIn": "Anmelden",

"signUp": "Registrieren",

"email": "E-Mail",

"password": "Passwort",

"confirmPassword": "Passwort bestätigen",

"name": "Vollständiger Name",

"forgotPassword": "Passwort vergessen?",

"createAccount": "Konto erstellen",

"alreadyHaveAccount": "Haben Sie bereits ein Konto?",

"dontHaveAccount": "Sie haben noch kein Konto?"

},

"landing": {

"heroTitle": "Intelligente Handelssignale für Privatanleger",

"heroSubtitle": "KI-gestützte Analyse von 500+ Aktien mit klarer 0-100 Bewertung",

"getStarted": "Loslegen",

"learnMore": "Mehr erfahren",

"features": {

"transparent": "Transparente Bewertung",

"ai": "KI-Erklärungen",

"multiLanguage": "Mehrsprachig",

"paperTrading": "Papierhandel"

}

}

}

// messages/ar.json

{

"metadata": {

"title": "كورزورا - إشارات التداول الذكية",

"description": "إشارات التداول المدعومة بالذكاء الاصطناعي للمستثمرين الأفراد"

},

"navigation": {

"dashboard": "لوحة التحكم",

"signals": "الإشارات",

"portfolio": "المحفظة",

"settings": "الإعدادات",

"signOut": "تسجيل الخروج"

},

"auth": {

"signIn": "تسجيل الدخول",

"signUp": "إنشاء حساب",

"email": "البريد الإلكتروني",

"password": "كلمة المرور",

"confirmPassword": "تأكيد كلمة المرور",

"name": "الاسم الكامل",

"forgotPassword": "نسيت كلمة المرور؟",

"createAccount": "إنشاء حساب",

"alreadyHaveAccount": "لديك حساب بالفعل؟",

"dontHaveAccount": "ليس لديك حساب؟"

},

"landing": {

"heroTitle": "إشارات التداول الذكية للمستثمرين الأفراد",

"heroSubtitle": "تحليل مدعوم بالذكاء الاصطناعي لأكثر من 500 سهم مع تقييم واضح من 0-100",

"getStarted": "ابدأ الآن",

"learnMore": "اعرف المزيد",

"features": {

"transparent": "تقييم شفاف",

"ai": "شروحات الذكاء الاصطناعي",

"multiLanguage": "متعدد اللغات",

"paperTrading": "التداول الورقي"

}

}

}

**🔄 RTL LAYOUT SYSTEM**

**1. RTL Context and Provider**

// contexts/RTLContext.tsx

'use client'

import { createContext, useContext, ReactNode } from 'react'

import { useLocale } from 'next-intl'

interface RTLContextType {

dir: 'ltr' | 'rtl'

isRTL: boolean

locale: string

}

const RTLContext = createContext<RTLContextType | undefined>(undefined)

export function RTLProvider({ children }: { children: ReactNode }) {

const locale = useLocale()

const isRTL = locale === 'ar'

const dir = isRTL ? 'rtl' : 'ltr'

return (

<RTLContext.Provider value={{ dir, isRTL, locale }}>

<div dir={dir} className={`${isRTL ? 'rtl' : 'ltr'} font-${getFontFamily(locale)}`}>

{children}

</div>

</RTLContext.Provider>

)

}

export const useRTL = () => {

const context = useContext(RTLContext)

if (context === undefined) {

throw new Error('useRTL must be used within an RTLProvider')

}

return context

}

function getFontFamily(locale: string): string {

switch (locale) {

case 'ar': return 'arabic'

case 'de': return 'german'

default: return 'english'

}

}

**2. Tailwind RTL Configuration**

// tailwind.config.js

/\*\* @type {import('tailwindcss').Config} \*/

module.exports = {

content: [

'./pages/\*\*/\*.{js,ts,jsx,tsx,mdx}',

'./components/\*\*/\*.{js,ts,jsx,tsx,mdx}',

'./app/\*\*/\*.{js,ts,jsx,tsx,mdx}',

],

theme: {

extend: {

fontFamily: {

'english': ['Inter', 'system-ui', 'sans-serif'],

'german': ['Inter', 'system-ui', 'sans-serif'],

'arabic': ['Noto Sans Arabic', 'Arial', 'sans-serif'],

},

colors: {

// Kurzora brand colors

brand: {

50: '#eff6ff',

100: '#dbeafe',

500: '#3b82f6',

600: '#2563eb',

700: '#1d4ed8',

}

}

},

},

plugins: [

require('@tailwindcss/forms'),

require('@tailwindcss/typography'),

// RTL support plugin

function({ addUtilities, addComponents }) {

const newUtilities = {

'.rtl': {

direction: 'rtl',

},

'.ltr': {

direction: 'ltr',

},

// RTL-specific margin utilities

'.rtl .mr-auto': {

'margin-right': 'auto',

'margin-left': '0',

},

'.rtl .ml-auto': {

'margin-left': 'auto',

'margin-right': '0',

},

// RTL-specific text alignment

'.rtl .text-left': {

'text-align': 'right',

},

'.rtl .text-right': {

'text-align': 'left',

},

// RTL-specific flex utilities

'.rtl .flex-row': {

'flex-direction': 'row-reverse',

},

'.rtl .flex-row-reverse': {

'flex-direction': 'row',

},

// RTL-specific border utilities

'.rtl .border-l': {

'border-left': 'none',

'border-right': '1px solid',

},

'.rtl .border-r': {

'border-right': 'none',

'border-left': '1px solid',

},

}

addUtilities(newUtilities)

const newComponents = {

'.btn-rtl': {

'@apply px-4 py-2 rounded-lg transition-colors duration-200': {},

'&.rtl': {

'@apply flex-row-reverse': {},

}

}

}

addComponents(newComponents)

},

],

}

**3. RTL-Aware Components**

// components/ui/DirectionAwareComponent.tsx

'use client'

import { ReactNode } from 'react'

import { useRTL } from '@/contexts/RTLContext'

import { cn } from '@/lib/utils'

interface DirectionAwareProps {

children: ReactNode

className?: string

rtlClassName?: string

ltrClassName?: string

}

export function DirectionAware({

children,

className,

rtlClassName,

ltrClassName

}: DirectionAwareProps) {

const { isRTL } = useRTL()

const directionClass = isRTL ? rtlClassName : ltrClassName

return (

<div className={cn(className, directionClass)}>

{children}

</div>

)

}

// Usage example

export function NavigationExample() {

return (

<DirectionAware

className="flex items-center gap-4"

rtlClassName="flex-row-reverse"

ltrClassName="flex-row"

>

<span>Navigation Item 1</span>

<span>Navigation Item 2</span>

</DirectionAware>

)

}

**🔐 AUTHENTICATION IMPLEMENTATION**

**1. Supabase Client Setup**

// lib/supabase.ts

import { createClient } from '@supabase/supabase-js'

const supabaseUrl = process.env.NEXT\_PUBLIC\_SUPABASE\_URL!

const supabaseAnonKey = process.env.NEXT\_PUBLIC\_SUPABASE\_ANON\_KEY!

export const supabase = createClient(supabaseUrl, supabaseAnonKey)

// Database types (auto-generated or manual)

export interface Database {

public: {

Tables: {

users: {

Row: {

id: string

email: string

name: string

preferred\_language: string

subscription\_tier: 'starter' | 'professional' | 'elite'

created\_at: string

updated\_at: string

}

Insert: {

id?: string

email: string

name: string

preferred\_language?: string

subscription\_tier?: 'starter' | 'professional' | 'elite'

created\_at?: string

updated\_at?: string

}

Update: {

id?: string

email?: string

name?: string

preferred\_language?: string

subscription\_tier?: 'starter' | 'professional' | 'elite'

updated\_at?: string

}

}

// ... other tables

}

}

}

**2. Authentication Context**

// contexts/AuthContext.tsx

'use client'

import { createContext, useContext, useEffect, useState, ReactNode } from 'react'

import { User, Session } from '@supabase/supabase-js'

import { supabase, Database } from '@/lib/supabase'

import { useRouter } from 'next/navigation'

import { useLocale } from 'next-intl'

interface AuthContextType {

user: User | null

session: Session | null

userProfile: Database['public']['Tables']['users']['Row'] | null

loading: boolean

signIn: (email: string, password: string) => Promise<{ error: any }>

signUp: (email: string, password: string, name: string) => Promise<{ error: any }>

signOut: () => Promise<void>

updateProfile: (updates: Partial<Database['public']['Tables']['users']['Update']>) => Promise<{ error: any }>

}

const AuthContext = createContext<AuthContextType | undefined>(undefined)

export function AuthProvider({ children }: { children: ReactNode }) {

const [user, setUser] = useState<User | null>(null)

const [session, setSession] = useState<Session | null>(null)

const [userProfile, setUserProfile] = useState<Database['public']['Tables']['users']['Row'] | null>(null)

const [loading, setLoading] = useState(true)

const router = useRouter()

const locale = useLocale()

useEffect(() => {

// Get initial session

const getInitialSession = async () => {

const { data: { session }, error } = await supabase.auth.getSession()

if (session) {

setSession(session)

setUser(session.user)

await fetchUserProfile(session.user.id)

}

setLoading(false)

}

getInitialSession()

// Listen for auth changes

const { data: { subscription } } = supabase.auth.onAuthStateChange(

async (event, session) => {

setSession(session)

setUser(session?.user ?? null)

if (session?.user) {

await fetchUserProfile(session.user.id)

} else {

setUserProfile(null)

}

setLoading(false)

}

)

return () => subscription.unsubscribe()

}, [])

const fetchUserProfile = async (userId: string) => {

try {

const { data, error } = await supabase

.from('users')

.select('\*')

.eq('id', userId)

.single()

if (error) throw error

setUserProfile(data)

} catch (error) {

console.error('Error fetching user profile:', error)

}

}

const signIn = async (email: string, password: string) => {

const { error } = await supabase.auth.signInWithPassword({

email,

password,

})

return { error }

}

const signUp = async (email: string, password: string, name: string) => {

const { data, error } = await supabase.auth.signUp({

email,

password,

options: {

data: {

name,

preferred\_language: locale,

}

}

})

if (data.user && !error) {

// Create user profile

const { error: profileError } = await supabase

.from('users')

.insert([

{

id: data.user.id,

email,

name,

preferred\_language: locale,

}

])

if (profileError) {

console.error('Error creating user profile:', profileError)

}

}

return { error }

}

const signOut = async () => {

await supabase.auth.signOut()

router.push(`/${locale}/auth/signin`)

}

const updateProfile = async (updates: Partial<Database['public']['Tables']['users']['Update']>) => {

if (!user) return { error: new Error('No user logged in') }

const { error } = await supabase

.from('users')

.update({ ...updates, updated\_at: new Date().toISOString() })

.eq('id', user.id)

if (!error) {

// Refresh user profile

await fetchUserProfile(user.id)

}

return { error }

}

const value = {

user,

session,

userProfile,

loading,

signIn,

signUp,

signOut,

updateProfile,

}

return <AuthContext.Provider value={value}>{children}</AuthContext.Provider>

}

export const useAuth = () => {

const context = useContext(AuthContext)

if (context === undefined) {

throw new Error('useAuth must be used within an AuthProvider')

}

return context

}

**3. Protected Route Component**

// components/auth/ProtectedRoute.tsx

'use client'

import { useEffect } from 'react'

import { useAuth } from '@/contexts/AuthContext'

import { useRouter } from 'next/navigation'

import { useLocale } from 'next-intl'

interface ProtectedRouteProps {

children: React.ReactNode

requireAuth?: boolean

redirectTo?: string

}

export function ProtectedRoute({

children,

requireAuth = true,

redirectTo

}: ProtectedRouteProps) {

const { user, loading } = useAuth()

const router = useRouter()

const locale = useLocale()

useEffect(() => {

if (!loading && requireAuth && !user) {

const redirect = redirectTo || `/${locale}/auth/signin`

router.push(redirect)

}

}, [user, loading, requireAuth, router, locale, redirectTo])

if (loading) {

return (

<div className="min-h-screen flex items-center justify-center">

<div className="animate-spin rounded-full h-32 w-32 border-b-2 border-brand-600"></div>

</div>

)

}

if (requireAuth && !user) {

return null

}

return <>{children}</>

}

**📱 COMPONENT LIBRARY**

**1. Landing Page Components**

// components/landing/HeroSection.tsx

'use client'

import { useTranslations } from 'next-intl'

import { useRTL } from '@/contexts/RTLContext'

import { Button } from '@/components/ui/Button'

import { ArrowRight, TrendingUp, Shield, Globe } from 'lucide-react'

import { motion } from 'framer-motion'

export function HeroSection() {

const t = useTranslations('landing')

const { isRTL } = useRTL()

const stats = [

{ label: t('stats.signals\_generated'), value: '10,000+' },

{ label: t('stats.win\_rate'), value: '73%' },

{ label: t('stats.active\_users'), value: '2,500+' },

{ label: t('stats.countries'), value: '25+' },

]

return (

<section className="relative overflow-hidden bg-gradient-to-br from-brand-50 to-white py-20">

<div className="container mx-auto px-4">

<div className="grid lg:grid-cols-2 gap-12 items-center">

{/\* Content \*/}

<motion.div

initial={{ opacity: 0, y: 20 }}

animate={{ opacity: 1, y: 0 }}

transition={{ duration: 0.6 }}

className={isRTL ? 'text-right' : 'text-left'}

>

<h1 className="text-5xl lg:text-6xl font-bold text-gray-900 mb-6">

{t('hero\_title')}

</h1>

<p className="text-xl text-gray-600 mb-8 leading-relaxed">

{t('hero\_subtitle')}

</p>

<div className={`flex gap-4 mb-12 ${isRTL ? 'flex-row-reverse' : ''}`}>

<Button size="lg" className="bg-brand-600 hover:bg-brand-700">

{t('get\_started')}

<ArrowRight className={`ml-2 h-5 w-5 ${isRTL ? 'rotate-180' : ''}`} />

</Button>

<Button variant="outline" size="lg">

{t('watch\_demo')}

</Button>

</div>

{/\* Stats \*/}

<div className="grid grid-cols-2 md:grid-cols-4 gap-6">

{stats.map((stat, index) => (

<motion.div

key={stat.label}

initial={{ opacity: 0, y: 20 }}

animate={{ opacity: 1, y: 0 }}

transition={{ duration: 0.6, delay: 0.2 + index \* 0.1 }}

className="text-center"

>

<div className="text-3xl font-bold text-brand-600 mb-1">

{stat.value}

</div>

<div className="text-sm text-gray-600">{stat.label}</div>

</motion.div>

))}

</div>

</motion.div>

{/\* Visual \*/}

<motion.div

initial={{ opacity: 0, scale: 0.8 }}

animate={{ opacity: 1, scale: 1 }}

transition={{ duration: 0.8 }}

className="relative"

>

<div className="bg-white rounded-2xl shadow-2xl p-8">

<div className="space-y-4">

<div className="flex items-center justify-between">

<h3 className="font-semibold text-gray-900">{t('live\_signals')}</h3>

<span className="bg-green-100 text-green-800 text-xs px-2 py-1 rounded-full">

{t('live')}

</span>

</div>

{/\* Mock signal cards \*/}

{['AAPL', 'MSFT', 'GOOGL'].map((ticker, index) => (

<div key={ticker} className="flex items-center justify-between p-3 bg-gray-50 rounded-lg">

<div>

<div className="font-medium">{ticker}</div>

<div className="text-sm text-gray-600">{t('bullish\_signal')}</div>

</div>

<div className="text-right">

<div className="font-bold text-green-600">

{95 - index \* 5}/100

</div>

<div className="text-xs text-gray-500">{t('score')}</div>

</div>

</div>

))}

</div>

</div>

</motion.div>

</div>

</div>

</section>

)

}

// components/landing/FeaturesSection.tsx

'use client'

import { useTranslations } from 'next-intl'

import { TrendingUp, Brain, Globe2, Shield, Smartphone, BarChart3 } from 'lucide-react'

import { motion } from 'framer-motion'

export function FeaturesSection() {

const t = useTranslations('features')

const features = [

{

icon: TrendingUp,

title: t('transparent\_scoring.title'),

description: t('transparent\_scoring.description'),

},

{

icon: Brain,

title: t('ai\_explanations.title'),

description: t('ai\_explanations.description'),

},

{

icon: Globe2,

title: t('multi\_language.title'),

description: t('multi\_language.description'),

},

{

icon: Shield,

title: t('islamic\_compliant.title'),

description: t('islamic\_compliant.description'),

},

{

icon: Smartphone,

title: t('paper\_trading.title'),

description: t('paper\_trading.description'),

},

{

icon: BarChart3,

title: t('real\_time.title'),

description: t('real\_time.description'),

},

]

return (

<section className="py-20 bg-gray-50">

<div className="container mx-auto px-4">

<div className="text-center mb-16">

<h2 className="text-4xl font-bold text-gray-900 mb-4">

{t('section\_title')}

</h2>

<p className="text-xl text-gray-600 max-w-3xl mx-auto">

{t('section\_subtitle')}

</p>

</div>

<div className="grid md:grid-cols-2 lg:grid-cols-3 gap-8">

{features.map((feature, index) => (

<motion.div

key={feature.title}

initial={{ opacity: 0, y: 20 }}

whileInView={{ opacity: 1, y: 0 }}

transition={{ duration: 0.6, delay: index \* 0.1 }}

viewport={{ once: true }}

className="bg-white rounded-xl p-6 shadow-lg hover:shadow-xl transition-shadow duration-300"

>

<div className="bg-brand-100 w-12 h-12 rounded-lg flex items-center justify-center mb-4">

<feature.icon className="h-6 w-6 text-brand-600" />

</div>

<h3 className="text-xl font-semibold text-gray-900 mb-2">

{feature.title}

</h3>

<p className="text-gray-600">

{feature.description}

</p>

</motion.div>

))}

</div>

</div>

</section>

)

}

**2. Dashboard Components**

// components/dashboard/SignalHeatmap.tsx

'use client'

import { useState, useEffect } from 'react'

import { useTranslations } from 'next-intl'

import { supabase } from '@/lib/supabase'

import { Card } from '@/components/ui/Card'

import { Badge } from '@/components/ui/Badge'

import { Button } from '@/components/ui/Button'

import { TrendingUp, TrendingDown, Filter, Search } from 'lucide-react'

interface Signal {

id: string

ticker: string

signal\_type: 'bullish' | 'bearish'

final\_score: number

strength: 'weak' | 'moderate' | 'strong'

entry\_price: number

stop\_loss: number

take\_profit: number

confidence: number

created\_at: string

}

export function SignalHeatmap() {

const [signals, setSignals] = useState<Signal[]>([])

const [loading, setLoading] = useState(true)

const [filter, setFilter] = useState<'all' | 'bullish' | 'bearish'>('all')

const [searchTerm, setSearchTerm] = useState('')

const t = useTranslations('signals')

useEffect(() => {

fetchSignals()

// Set up real-time subscription

const subscription = supabase

.channel('trading\_signals')

.on('postgres\_changes', {

event: '\*',

schema: 'public',

table: 'trading\_signals'

}, handleSignalUpdate)

.subscribe()

return () => {

subscription.unsubscribe()

}

}, [])

const fetchSignals = async () => {

try {

let query = supabase

.from('trading\_signals')

.select('\*')

.gte('final\_score', 80)

.order('final\_score', { ascending: false })

.limit(50)

const { data, error } = await query

if (error) throw error

setSignals(data || [])

} catch (error) {

console.error('Error fetching signals:', error)

} finally {

setLoading(false)

}

}

const handleSignalUpdate = (payload: any) => {

if (payload.eventType === 'INSERT') {

setSignals(prev => [payload.new, ...prev])

} else if (payload.eventType === 'UPDATE') {

setSignals(prev => prev.map(signal =>

signal.id === payload.new.id ? payload.new : signal

))

}

}

const filteredSignals = signals.filter(signal => {

const matchesFilter = filter === 'all' || signal.signal\_type === filter

const matchesSearch = signal.ticker.toLowerCase().includes(searchTerm.toLowerCase())

return matchesFilter && matchesSearch

})

const getScoreColor = (score: number) => {

if (score >= 90) return 'bg-green-500'

if (score >= 85) return 'bg-green-400'

if (score >= 80) return 'bg-yellow-400'

return 'bg-gray-400'

}

const getSignalIcon = (type: 'bullish' | 'bearish') => {

return type === 'bullish' ? TrendingUp : TrendingDown

}

if (loading) {

return (

<Card className="p-6">

<div className="animate-pulse space-y-4">

<div className="h-4 bg-gray-200 rounded w-1/4"></div>

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4 gap-4">

{[...Array(12)].map((\_, i) => (

<div key={i} className="h-32 bg-gray-200 rounded"></div>

))}

</div>

</div>

</Card>

)

}

return (

<Card className="p-6">

<div className="flex flex-col sm:flex-row justify-between items-start sm:items-center mb-6 gap-4">

<div>

<h2 className="text-2xl font-bold text-gray-900 mb-1">

{t('signal\_heatmap')}

</h2>

<p className="text-gray-600">

{filteredSignals.length} {t('active\_signals')}

</p>

</div>

<div className="flex gap-2">

{/\* Search \*/}

<div className="relative">

<Search className="absolute left-3 top-1/2 transform -translate-y-1/2 h-4 w-4 text-gray-400" />

<input

type="text"

placeholder={t('search\_ticker')}

value={searchTerm}

onChange={(e) => setSearchTerm(e.target.value)}

className="pl-10 pr-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-brand-500 focus:border-transparent"

/>

</div>

{/\* Filter \*/}

<select

value={filter}

onChange={(e) => setFilter(e.target.value as 'all' | 'bullish' | 'bearish')}

className="px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-brand-500 focus:border-transparent"

>

<option value="all">{t('all\_signals')}</option>

<option value="bullish">{t('bullish')}</option>

<option value="bearish">{t('bearish')}</option>

</select>

</div>

</div>

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4 gap-4">

{filteredSignals.map((signal) => {

const SignalIcon = getSignalIcon(signal.signal\_type)

return (

<div

key={signal.id}

className={`p-4 rounded-lg border-2 transition-all duration-200 hover:shadow-lg cursor-pointer ${

signal.signal\_type === 'bullish'

? 'border-green-200 bg-green-50 hover:border-green-300'

: 'border-red-200 bg-red-50 hover:border-red-300'

}`}

>

<div className="flex justify-between items-start mb-3">

<div>

<h3 className="font-bold text-lg text-gray-900">{signal.ticker}</h3>

<div className="flex items-center gap-1 mt-1">

<SignalIcon className={`h-4 w-4 ${

signal.signal\_type === 'bullish' ? 'text-green-600' : 'text-red-600'

}`} />

<span className={`text-sm font-medium ${

signal.signal\_type === 'bullish' ? 'text-green-600' : 'text-red-600'

}`}>

{t(signal.signal\_type)}

</span>

</div>

</div>

<div className="text-right">

<div className={`text-lg font-bold px-2 py-1 rounded text-white ${getScoreColor(signal.final\_score)}`}>

{signal.final\_score}

</div>

<div className="text-xs text-gray-500 mt-1">{t('score')}</div>

</div>

</div>

<div className="space-y-2 text-sm">

<div className="flex justify-between">

<span className="text-gray-600">{t('entry')}:</span>

<span className="font-medium">${signal.entry\_price}</span>

</div>

<div className="flex justify-between">

<span className="text-gray-600">{t('stop\_loss')}:</span>

<span className="font-medium">${signal.stop\_loss}</span>

</div>

<div className="flex justify-between">

<span className="text-gray-600">{t('target')}:</span>

<span className="font-medium">${signal.take\_profit}</span>

</div>

<div className="flex justify-between items-center pt-2">

<span className="text-gray-600">{t('strength')}:</span>

<Badge variant={signal.strength === 'strong' ? 'default' : 'secondary'}>

{t(signal.strength)}

</Badge>

</div>

</div>

</div>

)

})}

</div>

{filteredSignals.length === 0 && (

<div className="text-center py-12">

<div className="text-gray-400 mb-2">

<TrendingUp className="h-12 w-12 mx-auto" />

</div>

<h3 className="text-lg font-medium text-gray-900 mb-1">

{t('no\_signals\_found')}

</h3>

<p className="text-gray-600">

{t('no\_signals\_description')}

</p>

</div>

)}

</Card>

)

}

**🎨 STYLING AND THEMES**

**1. Global Styles with RTL Support**

/\* app/globals.css \*/

@tailwind base;

@tailwind components;

@tailwind utilities;

/\* Arabic Font Import \*/

@import url('https://fonts.googleapis.com/css2?family=Noto+Sans+Arabic:wght@400;500;600;700&display=swap');

/\* Base Styles \*/

@layer base {

\* {

@apply border-border;

}

body {

@apply bg-background text-foreground;

}

/\* RTL-specific styles \*/

.rtl {

direction: rtl;

}

.ltr {

direction: ltr;

}

/\* Arabic font for RTL \*/

.font-arabic {

font-family: 'Noto Sans Arabic', Arial, sans-serif;

}

/\* Custom scrollbar for RTL \*/

.rtl ::-webkit-scrollbar {

width: 8px;

}

.rtl ::-webkit-scrollbar-track {

@apply bg-gray-100;

}

.rtl ::-webkit-scrollbar-thumb {

@apply bg-gray-300 rounded;

}

}

/\* Component Styles \*/

@layer components {

.btn-rtl {

@apply px-4 py-2 rounded-lg transition-colors duration-200;

}

.rtl .btn-rtl {

@apply flex-row-reverse;

}

.input-rtl {

@apply w-full px-3 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-brand-500 focus:border-transparent;

}

.rtl .input-rtl {

text-align: right;

}

/\* Signal card animations \*/

.signal-card {

@apply transform transition-all duration-200 hover:scale-105 hover:shadow-lg;

}

/\* Loading states \*/

.loading-shimmer {

@apply animate-pulse bg-gradient-to-r from-gray-200 via-gray-300 to-gray-200 bg-size-200 bg-pos-0;

animation: shimmer 1.5s infinite;

}

@keyframes shimmer {

0% { background-position: -200% 0; }

100% { background-position: 200% 0; }

}

}

/\* Utility Classes \*/

@layer utilities {

.text-balance {

text-wrap: balance;

}

.bg-size-200 {

background-size: 200% 200%;

}

.bg-pos-0 {

background-position: 0% 0%;

}

/\* RTL-specific utilities \*/

.rtl .ml-auto {

margin-left: 0;

margin-right: auto;

}

.rtl .mr-auto {

margin-right: 0;

margin-left: auto;

}

.rtl .text-left {

text-align: right;

}

.rtl .text-right {

text-align: left;

}

.rtl .border-l {

border-left: none;

border-right: 1px solid;

}

.rtl .border-r {

border-right: none;

border-left: 1px solid;

}

.rtl .rounded-l-lg {

border-radius: 0 0.5rem 0.5rem 0;

}

.rtl .rounded-r-lg {

border-radius: 0.5rem 0 0 0.5rem;

}

}

**📊 PERFORMANCE OPTIMIZATION**

**1. Code Splitting and Lazy Loading**

// lib/lazy-imports.ts

import dynamic from 'next/dynamic'

// Lazy load heavy components

export const SignalHeatmap = dynamic(

() => import('@/components/dashboard/SignalHeatmap').then(mod => mod.SignalHeatmap),

{

loading: () => <div className="loading-shimmer h-64 rounded-lg" />,

ssr: false

}

)

export const TradingChart = dynamic(

() => import('@/components/charts/TradingChart').then(mod => mod.TradingChart),

{

loading: () => <div className="loading-shimmer h-96 rounded-lg" />,

ssr: false

}

)

export const PortfolioAnalytics = dynamic(

() => import('@/components/portfolio/PortfolioAnalytics').then(mod => mod.PortfolioAnalytics),

{

loading: () => <div className="loading-shimmer h-48 rounded-lg" />,

ssr: false

}

)

**2. Image Optimization**

// components/ui/OptimizedImage.tsx

import Image from 'next/image'

import { useState } from 'react'

interface OptimizedImageProps {

src: string

alt: string

width: number

height: number

className?: string

priority?: boolean

}

export function OptimizedImage({

src,

alt,

width,

height,

className,

priority = false

}: OptimizedImageProps) {

const [isLoading, setIsLoading] = useState(true)

return (

<div className={`relative overflow-hidden ${className}`}>

<Image

src={src}

alt={alt}

width={width}

height={height}

priority={priority}

className={`transition-opacity duration-300 ${

isLoading ? 'opacity-0' : 'opacity-100'

}`}

onLoad={() => setIsLoading(false)}

sizes="(max-width: 768px) 100vw, (max-width: 1200px) 50vw, 33vw"

/>

{isLoading && (

<div className="absolute inset-0 loading-shimmer" />

)}

</div>

)

}

**✅ TESTING STRATEGY**

**1. Component Testing Setup**

// \_\_tests\_\_/components/SignalHeatmap.test.tsx

import { render, screen, waitFor } from '@testing-library/react'

import { SignalHeatmap } from '@/components/dashboard/SignalHeatmap'

import { NextIntlClientProvider } from 'next-intl'

import { AuthProvider } from '@/contexts/AuthContext'

const messages = {

signals: {

signal\_heatmap: 'Signal Heatmap',

active\_signals: 'active signals',

bullish: 'Bullish',

bearish: 'Bearish',

}

}

const TestWrapper = ({ children }: { children: React.ReactNode }) => (

<NextIntlClientProvider locale="en" messages={messages}>

<AuthProvider>

{children}

</AuthProvider>

</NextIntlClientProvider>

)

describe('SignalHeatmap', () => {

it('renders signal heatmap with loading state', () => {

render(

<TestWrapper>

<SignalHeatmap />

</TestWrapper>

)

expect(screen.getByText('Signal Heatmap')).toBeInTheDocument()

})

it('displays signals when loaded', async () => {

// Mock Supabase client

jest.mock('@/lib/supabase', () => ({

supabase: {

from: jest.fn(() => ({

select: jest.fn(() => ({

gte: jest.fn(() => ({

order: jest.fn(() => ({

limit: jest.fn(() => Promise.resolve({

data: [

{

id: '1',

ticker: 'AAPL',

signal\_type: 'bullish',

final\_score: 95,

strength: 'strong'

}

],

error: null

}))

}))

}))

}))

})),

channel: jest.fn(() => ({

on: jest.fn(() => ({

subscribe: jest.fn()

}))

}))

}

}))

render(

<TestWrapper>

<SignalHeatmap />

</TestWrapper>

)

await waitFor(() => {

expect(screen.getByText('AAPL')).toBeInTheDocument()

})

})

})

**🚀 DEPLOYMENT PREPARATION**

**1. Build Optimization**

// next.config.js

/\*\* @type {import('next').NextConfig} \*/

const nextConfig = {

experimental: {

appDir: true,

},

images: {

domains: ['your-cdn-domain.com'],

formats: ['image/webp', 'image/avif'],

},

async headers() {

return [

{

source: '/(.\*)',

headers: [

{

key: 'X-Frame-Options',

value: 'DENY',

},

{

key: 'X-Content-Type-Options',

value: 'nosniff',

},

{

key: 'Referrer-Policy',

value: 'origin-when-cross-origin',

},

],

},

]

},

async redirects() {

return [

{

source: '/',

destination: '/en',

permanent: false,

},

]

},

webpack: (config, { dev, isServer }) => {

// Production optimizations

if (!dev && !isServer) {

config.optimization.splitChunks.chunks = 'all'

}

return config

},

}

module.exports = nextConfig

**2. Environment Configuration**

# .env.local

NEXT\_PUBLIC\_SUPABASE\_URL=your\_supabase\_url

NEXT\_PUBLIC\_SUPABASE\_ANON\_KEY=your\_supabase\_anon\_key

SUPABASE\_SERVICE\_ROLE\_KEY=your\_service\_role\_key

NEXT\_PUBLIC\_STRIPE\_PUBLISHABLE\_KEY=your\_stripe\_publishable\_key

STRIPE\_SECRET\_KEY=your\_stripe\_secret\_key

NEXT\_PUBLIC\_APP\_URL=https://kurzora.com

NEXT\_PUBLIC\_API\_URL=https://kurzora.com/api

# Development only

NEXT\_PUBLIC\_POLYGON\_API\_KEY=your\_polygon\_api\_key\_dev

# Analytics

NEXT\_PUBLIC\_GA\_TRACKING\_ID=your\_ga\_tracking\_id

**🎨 FRONTEND IMPLEMENTATION COMPLETE**

**This guide provides everything needed to build Kurzora's multi-language frontend with:**

* ✅ Complete internationalization (EN/DE/AR)
* ✅ RTL layout support for Arabic
* ✅ Authentication with Supabase
* ✅ Real-time signal heatmap
* ✅ Performance optimization
* ✅ Testing framework
* ✅ Production deployment preparation

**Ready for backend development next!**