**🤖 CURSOR INTEGRATION INSTRUCTIONS FOR KURZORA**

**📋 DOCUMENT STATUS**

* **Status:** ✅ **MASTER AI DEVELOPMENT GUIDE**
* **Version:** 1.0
* **Authority:** Single Source of Truth for Cursor AI Development
* **Target:** Efficient development using 60+ project documents

**🎯 CURSOR SETUP & OPTIMIZATION**

**Initial Cursor Configuration**

1. **Install Cursor Extensions:**

- TypeScript and JavaScript

- Tailwind CSS IntelliSense

- ES7+ React/Redux/React-Native snippets

- Auto Rename Tag

- GitLens

- Prettier - Code formatter

1. **Cursor Settings Configuration:**

{

"cursor.ai.model": "claude-3.5-sonnet",

"cursor.ai.maxTokens": 8000,

"cursor.ai.temperature": 0.3,

"cursor.ai.enableCodeCompletion": true,

"cursor.ai.enableChatMode": true,

"cursor.ai.contextWindow": "large"

}

1. **Project-Specific Settings:**

// .cursor/settings.json

{

"cursor.ai.rules": [

"Always reference Master API Endpoints.docx for API development",

"Follow i18n Architecture.docx for multi-language implementation",

"Use RTL Layout System.docx for Arabic support",

"Reference Master DB Schema.docx for database queries",

"Follow Complete Authentication System.docx for auth implementation"

]

}

**📁 DOCUMENT HIERARCHY USAGE**

**Master Documents (Always Reference First):**

📁 00\_MASTER/

├── 📄 Master DB Schema.docx → Database operations

├── 📄 Master API Endpoints.docx → All API development

├── 📄 Complete Authentication System.docx → Auth implementation

└── 📄 Complete Project File Structure.docx → Project organization

**Implementation Documents (Phase-Specific):**

📁 04\_IMPLEMENTATION/

├── 📄 Development Step-by-Step Guide.docx → Current roadmap

├── 📄 Landing Page.docx → Homepage development

├── 📄 Dashboard.docx → Dashboard implementation

├── 📄 Signals.docx → Signal components

├── 📄 Sign In Page.docx → Authentication UI

└── 📄 Settings.docx → User settings

**Technical Architecture (Reference as Needed):**

📁 02\_TECHNICAL/

├── 📄 i18n Architecture.docx → Multi-language setup

├── 📄 RTL Layout System.docx → Arabic/RTL support

├── 📄 Backend Architecture Analysis.docx → Backend implementation

└── 📄 Islamic Finance Compliance.docx → Religious requirements

**🚀 CURSOR PROMPT TEMPLATES**

**1. Component Development Prompts**

**Landing Page Component:**

Create a React component for the Kurzora landing page hero section following these specifications:

REFERENCE DOCUMENTS:

- Landing Page.docx (primary specifications)

- i18n Architecture.docx (multi-language support)

- RTL Layout System.docx (Arabic RTL support)

REQUIREMENTS:

- Multi-language support (EN/DE/AR) using next-intl

- RTL layout support for Arabic

- Responsive design with Tailwind CSS

- Hero section with animated statistics

- CTA buttons for sign-up/demo

- Follow the exact design specifications in Landing Page.docx

IMPLEMENTATION NOTES:

- Use TypeScript with proper interfaces

- Include loading states and error handling

- Ensure accessibility compliance

- Follow the component structure from Landing Page.docx

**Dashboard Component:**

Create the main dashboard component for Kurzora following these specifications:

REFERENCE DOCUMENTS:

- Dashboard.docx (primary specifications)

- Signals.docx (signal heatmap implementation)

- Master API Endpoints.docx (API integration)

- Complete Authentication System.docx (user context)

REQUIREMENTS:

- Signal heatmap for 500 S&P stocks (Phase 1)

- Real-time signal updates using Supabase subscriptions

- Multi-language support with RTL for Arabic

- Filter and search functionality

- Performance metrics display

- Paper trading portfolio summary

API INTEGRATION:

- Use GET /api/v1/signals/active for signal data

- Follow authentication patterns from Complete Authentication System.docx

- Implement proper error handling and loading states

**Authentication Forms:**

Create sign-in and sign-up forms following these specifications:

REFERENCE DOCUMENTS:

- Sign In Page.docx (UI specifications)

- Complete Authentication System.docx (auth logic)

- i18n Architecture.docx (multi-language forms)

REQUIREMENTS:

- Supabase Auth integration

- Form validation using react-hook-form + zod

- Multi-language error messages

- RTL support for Arabic

- Password strength validation

- Email verification flow

- Social login options (Google, Apple)

AUTHENTICATION FLOW:

- Follow the exact auth patterns from Complete Authentication System.docx

- Implement proper error handling

- Include loading states

- Handle authentication redirects

**2. API Development Prompts**

**Signal Processing API:**

Create API endpoints for signal processing following these specifications:

REFERENCE DOCUMENTS:

- Master API Endpoints.docx (endpoint specifications)

- Financial Data & Signal Processing.docx (signal algorithms)

- Backend Architecture Analysis.docx (implementation patterns)

REQUIREMENTS:

- Implement GET /api/v1/signals/active endpoint

- Signal scoring algorithm (0-100 points)

- Multi-timeframe analysis (1H, 4H, 1D, 1W)

- 5 core indicators: RSI, MACD, EMA, Volume, Bollinger Bands

- Filter signals ≥80 score threshold

- Rate limiting and authentication

- Error handling and logging

DATABASE INTEGRATION:

- Use Master DB Schema.docx for table structure

- Implement proper RLS policies

- Follow Supabase best practices

- Include proper indexing for performance

**User Management API:**

Create user management endpoints following these specifications:

REFERENCE DOCUMENTS:

- Master API Endpoints.docx (API structure)

- Complete Authentication System.docx (auth patterns)

- Settings.docx (user preferences)

REQUIREMENTS:

- User profile management

- Language preference storage

- Notification settings

- Subscription management

- Paper trading portfolio

- Islamic finance compliance preferences

IMPLEMENTATION:

- Follow REST patterns from Master API Endpoints.docx

- Include proper validation

- Multi-language support for user data

- Audit logging for security

**3. Database Query Prompts**

**Multi-language Database Queries:**

Create database queries for multi-language content following these specifications:

REFERENCE DOCUMENTS:

- Master DB Schema.docx (table structure)

- i18n Architecture.docx (translation patterns)

REQUIREMENTS:

- Query content with fallback to English

- Support for EN/DE/AR languages

- Efficient joins with translation tables

- Proper indexing for performance

- RLS policy compliance

EXAMPLE TABLES:

- content\_translations table for multi-language content

- users table with language preferences

- stock\_universe with translated company names

**4. Styling and Layout Prompts**

**RTL Layout Implementation:**

Create RTL-aware components following these specifications:

REFERENCE DOCUMENTS:

- RTL Layout System.docx (RTL patterns)

- i18n Architecture.docx (language detection)

REQUIREMENTS:

- Automatic RTL detection for Arabic

- Tailwind CSS RTL utilities

- Icon and text direction adjustments

- Layout flow modifications

- Arabic font integration

- Performance optimization for direction switching

IMPLEMENTATION:

- Use RTL context from RTL Layout System.docx

- Follow Tailwind RTL patterns

- Ensure proper text alignment

- Handle complex layouts (charts, tables, forms)

**🔍 CURSOR CHAT MODES**

**Quick Development Questions:**

@cursor How do I implement the signal heatmap component according to Dashboard.docx specifications with multi-language support?

**Code Review and Optimization:**

@cursor Review this authentication component against the specifications in Complete Authentication System.docx and suggest improvements for multi-language support.

**Bug Fixing:**

@cursor This RTL layout isn't working correctly for Arabic. Check against RTL Layout System.docx and fix the Tailwind classes.

**Feature Implementation:**

@cursor Implement the paper trading functionality following the specifications in Paper Trading.docx with proper API integration from Master API Endpoints.docx.

**⚡ CURSOR KEYBOARD SHORTCUTS**

**Essential Shortcuts for Kurzora Development:**

Ctrl+Shift+L → Multi-cursor selection

Ctrl+D → Select next occurrence

Ctrl+Shift+P → Command palette

Ctrl+` → Toggle terminal

Ctrl+Shift+E → Explorer

Ctrl+Shift+F → Search across files

Ctrl+G → Go to line

Ctrl+P → Quick file open

**AI-Specific Shortcuts:**

Ctrl+K → Cursor AI chat

Ctrl+Shift+K → Cursor AI code generation

Alt+\ → Inline AI suggestions

Ctrl+I → AI-powered refactoring

**📊 CURSOR WORKFLOW OPTIMIZATION**

**1. Development Phase Workflow**

**Starting a New Feature:**

1. **Reference Documents:** Check relevant documents in hierarchy
2. **Prompt Template:** Use appropriate template from this guide
3. **Context Setting:** Include specific requirements and constraints
4. **Code Generation:** Let Cursor generate initial implementation
5. **Review & Refine:** Check against specifications and refine

**Example Workflow for Dashboard Development:**

# Step 1: Reference documents

@cursor I need to implement the dashboard following Dashboard.docx, using Master API Endpoints.docx for data fetching, and i18n Architecture.docx for multi-language support.

# Step 2: Generate component structure

@cursor Create the dashboard layout component with proper TypeScript interfaces

# Step 3: Add signal heatmap

@cursor Add the signal heatmap component following the specifications in Signals.docx

# Step 4: Integrate API calls

@cursor Add API integration using the endpoints from Master API Endpoints.docx

# Step 5: Add multi-language support

@cursor Implement i18n support following i18n Architecture.docx patterns

**2. Code Review Workflow**

**Before Committing:**

# Check against specifications

@cursor Review this component against [Document Name].docx specifications

# Verify multi-language compliance

@cursor Ensure this code supports EN/DE/AR languages per i18n Architecture.docx

# Check API compliance

@cursor Verify API calls match Master API Endpoints.docx specifications

# Performance review

@cursor Optimize this code for performance and accessibility

**3. Debugging Workflow**

**Common Issues:**

# Authentication issues

@cursor This auth flow isn't working. Check against Complete Authentication System.docx

# Translation issues

@cursor The German translations aren't loading. Check i18n implementation

# RTL layout issues

@cursor Arabic layout is broken. Fix according to RTL Layout System.docx

# API errors

@cursor API call failing. Verify against Master API Endpoints.docx

**🛠️ CURSOR PROJECT SETUP**

**1. Project Initialization with Cursor**

# Create new project

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

# Open in Cursor

cursor kurzora-platform

# Initialize with project context

@cursor This is the Kurzora trading platform project. I have 60+ specification documents. The main references are:

- Master DB Schema.docx for database

- Master API Endpoints.docx for APIs

- Complete Authentication System.docx for auth

- i18n Architecture.docx for multi-language

Set up the project structure following Complete Project File Structure.docx

**2. Environment Configuration**

# Set up environment variables

@cursor Create .env.local file following the environment variables specified in Backend Architecture Analysis.docx and include all necessary API keys for Supabase, Stripe, Polygon.io, etc.

**3. Dependencies Installation**

# Install required packages

@cursor Install all dependencies needed for the Kurzora project based on the specifications in the technical documents, including Next.js, Supabase, Stripe, internationalization, and UI libraries.

**📋 CURSOR BEST PRACTICES**

**1. Document Reference Patterns**

**Always Specify Source Documents:**

❌ Bad: "Create a dashboard component"

✅ Good: "Create a dashboard component following Dashboard.docx with signal heatmap from Signals.docx"

**Include Context and Constraints:**

❌ Bad: "Add authentication"

✅ Good: "Add authentication using Supabase Auth following Complete Authentication System.docx with multi-language support from i18n Architecture.docx"

**2. Code Generation Guidelines**

**Specify Implementation Details:**

❌ Bad: "Make it responsive"

✅ Good: "Make it responsive using Tailwind CSS with RTL support for Arabic following RTL Layout System.docx"

**Include Error Handling:**

Always include: "Add proper error handling, loading states, and TypeScript interfaces"

**3. Quality Assurance**

**Regular Specification Checks:**

# Before each commit

@cursor Review all changes against the relevant specification documents and ensure compliance with multi-language requirements

**Performance Verification:**

# Regular performance checks

@cursor Optimize this code for performance and check bundle size impact

**🚨 COMMON CURSOR ISSUES & SOLUTIONS**

**1. Document Reference Issues**

**Problem:** Cursor not understanding document references **Solution:**

@cursor I'm referencing specification documents for the Kurzora project. When I mention "Dashboard.docx" I'm referring to the detailed UI specifications for the dashboard component. Please ask for clarification if you need specific details from any document.

**2. Context Loss**

**Problem:** Cursor forgetting project context **Solution:**

@cursor This is for the Kurzora international trading platform with multi-language support (EN/DE/AR). Always consider internationalization and RTL support in your suggestions.

**3. Inconsistent Code Style**

**Problem:** Generated code doesn't match project patterns **Solution:**

@cursor Follow the TypeScript and component patterns established in the project. Use proper interfaces, error handling, and follow the architecture from Backend Architecture Analysis.docx

**✅ CURSOR SUCCESS CHECKLIST**

**Before Starting Development:**

* [ ] Cursor installed with proper extensions
* [ ] Project opened in Cursor
* [ ] Environment variables configured
* [ ] Document hierarchy understood
* [ ] Prompt templates ready

**For Each Feature Implementation:**

* [ ] Relevant documents identified
* [ ] Proper prompt template used
* [ ] Multi-language support considered
* [ ] RTL support included (if applicable)
* [ ] API endpoints verified against Master API Endpoints.docx
* [ ] Code reviewed against specifications
* [ ] Testing completed

**Before Deployment:**

* [ ] All components verified against specifications
* [ ] Multi-language testing completed
* [ ] Performance optimization done
* [ ] Security review completed
* [ ] API compliance verified

**🚀 READY FOR EFFICIENT AI-ASSISTED DEVELOPMENT**

**These instructions will help you leverage Cursor AI effectively with your comprehensive Kurzora documentation. Follow the patterns and templates to build your international trading platform efficiently!**