**🌐 KURZORA PLATFORM - MASTER API ENDPOINTS**

**📋 DOCUMENT STATUS**

**Status:** ✅ **MASTER REFERENCE**  
**Version:** 1.0  
**Last Updated:** December 2024  
**Authority:** Single Source of Truth for ALL API Endpoints

**🎯 PURPOSE**

This document serves as the **ONLY authoritative reference** for all API endpoints in the Kurzora Platform. All other documents, frontend code, and backend implementations MUST conform to this specification.

**🏗️ API ARCHITECTURE PRINCIPLES**

**Base Configuration**

const API\_CONFIG = {

BASE\_URL: process.env.VITE\_API\_URL || 'http://localhost:3001',

VERSION: 'v1',

TIMEOUT: 30000, // 30 seconds

RETRY\_ATTEMPTS: 3

}

// Complete Base URL Pattern

const FULL\_BASE\_URL = `${API\_CONFIG.BASE\_URL}/api/${API\_CONFIG.VERSION}`

// Example: http://localhost:3001/api/v1

**Response Format Standard**

interface ApiResponse<T = any> {

success: boolean

data?: T

error?: {

code: string

message: string

details?: any

}

meta?: {

total?: number

page?: number

limit?: number

hasMore?: boolean

}

timestamp: string

requestId?: string

}

**🔐 AUTHENTICATION ENDPOINTS**

**Auth Base: /api/v1/auth**

const AUTH\_ENDPOINTS = {

// User Authentication

LOGIN: 'POST /api/v1/auth/login',

REGISTER: 'POST /api/v1/auth/register',

LOGOUT: 'POST /api/v1/auth/logout',

REFRESH\_TOKEN: 'POST /api/v1/auth/refresh',

// Password Management

FORGOT\_PASSWORD: 'POST /api/v1/auth/forgot-password',

RESET\_PASSWORD: 'POST /api/v1/auth/reset-password',

CHANGE\_PASSWORD: 'POST /api/v1/auth/change-password',

// Email Verification

VERIFY\_EMAIL: 'POST /api/v1/auth/verify-email',

RESEND\_VERIFICATION: 'POST /api/v1/auth/resend-verification',

// Session Management

GET\_SESSION: 'GET /api/v1/auth/session',

LOGOUT\_ALL\_DEVICES: 'POST /api/v1/auth/logout-all'

}

// Frontend Usage Examples:

const authAPI = {

login: (credentials) =>

POST(`${FULL\_BASE\_URL}/auth/login`, credentials),

register: (userData) =>

POST(`${FULL\_BASE\_URL}/auth/register`, userData),

refreshToken: (refreshToken) =>

POST(`${FULL\_BASE\_URL}/auth/refresh`, { refreshToken })

}

**👤 USER MANAGEMENT ENDPOINTS**

**User Base: /api/v1/user**

const USER\_ENDPOINTS = {

// Profile Management

GET\_PROFILE: 'GET /api/v1/user/profile',

UPDATE\_PROFILE: 'PUT /api/v1/user/profile',

UPLOAD\_AVATAR: 'POST /api/v1/user/profile/avatar',

DELETE\_AVATAR: 'DELETE /api/v1/user/profile/avatar',

// Preferences

GET\_PREFERENCES: 'GET /api/v1/user/preferences',

UPDATE\_PREFERENCES: 'PUT /api/v1/user/preferences',

RESET\_PREFERENCES: 'POST /api/v1/user/preferences/reset',

// Security Settings

GET\_SECURITY: 'GET /api/v1/user/security',

UPDATE\_SECURITY: 'PUT /api/v1/user/security',

ENABLE\_2FA: 'POST /api/v1/user/security/2fa/enable',

DISABLE\_2FA: 'POST /api/v1/user/security/2fa/disable',

VERIFY\_2FA: 'POST /api/v1/user/security/2fa/verify',

// API Keys Management

LIST\_API\_KEYS: 'GET /api/v1/user/api-keys',

CREATE\_API\_KEY: 'POST /api/v1/user/api-keys',

UPDATE\_API\_KEY: 'PUT /api/v1/user/api-keys/:id',

DELETE\_API\_KEY: 'DELETE /api/v1/user/api-keys/:id',

// Account Management

DELETE\_ACCOUNT: 'DELETE /api/v1/user/account',

EXPORT\_DATA: 'GET /api/v1/user/export'

}

// Frontend Usage Examples:

const userAPI = {

getProfile: () =>

GET(`${FULL\_BASE\_URL}/user/profile`),

updateProfile: (updates) =>

PUT(`${FULL\_BASE\_URL}/user/profile`, updates),

createApiKey: (keyData) =>

POST(`${FULL\_BASE\_URL}/user/api-keys`, keyData)

}

**📊 DASHBOARD ENDPOINTS**

**Dashboard Base: /api/v1/dashboard**

const DASHBOARD\_ENDPOINTS = {

// Main Dashboard Data

GET\_METRICS: 'GET /api/v1/dashboard/metrics',

GET\_OVERVIEW: 'GET /api/v1/dashboard/overview',

// Portfolio Summary

GET\_PORTFOLIO\_SUMMARY: 'GET /api/v1/dashboard/portfolio',

GET\_PERFORMANCE\_CHART: 'GET /api/v1/dashboard/performance',

// Recent Activity

GET\_RECENT\_ACTIVITY: 'GET /api/v1/dashboard/activity',

GET\_RECENT\_TRADES: 'GET /api/v1/dashboard/trades/recent',

// Alerts & Notifications

GET\_ALERTS: 'GET /api/v1/dashboard/alerts',

MARK\_ALERT\_READ: 'PUT /api/v1/dashboard/alerts/:id/read',

CLEAR\_ALL\_ALERTS: 'DELETE /api/v1/dashboard/alerts'

}

// Query Parameters for Dashboard:

interface DashboardParams {

// Performance Chart

period?: '1D' | '1W' | '1M' | '3M' | '6M' | '1Y' | 'ALL'

// Recent Activity

limit?: number

offset?: number

// Alerts

unreadOnly?: boolean

type?: 'signal' | 'trade' | 'system'

}

// Frontend Usage Examples:

const dashboardAPI = {

getMetrics: () =>

GET(`${FULL\_BASE\_URL}/dashboard/metrics`),

getPerformance: (period = '1M') =>

GET(`${FULL\_BASE\_URL}/dashboard/performance?period=${period}`),

getRecentTrades: (limit = 5) =>

GET(`${FULL\_BASE\_URL}/dashboard/trades/recent?limit=${limit}`)

}

**📈 SIGNALS ENDPOINTS**

**Signals Base: /api/v1/signals**

const SIGNALS\_ENDPOINTS = {

// Signal Data

LIST\_SIGNALS: 'GET /api/v1/signals',

GET\_SIGNAL: 'GET /api/v1/signals/:id',

GET\_SIGNAL\_HISTORY: 'GET /api/v1/signals/:id/history',

// Signal Actions

EXECUTE\_SIGNAL: 'POST /api/v1/signals/:id/execute',

BOOKMARK\_SIGNAL: 'POST /api/v1/signals/:id/bookmark',

UNBOOKMARK\_SIGNAL: 'DELETE /api/v1/signals/:id/bookmark',

// Signal Analytics

GET\_SIGNAL\_PERFORMANCE: 'GET /api/v1/signals/performance',

GET\_SIGNAL\_STATISTICS: 'GET /api/v1/signals/statistics',

// Real-time Updates

SUBSCRIBE\_UPDATES: 'WS /api/v1/signals/subscribe',

UNSUBSCRIBE\_UPDATES: 'WS /api/v1/signals/unsubscribe'

}

// Query Parameters for Signals:

interface SignalsParams {

// Filtering

timeframe?: '1H' | '4H' | '1D' | '1W'

minScore?: number

maxScore?: number

sector?: string

market?: 'US' | 'global'

shariahCompliant?: boolean

status?: 'active' | 'triggered' | 'expired'

// Pagination

page?: number

limit?: number

// Sorting

sortBy?: 'score' | 'timestamp' | 'symbol'

sortOrder?: 'asc' | 'desc'

}

// Frontend Usage Examples:

const signalsAPI = {

getSignals: (params: SignalsParams) =>

GET(`${FULL\_BASE\_URL}/signals`, { params }),

getSignal: (id: string) =>

GET(`${FULL\_BASE\_URL}/signals/${id}`),

executeSignal: (id: string, tradeData) =>

POST(`${FULL\_BASE\_URL}/signals/${id}/execute`, tradeData)

}

**📊 PORTFOLIO ENDPOINTS**

**Portfolio Base: /api/v1/portfolio**

const PORTFOLIO\_ENDPOINTS = {

// Portfolio Overview

GET\_PORTFOLIO: 'GET /api/v1/portfolio',

GET\_PERFORMANCE: 'GET /api/v1/portfolio/performance',

GET\_ANALYTICS: 'GET /api/v1/portfolio/analytics',

// Positions Management

LIST\_POSITIONS: 'GET /api/v1/portfolio/positions',

GET\_POSITION: 'GET /api/v1/portfolio/positions/:id',

UPDATE\_POSITION: 'PUT /api/v1/portfolio/positions/:id',

CLOSE\_POSITION: 'DELETE /api/v1/portfolio/positions/:id',

// Trade History

GET\_TRADE\_HISTORY: 'GET /api/v1/portfolio/trades',

GET\_TRADE: 'GET /api/v1/portfolio/trades/:id',

EXPORT\_TRADES: 'GET /api/v1/portfolio/trades/export',

// Paper Trading

EXECUTE\_PAPER\_TRADE: 'POST /api/v1/portfolio/paper-trade',

GET\_PAPER\_TRADES: 'GET /api/v1/portfolio/paper-trades',

// Portfolio Settings

GET\_SETTINGS: 'GET /api/v1/portfolio/settings',

UPDATE\_SETTINGS: 'PUT /api/v1/portfolio/settings',

RESET\_PORTFOLIO: 'POST /api/v1/portfolio/reset'

}

// Query Parameters for Portfolio:

interface PortfolioParams {

// Trade History

dateFrom?: string // ISO date

dateTo?: string // ISO date

symbol?: string

status?: 'open' | 'closed'

exitReason?: 'take\_profit' | 'stop\_loss' | 'manual' | 'expired'

// Performance

period?: '1D' | '1W' | '1M' | '3M' | '6M' | '1Y' | 'ALL'

benchmark?: 'SPY' | 'QQQ' | 'none'

// Pagination

page?: number

limit?: number

}

// Frontend Usage Examples:

const portfolioAPI = {

getPortfolio: () =>

GET(`${FULL\_BASE\_URL}/portfolio`),

getTradeHistory: (params: PortfolioParams) =>

GET(`${FULL\_BASE\_URL}/portfolio/trades`, { params }),

executePaperTrade: (tradeData) =>

POST(`${FULL\_BASE\_URL}/portfolio/paper-trade`, tradeData)

}

**📝 WATCHLIST ENDPOINTS**

**Watchlist Base: /api/v1/watchlists**

const WATCHLIST\_ENDPOINTS = {

// Watchlist Management

LIST\_WATCHLISTS: 'GET /api/v1/watchlists',

CREATE\_WATCHLIST: 'POST /api/v1/watchlists',

GET\_WATCHLIST: 'GET /api/v1/watchlists/:id',

UPDATE\_WATCHLIST: 'PUT /api/v1/watchlists/:id',

DELETE\_WATCHLIST: 'DELETE /api/v1/watchlists/:id',

// Watchlist Items

LIST\_ITEMS: 'GET /api/v1/watchlists/:id/items',

ADD\_ITEM: 'POST /api/v1/watchlists/:id/items',

REMOVE\_ITEM: 'DELETE /api/v1/watchlists/:id/items/:symbol',

UPDATE\_ITEM\_NOTES: 'PUT /api/v1/watchlists/:id/items/:symbol',

// Bulk Operations

BULK\_ADD\_ITEMS: 'POST /api/v1/watchlists/:id/items/bulk',

BULK\_REMOVE\_ITEMS: 'DELETE /api/v1/watchlists/:id/items/bulk',

// Watchlist Sharing (Future)

SHARE\_WATCHLIST: 'POST /api/v1/watchlists/:id/share',

IMPORT\_WATCHLIST: 'POST /api/v1/watchlists/import'

}

// Frontend Usage Examples:

const watchlistAPI = {

getWatchlists: () =>

GET(`${FULL\_BASE\_URL}/watchlists`),

addToWatchlist: (watchlistId: string, symbol: string) =>

POST(`${FULL\_BASE\_URL}/watchlists/${watchlistId}/items`, { symbol }),

removeFromWatchlist: (watchlistId: string, symbol: string) =>

DELETE(`${FULL\_BASE\_URL}/watchlists/${watchlistId}/items/${symbol}`)

}

**💳 SUBSCRIPTION & BILLING ENDPOINTS**

**Billing Base: /api/v1/billing**

const BILLING\_ENDPOINTS = {

// Subscription Management

GET\_SUBSCRIPTION: 'GET /api/v1/billing/subscription',

UPDATE\_SUBSCRIPTION: 'PUT /api/v1/billing/subscription',

CANCEL\_SUBSCRIPTION: 'DELETE /api/v1/billing/subscription',

REACTIVATE\_SUBSCRIPTION: 'POST /api/v1/billing/subscription/reactivate',

// Payment Methods

LIST\_PAYMENT\_METHODS: 'GET /api/v1/billing/payment-methods',

ADD\_PAYMENT\_METHOD: 'POST /api/v1/billing/payment-methods',

UPDATE\_PAYMENT\_METHOD: 'PUT /api/v1/billing/payment-methods/:id',

DELETE\_PAYMENT\_METHOD: 'DELETE /api/v1/billing/payment-methods/:id',

SET\_DEFAULT\_PAYMENT: 'POST /api/v1/billing/payment-methods/:id/default',

// Billing History

GET\_INVOICES: 'GET /api/v1/billing/invoices',

GET\_INVOICE: 'GET /api/v1/billing/invoices/:id',

DOWNLOAD\_INVOICE: 'GET /api/v1/billing/invoices/:id/download',

// Plans & Pricing

LIST\_PLANS: 'GET /api/v1/billing/plans',

GET\_PLAN: 'GET /api/v1/billing/plans/:id',

// Stripe Integration

CREATE\_PAYMENT\_INTENT: 'POST /api/v1/billing/payment-intent',

CONFIRM\_PAYMENT: 'POST /api/v1/billing/confirm-payment'

}

// Frontend Usage Examples:

const billingAPI = {

getSubscription: () =>

GET(`${FULL\_BASE\_URL}/billing/subscription`),

getPlans: () =>

GET(`${FULL\_BASE\_URL}/billing/plans`),

updateSubscription: (planId: string) =>

PUT(`${FULL\_BASE\_URL}/billing/subscription`, { planId })

}

**🔔 NOTIFICATIONS ENDPOINTS**

**Notifications Base: /api/v1/notifications**

const NOTIFICATION\_ENDPOINTS = {

// Notification Management

LIST\_NOTIFICATIONS: 'GET /api/v1/notifications',

GET\_NOTIFICATION: 'GET /api/v1/notifications/:id',

MARK\_READ: 'PUT /api/v1/notifications/:id/read',

MARK\_UNREAD: 'PUT /api/v1/notifications/:id/unread',

DELETE\_NOTIFICATION: 'DELETE /api/v1/notifications/:id',

// Bulk Operations

MARK\_ALL\_READ: 'PUT /api/v1/notifications/mark-all-read',

DELETE\_ALL: 'DELETE /api/v1/notifications/all',

// Notification Settings

GET\_SETTINGS: 'GET /api/v1/notifications/settings',

UPDATE\_SETTINGS: 'PUT /api/v1/notifications/settings',

// Push Notifications

REGISTER\_DEVICE: 'POST /api/v1/notifications/devices',

UNREGISTER\_DEVICE: 'DELETE /api/v1/notifications/devices/:token',

// Email Preferences

GET\_EMAIL\_PREFERENCES: 'GET /api/v1/notifications/email-preferences',

UPDATE\_EMAIL\_PREFERENCES: 'PUT /api/v1/notifications/email-preferences',

UNSUBSCRIBE\_EMAIL: 'POST /api/v1/notifications/unsubscribe'

}

// Frontend Usage Examples:

const notificationAPI = {

getNotifications: (unreadOnly = false) =>

GET(`${FULL\_BASE\_URL}/notifications?unreadOnly=${unreadOnly}`),

markAsRead: (id: string) =>

PUT(`${FULL\_BASE\_URL}/notifications/${id}/read`),

updateSettings: (settings) =>

PUT(`${FULL\_BASE\_URL}/notifications/settings`, settings)

}

**🛠️ ADMIN ENDPOINTS**

**Admin Base: /api/v1/admin**

const ADMIN\_ENDPOINTS = {

// User Management

LIST\_USERS: 'GET /api/v1/admin/users',

GET\_USER: 'GET /api/v1/admin/users/:id',

UPDATE\_USER: 'PUT /api/v1/admin/users/:id',

DELETE\_USER: 'DELETE /api/v1/admin/users/:id',

SUSPEND\_USER: 'POST /api/v1/admin/users/:id/suspend',

ACTIVATE\_USER: 'POST /api/v1/admin/users/:id/activate',

// Signal Management

LIST\_ALL\_SIGNALS: 'GET /api/v1/admin/signals',

CREATE\_SIGNAL: 'POST /api/v1/admin/signals',

UPDATE\_SIGNAL: 'PUT /api/v1/admin/signals/:id',

DELETE\_SIGNAL: 'DELETE /api/v1/admin/signals/:id',

FORCE\_SIGNAL\_EXPIRY: 'POST /api/v1/admin/signals/:id/expire',

// System Management

GET\_SYSTEM\_STATS: 'GET /api/v1/admin/system/stats',

GET\_SYSTEM\_HEALTH: 'GET /api/v1/admin/system/health',

GET\_ERROR\_LOGS: 'GET /api/v1/admin/system/logs/errors',

GET\_AUDIT\_LOGS: 'GET /api/v1/admin/system/logs/audit',

// Configuration

GET\_CONFIG: 'GET /api/v1/admin/config',

UPDATE\_CONFIG: 'PUT /api/v1/admin/config',

// Analytics

GET\_ANALYTICS: 'GET /api/v1/admin/analytics',

EXPORT\_DATA: 'GET /api/v1/admin/export'

}

// Frontend Usage Examples:

const adminAPI = {

getUsers: (params) =>

GET(`${FULL\_BASE\_URL}/admin/users`, { params }),

getSystemHealth: () =>

GET(`${FULL\_BASE\_URL}/admin/system/health`),

getAnalytics: (dateRange) =>

GET(`${FULL\_BASE\_URL}/admin/analytics`, { params: dateRange })

}

**🌐 PUBLIC ENDPOINTS**

**Public Base: /api/v1/public**

const PUBLIC\_ENDPOINTS = {

// Demo & Marketing

GET\_DEMO\_SIGNALS: 'GET /api/v1/public/demo/signals',

GET\_PERFORMANCE\_STATS: 'GET /api/v1/public/performance',

GET\_TESTIMONIALS: 'GET /api/v1/public/testimonials',

// Health & Status

HEALTH\_CHECK: 'GET /api/v1/public/health',

STATUS: 'GET /api/v1/public/status',

VERSION: 'GET /api/v1/public/version',

// Contact & Support

SUBMIT\_CONTACT: 'POST /api/v1/public/contact',

SUBSCRIBE\_NEWSLETTER: 'POST /api/v1/public/newsletter'

}

// Frontend Usage Examples:

const publicAPI = {

getDemoSignals: () =>

GET(`${FULL\_BASE\_URL}/public/demo/signals`),

submitContact: (contactData) =>

POST(`${FULL\_BASE\_URL}/public/contact`, contactData),

healthCheck: () =>

GET(`${FULL\_BASE\_URL}/public/health`)

}

**🔌 WEBSOCKET ENDPOINTS**

**WebSocket Base: ws://localhost:3001/ws/v1**

const WEBSOCKET\_ENDPOINTS = {

// Real-time Data Streams

SIGNALS\_STREAM: 'ws://localhost:3001/ws/v1/signals',

PORTFOLIO\_STREAM: 'ws://localhost:3001/ws/v1/portfolio',

NOTIFICATIONS\_STREAM: 'ws://localhost:3001/ws/v1/notifications',

// Market Data

MARKET\_DATA\_STREAM: 'ws://localhost:3001/ws/v1/market-data',

PRICE\_UPDATES: 'ws://localhost:3001/ws/v1/prices',

// System Updates

SYSTEM\_ALERTS: 'ws://localhost:3001/ws/v1/system-alerts'

}

// WebSocket Message Format:

interface WebSocketMessage {

type: 'signal\_update' | 'price\_update' | 'notification' | 'error'

data: any

timestamp: string

channel: string

}

// Frontend Usage Examples:

const wsAPI = {

connectToSignals: (onMessage) => {

const ws = new WebSocket(WEBSOCKET\_ENDPOINTS.SIGNALS\_STREAM)

ws.onmessage = (event) => onMessage(JSON.parse(event.data))

return ws

}

}

**📱 FRONTEND API CLIENT IMPLEMENTATION**

**Complete API Client Setup**

// api/client.ts - Master API Client

import axios, { AxiosResponse } from 'axios'

const API\_CLIENT = axios.create({

baseURL: `${process.env.VITE\_API\_URL}/api/v1`,

timeout: 30000,

headers: {

'Content-Type': 'application/json'

}

})

// Request Interceptor (Add Auth Token)

API\_CLIENT.interceptors.request.use((config) => {

const token = localStorage.getItem('accessToken')

if (token) {

config.headers.Authorization = `Bearer ${token}`

}

return config

})

// Response Interceptor (Handle Errors)

API\_CLIENT.interceptors.response.use(

(response: AxiosResponse) => response,

async (error) => {

if (error.response?.status === 401) {

// Handle token refresh or redirect to login

await refreshToken()

}

return Promise.reject(error)

}

)

// Helper Functions

const GET = (url: string, params?: any) =>

API\_CLIENT.get(url, { params })

const POST = (url: string, data?: any) =>

API\_CLIENT.post(url, data)

const PUT = (url: string, data?: any) =>

API\_CLIENT.put(url, data)

const DELETE = (url: string) =>

API\_CLIENT.delete(url)

export { GET, POST, PUT, DELETE, API\_CLIENT }

**Centralized API Functions**

// api/index.ts - Export All API Functions

export const kurzorAPI = {

// Authentication

auth: {

login: (credentials) => POST('/auth/login', credentials),

register: (userData) => POST('/auth/register', userData),

logout: () => POST('/auth/logout'),

refreshToken: (token) => POST('/auth/refresh', { refreshToken: token })

},

// User Management

user: {

getProfile: () => GET('/user/profile'),

updateProfile: (data) => PUT('/user/profile', data),

getPreferences: () => GET('/user/preferences'),

updatePreferences: (prefs) => PUT('/user/preferences', prefs)

},

// Dashboard

dashboard: {

getMetrics: () => GET('/dashboard/metrics'),

getPortfolio: () => GET('/dashboard/portfolio'),

getPerformance: (period) => GET(`/dashboard/performance?period=${period}`)

},

// Signals

signals: {

getSignals: (params) => GET('/signals', params),

getSignal: (id) => GET(`/signals/${id}`),

executeSignal: (id, tradeData) => POST(`/signals/${id}/execute`, tradeData)

},

// Portfolio

portfolio: {

getPortfolio: () => GET('/portfolio'),

getPositions: () => GET('/portfolio/positions'),

getTradeHistory: (params) => GET('/portfolio/trades', params),

executeTrade: (tradeData) => POST('/portfolio/paper-trade', tradeData)

},

// Watchlists

watchlists: {

getWatchlists: () => GET('/watchlists'),

createWatchlist: (data) => POST('/watchlists', data),

addToWatchlist: (id, symbol) => POST(`/watchlists/${id}/items`, { symbol })

}

}

**🚀 IMPLEMENTATION CHECKLIST**

**Backend Implementation**

* [ ] Create routes following exact endpoint patterns above
* [ ] Implement middleware for authentication, validation, rate limiting
* [ ] Add proper error handling with standardized error codes
* [ ] Set up WebSocket connections for real-time features
* [ ] Implement API versioning (v1 prefix)

**Frontend Implementation**

* [ ] Use the centralized API client above
* [ ] Import kurzorAPI in all components
* [ ] Replace any hardcoded endpoints with the standardized ones
* [ ] Implement proper error handling for all API calls
* [ ] Add loading states and retry logic

**Testing & Validation**

* [ ] Test all endpoints with proper authentication
* [ ] Validate response formats match ApiResponse interface
* [ ] Test WebSocket connections and message handling
* [ ] Verify error handling and edge cases

**⚠️ CRITICAL COMPLIANCE RULES**

**🚨 MANDATORY REQUIREMENTS**

1. **ALL endpoints MUST follow the exact patterns defined above**
2. **NO deviations from the URL structure without updating this document**
3. **ALL API responses MUST use the standardized ApiResponse interface**
4. **ALL authentication endpoints MUST use JWT Bearer tokens**
5. **ALL WebSocket connections MUST follow the defined message format**

**🔄 Document Updates**

Any changes to API endpoints MUST:

1. Be approved by the technical lead
2. Update this master document first
3. Be communicated to all developers
4. Include migration guide for existing endpoints

**📞 SUPPORT & QUESTIONS**

For any questions about API endpoints or conflicts with existing implementations:

1. **First:** Check this document (single source of truth)
2. **Second:** Update your code to match these specifications
3. **Third:** If you need to change an endpoint, update this document first

**Remember:** This document supersedes ALL other API documentation. When in doubt, follow this specification.

**✅ This master document resolves all API endpoint naming conflicts and serves as the ONLY authoritative reference for the Kurzora Platform.**

Ma