📊 Database Schema Documentation

Overview

This document describes the complete database schema for MyDen investment application. All models use MongoDB with Mongoose ODM.

📋 Table of Contents

User Model

Location: `backend/src/models/User.js`

Purpose

Stores user account information, profile data, preferences, and security settings.

Schema Fields

Basic Information

- `firstName` (String, required) - User's first name
- `lastName` (String, required) - User's last name
- `email` (String, required, unique) - User email address
- `phone` (String, required) - Phone number
- `password` (String, required) - Hashed password
- `role` (String) - User role: 'user' or 'admin'

Profile Information

- `profilePicture` (String) - URL to profile image
- `bio` (String, max 500) - User biography
- `dateOfBirth` (Date) - Date of birth
- `country` (String) - User's country
- `timezone` (String) - User timezone (default: America/New_York)
- `currency` (String) - Preferred currency (default: USD)

User Preferences

- `preferences.riskTolerance` (String) - conservative | moderate | aggressive
- `preferences.defaultChart` (String) - Default chart timeframe (default: 7d)
- `preferences.theme` (String) - light | dark | auto
- `preferences.language` (String) - Language code (default: en)

- `preferences.dashboardLayout` (String) - Dashboard layout preference

Notification Settings

- `notifications.email` (Boolean) - Email notifications enabled
- `notifications.push` (Boolean) - Push notifications enabled
- `notifications.priceAlerts` (Boolean) - Price alert notifications
- `notifications.portfolioUpdates` (Boolean) - Portfolio update notifications
- `notifications.marketNews` (Boolean) - Market news notifications
- `notifications.weeklyReport` (Boolean) - Weekly report emails

Privacy Settings

- `privacy.showPortfolio` (Boolean) - Show portfolio publicly
- `privacy.showInLeaderboard` (Boolean) - Appear in leaderboards
- `privacy.allowSocialSharing` (Boolean) - Allow social sharing

Security & Verification

- `isEmailVerified` (Boolean) - Email verification status
- `emailVerificationToken` (String) - Token for email verification
- `emailVerificationExpires` (Date) - Token expiration
- `twoFactorEnabled` (Boolean) - 2FA status
- `twoFactorSecret` (String) - 2FA secret key
- `resetPasswordToken` (String) - Password reset token
- `resetPasswordExpires` (Date) - Reset token expiration

App Features

- `watchlist` (Array of Strings) - Watched cryptocurrency symbols
- `isPremium` (Boolean) - Premium subscription status
- `premiumExpiresAt` (Date) - Premium expiration date

Activity Tracking

- `lastLogin` (Date) - Last login timestamp
- `loginCount` (Number) - Total login count
- `createdAt` (Date) - Account creation date
- `updatedAt` (Date) - Last profile update

Indexes

- `email` (ascending)
- `createdAt` (descending)
- `isPremium` (ascending)

Methods

- `matchPassword(enteredPassword)` - Compare password with hash

Hooks

- **pre('save')** - Hash password before saving

Portfolio Model

Location: `backend/src/models/Portfolio.js`

Purpose

Tracks user's cryptocurrency holdings with current values and profit/loss calculations.

Schema Fields

- `user` (ObjectId, ref: User, required) - Portfolio owner
- `symbol` (String, required) - Cryptocurrency symbol (e.g., BTC, ETH)
- `quantity` (Number, required) - Amount held
- `averageBuyPrice` (Number, required) - Average purchase price
- `totalInvested` (Number, required) - Total invested amount
- `currentPrice` (Number) - Current market price (cached)
- `currentValue` (Number) - Current value (quantity × currentPrice)
- `profitLoss` (Number) - Profit/loss amount
- `profitLossPercent` (Number) - Profit/loss percentage
- `lastUpdated` (Date) - Last price update
- `createdAt` (Date) - Position creation date

Indexes

- `user + symbol` (compound index)
- `user + currentValue` (descending)

Transaction Model

Location: `backend/src/models/Transaction.js`

Purpose

Records all buy and sell transactions for accounting and history tracking.

Schema Fields

- `user` (ObjectId, ref: User, required) - Transaction owner
- `type` (String, required) - 'buy' or 'sell'
- `symbol` (String, required) - Cryptocurrency symbol
- `quantity` (Number, required) - Amount traded
- `price` (Number, required) - Price per unit
- `total` (Number, required) - Total transaction value
- `fees` (Number) - Transaction fees
- `notes` (String) - User notes
- `source` (String) - manual | api | auto-rebalance | quick-trade
- `date` (Date) - Transaction date

Indexes

- `user + date` (descending)
- `user + symbol` (compound index)

InvestmentStrategy Model

Location: `backend/src/models/InvestmentStrategy.js`

Purpose

Stores AI-powered investment strategy templates for different risk levels and market conditions.

Schema Fields

- `name` (String, required, unique) - Strategy name
- `description` (String, required) - Strategy description
- `riskLevel` (String, required) - conservative | moderate | aggressive
- `allocation` (Map of Numbers, required) - Asset allocation percentages
- `marketConditions` (Array) - bull | bear | sideways | volatile | all
- `targetReturn` (Number) - Expected annual return %
- `maxDrawdown` (Number) - Maximum expected loss %
- `rebalanceFrequency` (String) - daily | weekly | monthly | quarterly
- `isActive` (Boolean) - Strategy status
- `createdAt` (Date) - Creation timestamp
- `updatedAt` (Date) - Last update timestamp

Methods

- `validateAllocation()` - Ensure allocation totals 100%

PriceAlert Model

Location: `backend/src/models/PriceAlert.js`

Purpose

Manages smart price alerts with multiple trigger conditions and notification methods.

Schema Fields

Basic Info

- `user` (ObjectId, ref: User, required) - Alert owner
- `symbol` (String, required) - Cryptocurrency symbol
- `name` (String) - Custom alert name

Alert Configuration

- `targetPrice` (Number, required) - Price target
- `condition` (String, required) - above | below | crosses_above | crosses_below
- `alertType` (String) - price | percentage_change | volume | market_cap
- `percentageThreshold` (Number) - For percentage change alerts
- `timeframe` (String) - 1h | 24h | 7d | 30d

Notification Settings

- `notificationMethod.email` (Boolean) - Send email
- `notificationMethod.push` (Boolean) - Send push notification
- `notificationMethod.sms` (Boolean) - Send SMS

Alert Status

- `isActive` (Boolean) - Alert active status
- `triggered` (Boolean) - Has been triggered

- `triggeredAt` (Date) - Trigger timestamp
- `triggeredPrice` (Number) - Price when triggered

Recurrence

- `recurring` (Boolean) - Auto-reset after triggering
- `triggerCount` (Number) - Number of times triggered

Metadata

- `note` (String) - User notes
- `priority` (String) - low | medium | high
- `createdAt` (Date) - Creation date
- `updatedAt` (Date) - Last update

Indexes

- `user + isActive` (compound)
- `user + triggered` (compound)
- `symbol + isActive` (compound)
- `triggeredAt` (descending)

Methods

- `shouldTrigger(currentPrice)` - Check if alert should trigger
- `trigger(currentPrice)` - Mark alert as triggered

Hooks

- **pre('save')** - Update timestamp

Notification Model

Location: `backend/src/models/Notification.js`

Purpose

Manages in-app notifications with support for multiple delivery channels.

Schema Fields

Content

- `user` (ObjectId, ref: User, required) - Notification recipient
- `type` (String, required) - alert | system | achievement | trade | portfolio | security | promotion | news
- `title` (String, required, max 100) - Notification title
- `message` (String, required, max 500) - Notification message
- `metadata` (Mixed) - Additional data

State

- `read` (Boolean) - Read status
- `readAt` (Date) - When marked as read

Action

- `actionUrl` (String) - URL to navigate
- `actionLabel` (String) - Action button label

Visual

- `icon` (String) - Icon name or emoji
- `color` (String) - info | success | warning | error

Priority & Display

- `priority` (String) - low | medium | high | urgent
- `persistent` (Boolean) - Don't auto-dismiss

Delivery Status

- `sent` (Boolean) - Sent status
- `sentAt` (Date) - Sent timestamp
- `deliveryChannels` - inApp, email, push, sms flags

Timestamps

- `expiresAt` (Date) - Auto-delete date (TTL)
- `createdAt` (Date) - Creation date

Indexes

- `user + read + createdAt` (compound)
- `user + type` (compound)
- `createdAt` (descending)
- `expiresAt` (TTL index)

Methods

- `markAsRead()` - Mark single notification as read
- **static** `markMultipleAsRead(userId, ids)` - Bulk mark as read
- **static** `getUnreadCount(userId)` - Get unread count
- **static** `cleanupOldNotifications(daysOld)` - Delete old notifications

ChatMessage Model

Location: `backend/src/models/ChatMessage.js`

Purpose

Stores AI chat assistant conversation history with context and metadata.

Schema Fields

Basic Info

- `user` (ObjectId, ref: User, required) - User who sent message
- `conversationId` (String, required) - Groups messages in conversation
- `role` (String, required) - user | assistant | system
- `message` (String, required, max 2000) - Message content
- `response` (String) - AI response (for assistant role)

Context Data

- `context.portfolioValue` (Number) - User's portfolio value
- `context.portfolioAssets` (Array) - Current holdings
- `context.marketCondition` (String) - Current market state

- `context.userRiskTolerance` (String) - User's risk level
- `context.recentTransactions` (Array of ObjectIds) - Recent trades

AI Metadata

- `aiModel` (String) - AI model used (e.g., gpt-4)
- `tokens.prompt` (Number) - Prompt tokens used
- `tokens.completion` (Number) - Completion tokens used
- `tokens.total` (Number) - Total tokens
- `responseTime` (Number) - Response time in milliseconds

Message Metadata

- `intent` (String) - general_question | portfolio_advice | market_analysis | etc.
- `suggestedActions` (Array) - Action suggestions with labels and URLs

Feedback

- `helpful` (Boolean) - User feedback
- `feedbackNote` (String) - Additional feedback

Status

- `isError` (Boolean) - Error status
- `errorMessage` (String) - Error details
- `timestamp` (Date) - Message timestamp

Indexes

- `user + conversationId + timestamp` (compound)
- `user + timestamp` (descending)
- `conversationId + timestamp` (ascending)

Methods

- **static** `getConversationHistory(userId, conversationId, limit)` - Get messages
- **static** `getRecentConversations(userId, limit)` - Get conversation list
- **static** `generateConversationId()` - Create new conversation ID
- `markAsHelpful(isHelpful, note)` - Record user feedback

UserSettings Model

Location: `backend/src/models/UserSettings.js`

Purpose

Stores advanced user settings and preferences separate from the User model.

Schema Fields

Trading Settings

- `trading.defaultOrderType` (String) - market | limit
- `trading.confirmBeforeTrade` (Boolean) - Confirmation required
- `trading.defaultSlippage` (Number) - Slippage percentage
- `trading.autoRebalance` (Boolean) - Auto-rebalancing enabled
- `trading.rebalanceThreshold` (Number) - Deviation threshold %

- `trading.rebalanceFrequency` (String) - daily | weekly | monthly | manual

Display Settings

- `display.currency` (String) - Display currency
- `display.numberFormat` (String) - Number format style
- `display.dateFormat` (String) - Date format
- `display.timeFormat` (String) - 12h | 24h
- `display.chartType` (String) - candlestick | line | area
- `display.showPortfolioValue` (Boolean) - Show/hide value
- `display.showProfitLoss` (Boolean) - Show/hide P/L
- `display.compactMode` (Boolean) - Compact UI mode

Alert Preferences

- `alertPreferences.priceAlerts` - Sound, vibration settings
- `alertPreferences.portfolioAlerts` - Daily/weekly thresholds
- `alertPreferences.newsAlerts` - Keywords, sources

API Keys

- `apiKeys` (Array) - Exchange API keys
  - `exchange` - coinbase | binance | kraken | gemini | other
  - `label` - Custom label
  - `apiKey` - API key (should be encrypted)
  - `apiSecret` - API secret (should be encrypted)
  - `isActive` - Active status
  - `permissions` - Array of permissions
  - `lastUsed` - Last used timestamp
  - `createdAt` - Added date

AI Assistant Settings

- `aiAssistant.enabled` (Boolean) - AI enabled
- `aiAssistant.personality` (String) - professional | friendly | concise | detailed
- `aiAssistant.contextAware` (Boolean) - Use portfolio data
- `aiAssistant.suggestionLevel` (String) - minimal | moderate | aggressive

Privacy & Data

- `privacy.shareAnonymousData` (Boolean) - Analytics consent
- `privacy.allowAnalytics` (Boolean) - Tracking consent
- `privacy.marketingEmails` (Boolean) - Marketing opt-in
- `privacy.activityTracking` (Boolean) - Activity logs

Advanced Features

- `advanced.developerMode` (Boolean) - Developer features
- `advanced.betaFeatures` (Boolean) - Beta access
- `advanced.debugMode` (Boolean) - Debug mode

Backup & Export

- `backup.autoBackup` (Boolean) - Auto-backup enabled
- `backup.backupFrequency` (String) - daily | weekly | monthly
- `backup.lastBackup` (Date) - Last backup timestamp

Timestamps

- `createdAt` (Date) - Settings created
- `updatedAt` (Date) - Last updated

Indexes

- `user` (unique)

Methods

- **static** `getOrCreate(userId)` - Get existing or create new settings
- `addApiKey(exchangeData)` - Add new API key
- `removeApiKey(apiKeyId)` - Remove API key

Hooks

- **pre('save')** - Update timestamp

Database Relationships

```
User (1) ──< (Many) Portfolio
User (1) ──< (Many) Transaction
User (1) ──< (Many) PriceAlert
User (1) ──< (Many) Notification
User (1) ──< (Many) ChatMessage
User (1) ─── (1) UserSettings

Transaction (Many) ──> (1) User
Portfolio (Many) ──> (1) User
```

Best Practices

Performance

1. All foreign keys have indexes
2. Compound indexes for common query patterns
3. TTL indexes for auto-cleanup (Notifications)
4. Sparse indexes on optional fields

Security

1. Passwords are hashed with bcrypt
2. API keys should be encrypted (TODO in production)
3. Sensitive fields excluded from API responses
4. Email verification tokens expire

Data Integrity

1. Required fields enforced at schema level
2. Enum validation for categorical data
3. Unique indexes prevent duplicates
4. Timestamps track data lifecycle

Scalability

1. Pagination supported via indexes

2. Aggregation pipelines for analytics
3. Reference-based relationships
4. Minimal embedded documents

Migration Notes

For Existing Users

When deploying these schema changes:

1. **User Model**: Existing users will get default values for new fields
2. **Portfolio/Transaction**: Existing records get default values (0 for numbers, false for booleans)
3. **New Models**: No migration needed, created as needed

Recommended Migration Steps

1. Deploy new schema definitions
2. Run data migration script to populate default values
3. Update API endpoints to handle new fields
4. Update frontend to use new features

Future Enhancements

Planned Additions

1. **Achievement Model** - Gamification system
2. **Referral Model** - Referral program tracking
3. **Order Model** - Advanced order types (stop-loss, limit, etc.)
4. **News Model** - Curated crypto news
5. **SocialPost Model** - Community features

Optimization Ideas

1. Add Redis caching for frequently accessed data
2. Archive old transactions to separate collection
3. Implement sharding for user data
4. Add full-text search indexes

Support

For questions about the database schema:

- Check model files in `backend/src/models/`
- Review indexes for query optimization
- Test with sample data before production
- Monitor query performance with MongoDB profiler

Last Updated: December 25, 2024
Schema Version: 2.0.0