Crypto Price Monitor

Project Documentation

What This Application Does

This is a full-stack web application that provides real-time cryptocurrency price monitoring with intelligent alert systems. It allows users to track multiple cryptocurrencies simultaneously and receive instant notifications when specific market conditions are met.

Core Functionality

Real-Time Price Tracking

- Continuously fetches live cryptocurrency prices from CoinGecko API every 10 seconds
- Displays current prices, 24-hour percentage changes, market capitalization, and trading volume
- Uses WebSocket connections to push updates instantly to all connected users
- No page refresh required - prices update automatically in the browser

Intelligent Alert System

- Create custom price alerts with multiple condition types:
    - Price threshold alerts: Trigger when price goes above or below a specific dollar amount
    - Percentage change alerts: Trigger when price increases or decreases by a set percentage
- Automatic email notifications sent via SendGrid when alerts are triggered
- Smart alert deactivation prevents spam by disabling alerts after they trigger
- Personal alert dashboard for managing multiple alerts per user

Secure User Management

- Complete authentication system with user registration and login
- Email verification required for account activation
- Password reset functionality with secure token-based system
- JWT-based session management with secure cookies
- Personal user profiles with notification preferences

Data Storage & Analytics

- Stores historical price data for trend analysis
- Maintains user alert statistics and performance tracking
- Redis caching for improved API response times
- MongoDB for persistent data storage

Supported Cryptocurrencies

The application currently monitors 10 major cryptocurrencies:

- Bitcoin (BTC)
- Ethereum (ETH)
- Binance Coin (BNB)
- Cardano (ADA)
- Solana (SOL)
- Polkadot (DOT)
- Dogecoin (DOGE)
- Avalanche (AVAX)
- Polygon (MATIC)
- Chainlink (LINK)

System Architecture & Data Flow

Architecture Overview

Frontend Layer (React + TypeScript)

- Single Page Application with real-time WebSocket connections
- Responsive design that works on desktop, tablet, and mobile
- Real-time connection status indicators
- Interactive price cards with live updates

Backend Layer (Node.js + Express)

- RESTful API with TypeScript for type safety
- WebSocket server for real-time communications

- Scheduled cron jobs for price updates and alert checking
- Comprehensive logging and error handling

Data Layer

- MongoDB for persistent storage (users, alerts, price history)
- Redis for caching and session management
- External API integration with CoinGecko

Email Services

- SendGrid integration for transactional emails
- Email verification, password reset, and alert notifications

Real-Time Data Flow

1. **Price Updates**: Cron job runs every 10 seconds → Fetches data from CoinGecko API → Processes and caches data → Broadcasts to all connected WebSocket clients → Frontend updates prices
2. **Alert Processing**: Price update received → Checks against all active user alerts → Sends email notification if conditions met → Broadcasts alert notification → Updates alert status
3. **User Actions**: Frontend sends API request → Backend validates and processes → Database updated → Response sent to frontend → UI updated

WebSocket Implementation

The application uses Socket.IO for real-time communications:

- Automatic reconnection on connection loss
- Connection status monitoring and display
- Targeted broadcasting to subscribed users
- Error handling and fallback to HTTP polling

Technical Implementation

Backend Technologies

Core Framework

- Node.js with Express.js for the web server

- TypeScript for type safety and better development experience
- Socket.IO for real-time bidirectional communication

Database & Caching

- MongoDB with Mongoose ODM for data modeling and validation
- Redis for high-performance caching and session storage
- Database indexing for optimized query performance

Authentication & Security

- JWT (JSON Web Tokens) for stateless authentication
- bcryptjs for secure password hashing (12 salt rounds)
- Joi for comprehensive input validation
- Helmet.js for security headers
- Rate limiting to prevent API abuse

External Integrations

- CoinGecko API for cryptocurrency price data
- SendGrid API for email delivery services
- Winston for structured logging

Frontend Technologies

UI Framework

- React 18 with TypeScript for type-safe component development
- Socket.IO client for real-time updates
- Modern CSS with CSS Grid and Flexbox for responsive layouts

State Management

- React hooks (useState, useEffect) for local component state
- Context API for user authentication state
- Real-time state updates via WebSocket events

User Experience

- Responsive design principles for all screen sizes
- Toast notifications for user feedback

- Loading states and error handling
- Real-time connection status indicators

Database Schema

Users Collection

```
{
  email: String (unique, required),
  password: String (hashed, required),
  firstName: String (required),
  lastName: String (required),
  isEmailVerified: Boolean,
  emailVerificationToken: String,
  preferences: {
    emailNotifications: Boolean,
    theme: String
  },
  createdAt: Date,
  updatedAt: Date
}
```

Alerts Collection

```
{
  userId: ObjectId (reference to Users),
  symbol: String (crypto symbol, required),
  condition: String (above|below|percent_increase|percent_decrease),
  targetPrice: Number (for price alerts),
  percentageChange: Number (for percentage alerts),
  isActive: Boolean,
  createdAt: Date,
  updatedAt: Date
```

}

PriceHistory Collection

{

symbol: String (crypto symbol),

price: Number (current price),

timestamp: Date,

volume_24h: Number,

market_cap: Number

}

Installation & Configuration Guide

Prerequisites

System Requirements

- Node.js version 18 or higher
- MongoDB database (local installation or cloud service like MongoDB Atlas)
- Redis server (local installation or cloud service)
- SendGrid account for email services

Development Tools (recommended)

- Visual Studio Code or similar IDE
- MongoDB Compass for database management
- Redis CLI for cache management

Step-by-Step Installation

1. Project Setup

Clone repository

git clone <repository-url>

cd crypto-price-monitor

Install backend dependencies

cd backend

npm install

Install frontend dependencies

cd ../frontend

npm install

2. Backend Configuration

Create backend/.env file with the following variables:

Server Configuration

PORT=3001

NODE_ENV=development

Database Connections

MONGODB_URI=mongodb://localhost:27017/crypto-monitor

REDIS_URI=redis://localhost:6379

Authentication

JWT_SECRET=your-super-secret-jwt-key-minimum-32-characters-long

JWT_EXPIRES_IN=30d

JWT_COOKIE_EXPIRE=30

External APIs

COINGECKO_API_KEY=your-coingecko-api-key-optional

Email Configuration (SendGrid)

EMAIL_HOST=smtp.sendgrid.net

EMAIL_PORT=587

EMAIL_SECURE=false

EMAIL_USERNAME=apikey

EMAIL_PASSWORD=your-sendgrid-api-key

EMAIL_FROM_NAME=Crypto Monitor

EMAIL_FROM=your-verified-email@yourdomain.com

Frontend URLs

FRONTEND_URL=http://localhost:3000

CLIENT_URL=http://localhost:3000

Performance Settings

PRICE_UPDATE_INTERVAL=10000

CACHE_DEFAULT_TTL=60000

API_RATE_LIMIT_WINDOW_MS=900000

API_RATE_LIMIT_MAX=100

3. Frontend Configuration

Create frontend/.env file:

REACT_APP_API_URL=http://localhost:3001/api

REACT_APP_WS_URL=http://localhost:3001

4. Database Setup

Ensure MongoDB and Redis are running:

Check MongoDB connection

mongosh mongodb://localhost:27017/crypto-monitor

Check Redis connection

redis-cli ping

5. SendGrid Email Setup

1. Create SendGrid account at https://sendgrid.com
2. Complete sender identity verification (verify your email or domain)

3. Generate API key:
    - Navigate to Settings → API Keys
    - Click "Create API Key"
    - Choose "Restricted Access"
    - Enable "Mail Send" permissions only
    - Copy the generated key
4. Update EMAIL_PASSWORD in .env with the API key
5. Set EMAIL_FROM to your verified sender email

6. Start the Application

Start backend server (from backend directory)

npm run dev

Start frontend development server (from frontend directory)

npm start

7. Verify Installation

- Frontend: http://localhost:3000
- Backend API: http://localhost:3001
- Health check: http://localhost:3001/health
- WebSocket connection should show "Live" status in the navigation

---

API Reference

Authentication Endpoints

POST /api/auth/register

- Creates new user account
- Sends email verification
- Returns JWT token and user data

POST /api/auth/login

- Authenticates existing user

- Returns JWT token and user data

GET /api/auth/me

- Returns current authenticated user information
- Requires valid JWT token

GET /api/auth/verify-email/:token

- Verifies user email address
- Activates account for email notifications

POST /api/auth/resend-verification

- Sends new email verification link
- Requires authentication

POST /api/auth/logout

- Invalidates current session
- Clears authentication cookies

Price Data Endpoints

GET /api/prices

- Returns current prices for all supported cryptocurrencies
- Includes price, market cap, volume, and 24h change data
- Cached for performance

GET /api/prices/:symbol

- Returns detailed price data for specific cryptocurrency
- Symbol parameter (e.g., 'bitcoin', 'ethereum')

GET /api/prices/:symbol/history

- Returns historical price data
- Query parameters: days (default: 7), limit (default: 100)

Alert Management Endpoints

GET /api/alerts

- Returns all alerts for authenticated user
- Includes active and inactive alerts

POST /api/alerts

- Creates new price alert
- Requires authentication
- Validates alert conditions and parameters

PUT /api/alerts/:id

- Updates existing alert
- User can only update their own alerts

DELETE /api/alerts/:id

- Deletes specific alert
- User can only delete their own alerts

GET /api/alerts/stats

- Returns alert statistics for authenticated user
- Includes total alerts, active alerts, and alerts by symbol

WebSocket Events

Client → Server Events

- subscribe-to-symbol - Subscribe to specific cryptocurrency updates
- unsubscribe-from-symbol - Unsubscribe from cryptocurrency updates

Server → Client Events

- price-update - Broadcast of current prices for all cryptocurrencies
- alert-triggered - Notification when user's alert conditions are met
- connection-status - Connection status updates and confirmations

---

User Experience Flow

New User Journey

1. **Registration**: User creates account with email and password
2. **Email Verification**: System sends verification email via SendGrid
3. **Account Activation**: User clicks email link to verify account
4. **Dashboard Access**: User logs in and sees live cryptocurrency prices

5. **Alert Creation**: User creates first price alert with preferred conditions
6. **Real-Time Monitoring**: User receives live price updates via WebSocket
7. **Alert Notification**: When conditions are met, user receives email and browser notification

Returning User Journey

1. **Login**: User authenticates with existing credentials
2. **Dashboard**: Immediate access to live prices and connection status
3. **Alert Management**: View, edit, or create additional alerts
4. **Real-Time Updates**: Continuous price monitoring and notifications

---

Security & Performance Features

Security Implementation

Authentication Security

- Passwords hashed using bcrypt with 12 salt rounds
- JWT tokens with configurable expiration (default: 30 days)
- Secure HTTP-only cookies for session management
- Email verification required for account activation

API Security

- Rate limiting: 100 requests per 15-minute window per IP
- Input validation using Joi schemas on all endpoints
- CORS protection configured for specific frontend origins
- Security headers via Helmet.js middleware
- MongoDB injection protection through Mongoose validation

Data Protection

- User passwords never stored in plain text
- Sensitive configuration stored in environment variables
- Database queries use parameterized statements
- Authentication required for all user-specific endpoints

Performance Optimizations

Caching Strategy

- Redis caching for CoinGecko API responses (60-second TTL)
- Price data cached to reduce external API calls
- User session data cached for fast authentication

Real-Time Efficiency

- WebSocket connections for instant updates (no polling)
- Targeted broadcasting to subscribed users only
- Connection pooling for database operations
- Optimized MongoDB queries with proper indexing

API Rate Management

- CoinGecko API rate limit compliance (30 requests/minute)
- Request queuing to prevent API overuse
- Fallback to cached data if API is unavailable

Troubleshooting Guide

Common Issues & Solutions

Real-Time Updates Not Working

Symptoms: Prices not updating automatically, connection shows "Offline"

Debugging Steps:

1. Check browser console for WebSocket connection errors
2. Verify backend server is running on port 3001
3. Test health endpoint: curl http://localhost:3001/health
4. Check for CORS errors in browser developer tools
5. Verify FRONTEND_URL in backend .env matches frontend URL exactly

Solutions:

- Restart backend server
- Clear browser cache and cookies

- Check firewall settings blocking WebSocket connections
- Verify environment variables are loaded correctly

Email Verification Not Working

Symptoms: Verification emails not received, registration completes but no email

Debugging Steps:

1. Check backend logs for email service errors
2. Verify SendGrid API key has Mail Send permissions
3. Confirm sender email is verified in SendGrid dashboard
4. Check spam/junk folders for emails
5. Test SendGrid configuration using SendGrid's testing tools

Solutions:

- Regenerate SendGrid API key with proper permissions
- Verify sender identity in SendGrid account
- Check EMAIL_FROM matches verified sender in SendGrid
- Monitor SendGrid activity dashboard for delivery status

Database Connection Issues

Symptoms: Application crashes on startup, authentication fails

Debugging Steps:

1. Verify MongoDB is running: mongosh mongodb://localhost:27017
2. Check Redis connection: redis-cli ping
3. Review connection strings in .env file
4. Check database server status and available disk space

Solutions:

- Start MongoDB and Redis services
- Update connection strings for cloud databases
- Check network connectivity to database servers
- Verify database credentials and permissions

Price Data Not Loading

Symptoms: Empty price cards, "No price data available" message

Debugging Steps:

1. Check CoinGecko API status: curl https://api.coingecko.com/api/v3/ping
2. Review backend logs for API errors
3. Verify supported coins list in backend configuration
4. Check API rate limiting logs

Solutions:

- Wait for CoinGecko API to recover if experiencing outages
- Add CoinGecko API key if hitting rate limits
- Verify supported coins are available in CoinGecko API
- Check network connectivity from server to CoinGecko

Development Debugging

Enable Detailed Logging

```
# Backend - set LOG_LEVEL=debug in .env
LOG_LEVEL=debug

# Frontend - enable Socket.IO debug logs
localStorage.debug = 'socket.io-client:*'
```

Health Check Endpoints

- Backend health: GET /api/health
- Socket.IO stats: GET /api/socket/stats
- Database connectivity included in health check response

This crypto price monitor provides a professional-grade solution for real-time cryptocurrency tracking with enterprise-level security, performance optimizations, and user experience design.