

Automated Crypto Trading Application - Requirements Document

1. Executive Summary

1.1 Project Overview

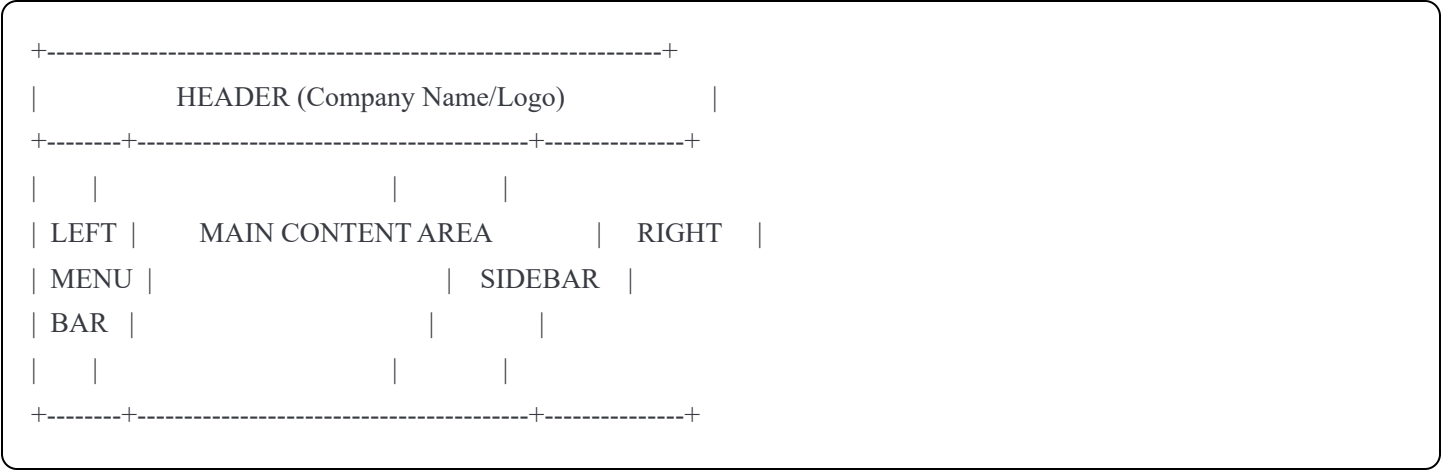
Multi-user automated cryptocurrency trading application that uses multi-timeframe analysis (Daily, Hourly, 5-Minute) to identify rise/fall cycles and execute automated trades with Take Profit (TP) and Stop Loss (SL) parameters.

1.2 Core Trading Logic

- **Daily Data:** Identify top moving cryptocurrencies with highest daily gains and multiple rise/fall cycles
- **Hourly Data:** Determine exact timing of rise/fall cycle transitions
- **5-Minute Data:** Execute trades on selected top 10 cryptocurrencies to capture intraday cycles
- **Automated Execution:** System automatically executes multiple trades across all detected rise/fall cycles

2. Application Layout

2.1 Screen Structure



2.2 Header Section

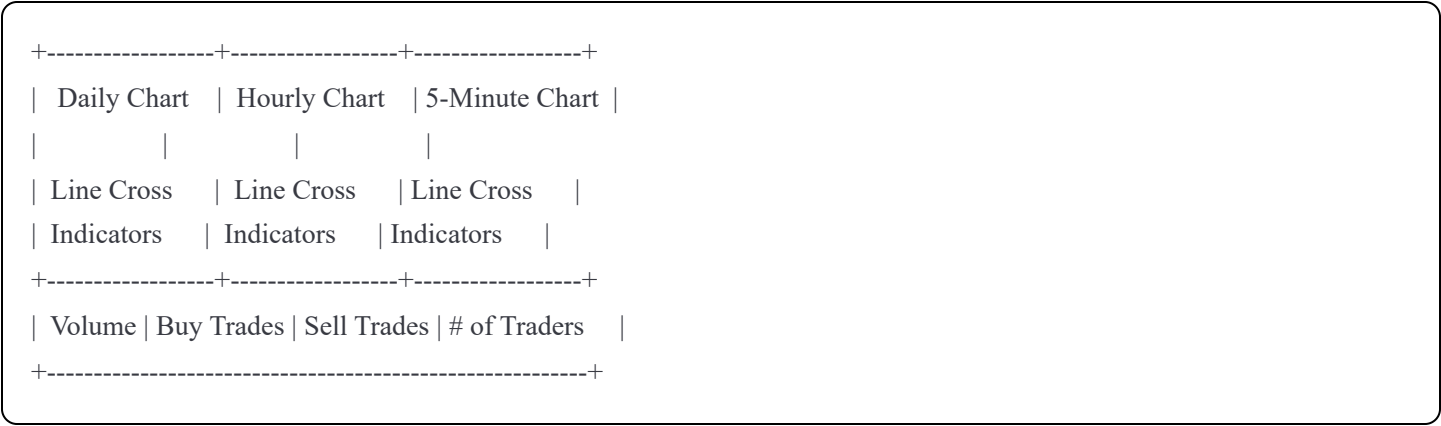
- Company name/logo (left)
- Total portfolio value (center)
- User account menu (right)
- Current date/time display

2.3 Left Menu Bar

- Dashboard (Home)
- Live Trading
- Trade History
- Deposits/Withdrawals
- Settings
 - Trade Parameters
 - API Configuration
 - User Management
- Reports
- Admin Panel (admin users only)

2.4 Main Content Area Layout

Top Section - Multi-Timeframe Charts:



Bottom Section - Crypto Performance Dashboard:

```

+-----+-----+-----+-----+-----+-----+-----+
| BTC | ETH | BNB | Gainer| Gainer| Loser | Loser |
|     |     |     | #1  | #2  | #1  | #2  |
| Graph| Graph| Graph| Graph| Graph| Graph| Graph|
| Price| Price| Price| Price| Price| Price| Price|
| Gain%| Gain%| Gain%| Gain%| Gain%| Gain%| Gain%|
| Trades| Trades| Trades| Trades| Trades| Trades| Trades|
| Buy  | Buy  | Buy  | Buy  | Buy  | Buy  | Buy  |
| Sell | Sell | Sell | Sell | Sell | Sell | Sell |
| Traders|Traders|Traders|Traders|Traders|Traders|Traders|
+-----+-----+-----+-----+-----+-----+-----+

```

2.5 Right Sidebar

- **Trade Setup Window**
 - Selected Crypto Symbol
 - Trade Amount (USD)
 - Take Profit % (TP%)
 - Stop Loss % (SL%)
 - Start/Stop Auto-Trading Toggle
 - Current Status Indicator
 - **Active Trades Panel**
 - List of currently open positions
 - Entry price, current P&L
 - TP/SL targets
 - **Quick Stats**
 - Today's trades executed
 - Today's win rate
 - Today's total P&L
 - Account balance
-

3. Trading Strategy & Automation

3.1 Multi-Timeframe Analysis Process

Step 1: Daily Screening

- Scan all available cryptocurrencies
- Calculate daily gain/loss percentage
- Count number of rise/fall cycles in the day
- Rank by: highest daily gain AND most rise/fall cycles
- Output: Top 10 cryptocurrencies for potential trading

Step 2: Hourly Timing

- For selected top 10 cryptos, analyze hourly candlesticks
- Identify exact hour when rise cycle begins (line crossover)
- Identify exact hour when fall cycle begins (line crossover)
- Validate cycle strength using volume and trader count

Step 3: 5-Minute Execution

- Monitor 5-minute candlesticks for selected cryptos
- Detect rise cycle start using indicator line crossovers
- Execute BUY order automatically
- Monitor for fall cycle start indicator
- Execute SELL order automatically when TP hit OR SL triggered

3.2 Rise/Fall Cycle Detection

Indicators Used:

1. Moving Average Crossovers

- Fast MA (9-period) crosses above Slow MA (21-period) = Rise Cycle Start
- Fast MA (9-period) crosses below Slow MA (21-period) = Fall Cycle Start

2. Supporting Indicators

- Volume spike confirmation
- RSI (Relative Strength Index) > 50 for rise, < 50 for fall

- MACD histogram turning positive/negative

Visual Display:

- Green vertical line on chart = Rise Cycle Start (BUY signal)
- Red vertical line on chart = Fall Cycle Start (SELL signal)
- Shaded region = Active rise cycle (potential holding period)

3.3 Automated Trade Execution Logic

FOR EACH selected crypto in top 10:

WHILE auto-trading is enabled:

// Monitor 5-minute chart

IF rise_cycle_detected():

 entry_price = current_market_price

 tp_target = entry_price * (1 + TP_percentage/100)

 sl_target = entry_price * (1 - SL_percentage/100)

EXECUTE BUY order with trade_amount

LOG trade to database

MONITOR position:

 IF current_price >= tp_target:

 EXECUTE SELL order

 RECORD win

 RESET for next cycle

 ELSE IF current_price <= sl_target:

 EXECUTE SELL order

 RECORD loss

 RESET for next cycle

 ELSE IF fall_cycle_detected():

 EXECUTE SELL order at market

 RECORD result

 RESET for next cycle

WAIT 5 minutes

REPEAT

3.4 Trade Parameters Configuration

Users can set:

- **Trade Amount:** USD amount per trade (\$10 - \$10,000)
 - **Take Profit %:** Target profit percentage (1% - 50%)
 - **Stop Loss %:** Maximum acceptable loss (0.5% - 20%)
 - **Max Concurrent Trades:** Maximum number of simultaneous positions (1-10)
 - **Trading Hours:** Specify active trading hours (24/7 or custom)
 - **Crypto Selection:** Choose specific cryptos or use auto-selection
-

4. Database Schema

4.1 Existing Tables (Already Created)

Table 1: daily_crypto_data

sql

Fields to capture from your existing table:

- id (primary key)
- symbol (e.g., BTC, ETH, BNB)
- date
- open_price
- high_price
- low_price
- close_price
- volume
- trade_count
- buy_volume
- sell_volume
- trader_count
- vwap (volume weighted average price)
- num_trades
- rsi (Relative Strength Index)
- macd
- macd_signal
- macd_histogram
- bb_upper (Bollinger Band Upper)
- bb_middle (Bollinger Band Middle)
- bb_lower (Bollinger Band Lower)
- sma_9 (Simple Moving Average 9)
- sma_21 (Simple Moving Average 21)
- ema_9 (Exponential Moving Average 9)
- ema_21 (Exponential Moving Average 21)
- created_at

Table 2: hourly_crypto_data

sql

Same fields as daily_crypto_data but:

- timestamp (instead of date)
- Hourly granularity

Table 3: minute5_crypto_data

sql

Same fields as daily_crypto_data but:

- timestamp (with minute precision)
- 5-minute granularity

4.2 New Tables Required

Table 4: users

```
sql

CREATE TABLE users (
  user_id SERIAL PRIMARY KEY,
  username VARCHAR(50) UNIQUE NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  full_name VARCHAR(100),
  role ENUM('user', 'admin') DEFAULT 'user',
  account_balance DECIMAL(15, 2) DEFAULT 0.00,
  is_active BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  last_login TIMESTAMP
);
```

Table 5: trade_parameters

```
sql

CREATE TABLE trade_parameters (
  param_id SERIAL PRIMARY KEY,
  user_id INTEGER REFERENCES users(user_id),
  crypto_symbol VARCHAR(10),
  trade_amount DECIMAL(10, 2),
  take_profit_percentage DECIMAL(5, 2),
  stop_loss_percentage DECIMAL(5, 2),
  max_concurrent_trades INTEGER DEFAULT 1,
  auto_trading_enabled BOOLEAN DEFAULT FALSE,
  trading_hours_start TIME,
  trading_hours_end TIME,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Table 6: executed_trades

sql

```
CREATE TABLE executed_trades (  
  trade_id SERIAL PRIMARY KEY,  
  user_id INTEGER REFERENCES users(user_id),  
  crypto_symbol VARCHAR(10),  
  trade_type ENUM('BUY', 'SELL'),  
  entry_price DECIMAL(15, 8),  
  exit_price DECIMAL(15, 8),  
  quantity DECIMAL(15, 8),  
  trade_amount DECIMAL(15, 2),  
  profit_loss DECIMAL(15, 2),  
  profit_loss_percentage DECIMAL(8, 4),  
  take_profit_target DECIMAL(15, 8),  
  stop_loss_target DECIMAL(15, 8),  
  execution_reason ENUM('TP_HIT', 'SL_HIT', 'FALL_CYCLE', 'MANUAL'),  
  timeframe_detected VARCHAR(10), -- 'daily', 'hourly', '5min'  
  cycle_indicators JSON, -- Store indicator values at trade time  
  executed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  status ENUM('open', 'closed') DEFAULT 'open'  
);
```

Table 7: deposits_withdrawals

sql

```
CREATE TABLE deposits_withdrawals (  
  transaction_id SERIAL PRIMARY KEY,  
  user_id INTEGER REFERENCES users(user_id),  
  transaction_type ENUM('DEPOSIT', 'WITHDRAWAL'),  
  amount DECIMAL(15, 2),  
  currency VARCHAR(10) DEFAULT 'USD',  
  payment_method VARCHAR(50),  
  status ENUM('pending', 'completed', 'failed') DEFAULT 'pending',  
  transaction_reference VARCHAR(100),  
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  completed_at TIMESTAMP  
);
```

Table 8: cycle_detection_log

sql

```
CREATE TABLE cycle_detection_log (
  cycle_id SERIAL PRIMARY KEY,
  crypto_symbol VARCHAR(10),
  timeframe VARCHAR(10), -- 'daily', 'hourly', '5min'
  cycle_type ENUM('RISE', 'FALL'),
  detection_timestamp TIMESTAMP,
  price_at_detection DECIMAL(15, 8),
  sma_9_value DECIMAL(15, 8),
  sma_21_value DECIMAL(15, 8),
  volume DECIMAL(20, 2),
  rsi_value DECIMAL(5, 2),
  macd_value DECIMAL(10, 4),
  confidence_score DECIMAL(3, 2), -- 0.00 to 1.00
  trade_triggered BOOLEAN DEFAULT FALSE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Table 9: daily_performance_summary

```
sql

CREATE TABLE daily_performance_summary (
  summary_id SERIAL PRIMARY KEY,
  user_id INTEGER REFERENCES users(user_id),
  summary_date DATE,
  total_trades INTEGER DEFAULT 0,
  winning_trades INTEGER DEFAULT 0,
  losing_trades INTEGER DEFAULT 0,
  total_profit_loss DECIMAL(15, 2) DEFAULT 0.00,
  win_rate DECIMAL(5, 2),
  largest_win DECIMAL(15, 2),
  largest_loss DECIMAL(15, 2),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UNIQUE(user_id, summary_date)
);
```

Table 10: system_settings

```
sql
```

```
CREATE TABLE system_settings (  
  setting_id SERIAL PRIMARY KEY,  
  setting_key VARCHAR(50) UNIQUE NOT NULL,  
  setting_value TEXT,  
  description TEXT,  
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  updated_by INTEGER REFERENCES users(user_id)  
);
```

5. Feature Specifications

5.1 Multi-Timeframe Chart Display

Daily Chart (Left Panel):

- Candlestick chart with last 90 days
- Display indicators: SMA 9, SMA 21, Volume bars
- Green/Red vertical lines for rise/fall cycle detection
- Symbol selector dropdown
- Auto-refresh every 24 hours

Hourly Chart (Center Panel):

- Candlestick chart with last 7 days (168 hours)
- Display indicators: EMA 9, EMA 21, Volume bars
- Green/Red vertical lines for rise/fall cycle detection
- Linked to daily chart selection
- Auto-refresh every hour

5-Minute Chart (Right Panel):

- Candlestick chart with last 24 hours (288 candles)
- Display indicators: EMA 9, EMA 21, MACD, RSI
- Green/Red vertical lines for rise/fall cycle detection
- Linked to daily chart selection
- Auto-refresh every 5 minutes

- Real-time trade execution markers

Below Charts - Metrics Bar:

| Volume: \$XXX,XXX,XXX | Buy Trades: XXX | Sell Trades: XXX | Traders: X,XXX |

5.2 Crypto Performance Dashboard

Display Grid (7 Cards):

Each card shows:

```
+-----+
| [Symbol] BTC |
| [Mini Chart] |
|           |
| $XX,XXX.XX  |
| +X.XX% (24h) |
|           |
| Daily | Hourly | Monthly
| +X.X% | +X.X% | +XX.X%
|           |
| Trades: XXX  |
| Buy: XXX     |
| Sell: XXX    |
| Traders: X,XXX |
+-----+
```

7 Cards:

1. Bitcoin (BTC)
2. Ethereum (ETH)
3. Binance Coin (BNB)
4. Top Gainer #1 (dynamic)
5. Top Gainer #2 (dynamic)
6. Top Gainer #3 (dynamic)
7. Top Gainer #4 (dynamic)
8. Top Loser #1 (dynamic)

9. Top Loser #2 (dynamic)

10. Top Loser #3 (dynamic)

5.3 Trade Setup Window (Right Sidebar)

+-----+
| TRADE SETUP |
+-----+
| Symbol: [BTC ▼] |
| |
| Trade Amount (USD) |
| [\$500.00] |
| |
| Take Profit % |
| [3.00%] |
| |
| Stop Loss % |
| [1.50%] |
| |
| Max Concurrent Trades |
| [3] |
| |
| [✓] Auto-Trading ON |
| |
| [Start Trading] |
+-----+

5.4 All Trades Window

Table Columns:

- Trade ID
- Date/Time
- Symbol
- Type (BUY/SELL)
- Entry Price
- Exit Price
- Quantity
- Amount

- P&L (\$)
- P&L (%)
- Exit Reason (TP/SL/Fall Cycle)
- Status

Features:

- Sort by any column
- Filter by: Date range, Symbol, Status, P&L
- Export to CSV
- Pagination (50 trades per page)
- Color coding: Green (profitable), Red (loss)

5.5 Deposits/Withdrawals Window

Table Columns:

- Transaction ID
- Date/Time
- Type (Deposit/Withdrawal)
- Amount
- Currency
- Method
- Status
- Reference

Features:

- Add new deposit button
- Add new withdrawal button
- Status filter
- Date range filter
- Export to CSV

5.6 Trade Results Summary Window

Display Metrics:

+-----+	
TRADING PERFORMANCE SUMMARY	
+-----+	
Time Period: [Last 30 Days ▼]	
Total Trades: XXX	Win Rate: XX.X%
Winning Trades: XXX	Total Profit: \$X,XXX
Losing Trades: XXX	Total Loss: -\$XXX
Net P&L: \$X,XXX	Avg Trade: \$XX.XX
ROI: XX.XX%	Best Trade: \$XXX
Sharpe Ratio: X.XX	Worst Trade: -\$XXX
+-----+	
[Performance Chart - Cumulative P&L Over Time]	
+-----+	

6. Admin Features

6.1 User Management

- View all users
- Add/Edit/Deactivate users
- Reset user passwords
- View user trading activity
- Adjust user account balances (for testing)

6.2 System Monitoring

- View all active trades across all users
- Monitor system performance
- View API connection status
- Database health check

- Error logs viewer

6.3 Configuration Management

- Manage trading pairs (add/remove cryptos)
 - Set system-wide trading limits
 - Configure indicator parameters
 - API key management (Kraken Pro)
 - Backup/restore database
-

7. Technical Implementation

7.1 Technology Stack

Frontend:

- React.js with TypeScript
- TailwindCSS for styling
- Recharts or Chart.js for candlestick charts
- WebSocket for real-time data

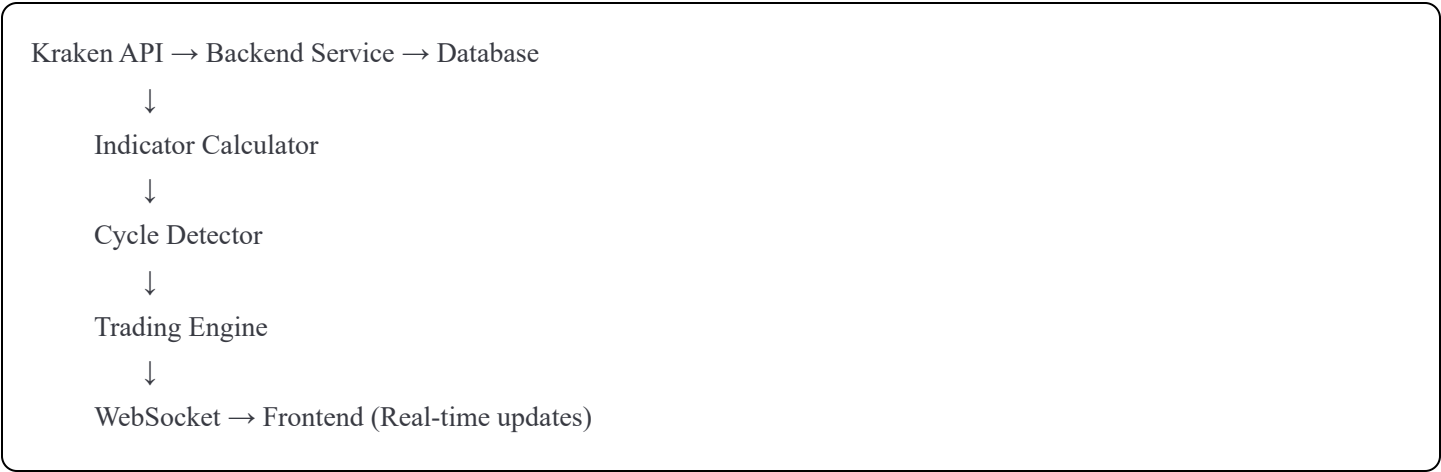
Backend:

- Node.js with Express OR Python with FastAPI
- WebSocket server for real-time updates
- PostgreSQL database
- Redis for caching and real-time data

Data Processing:

- Scheduled jobs (cron) for:
 - Fetching data from Kraken API
 - Calculating indicators
 - Running cycle detection algorithm
 - Executing trades

7.2 Real-Time Data Flow



7.3 Trading Engine Logic

Pseudo-code for Main Trading Loop:

```
python
def main_trading_loop():
    while True:
        # Fetch real-time data from Kraken API
        data = fetch_real_time_data()

        # Calculate indicators
        indicators = calculate_indicators(data)

        # Detect trading cycles
        cycle_detected = detect_cycle(indicators)

        # Execute trading strategy
        if cycle_detected:
            execute_trade_strategy(data)

        # Send real-time updates to Frontend
        send_real_time_updates(data)

        # Sleep for a short duration to avoid busy-waiting
        time.sleep(0.1)
```

Run every 5 minutes

def trading_engine_cycle():

Get all users with auto-trading enabled

active_users = get_users_with_autotrading()

for user **in** active_users:

Get user's trade parameters

params = get_trade_parameters(user.id)

Get selected cryptos (top 10 or user selected)

cryptos = get_selected_cryptos(user.id)

for crypto **in** cryptos:

Check if user has capacity for new trade

open_positions = count_open_positions(user.id, crypto)

if open_positions >= params.max_concurrent_trades:

continue

Get latest 5-min data

latest_data = get_latest_5min_data(crypto)

Detect cycle

cycle = detect_cycle(latest_data)

If rise cycle detected and no open position

if cycle == 'RISE' **and** open_positions == 0:

execute_buy_order(

 user_id=user.id,

 symbol=crypto,

 amount=params.trade_amount,

 tp_percentage=params.take_profit_percentage,

 sl_percentage=params.stop_loss_percentage

)

Check existing positions for exit conditions

positions = get_open_positions(user.id, crypto)

for position **in** positions:

 current_price = latest_data.close_price

Check TP

if current_price >= position.tp_target:

 execute_sell_order(position, reason='TP_HIT')

```
# Check SL
elif current_price <= position.sl_target:
    execute_sell_order(position, reason='SL_HIT')

# Check fall cycle
elif cycle == 'FALL':
    execute_sell_order(position, reason='FALL_CYCLE')
```

7.4 Cycle Detection Algorithm

python

```
def detect_cycle(data_array):
```

```
    """
```

Detect rise or fall cycle using multiple indicators

Parameters:

- data_array: Array of candlestick data with indicators

Returns: 'RISE', 'FALL', or 'NEUTRAL'

```
    """
```

Get last 3 candles for confirmation

```
current = data_array[-1]
```

```
previous = data_array[-2]
```

Primary Signal: Moving Average Crossover

```
sma_9_current = current.sma_9
```

```
sma_21_current = current.sma_21
```

```
sma_9_previous = previous.sma_9
```

```
sma_21_previous = previous.sma_21
```

Detect crossover

```
bullish_crossover = (sma_9_previous <= sma_21_previous and  
                    sma_9_current > sma_21_current)
```

```
bearish_crossover = (sma_9_previous >= sma_21_previous and  
                    sma_9_current < sma_21_current)
```

Confirmation indicators

```
volume_spike = current.volume > (sum([d.volume for d in data_array[-10:]]) / 10) * 1.2
```

```
rsi = current.rsi
```

```
macd_positive = current.macd_histogram > 0
```

Rise Cycle Detection

```
if bullish_crossover:
```

```
    confidence = 0.6
```

```
    if rsi > 50:
```

```
        confidence += 0.2
```

```
    if macd_positive:
```

```
        confidence += 0.1
```

```
    if volume_spike:
```

```
        confidence += 0.1
```

```
    if confidence >= 0.7:
```

```
        log_cycle_detection('RISE', current, confidence)
    return 'RISE'

# Fall Cycle Detection
elif bearish_crossover:
    confidence = 0.6
    if rsi < 50:
        confidence += 0.2
    if not macd_positive:
        confidence += 0.1
    if volume_spike:
        confidence += 0.1

    if confidence >= 0.7:
        log_cycle_detection('FALL', current, confidence)
        return 'FALL'

return 'NEUTRAL'
```

8. User Interface Flow

8.1 Login/Registration

- User enters credentials
- System authenticates
- Redirect to Dashboard

8.2 Dashboard View

- User sees overview of:
 - Current portfolio value
 - Today's P&L
 - Active trades count
 - 3 timeframe charts
 - Crypto performance grid

8.3 Setup Trading

1. User clicks "Settings" → "Trade Parameters"

2. Selects cryptocurrency
3. Sets Trade Amount, TP%, SL%
4. Enables Auto-Trading
5. System starts monitoring and trading automatically

8.4 Monitor Trades

- Real-time updates on charts
- Notifications when trades execute
- View active positions in right sidebar
- Check trade history window

8.5 Withdraw Profits

- User goes to Deposits/Withdrawals
 - Clicks "Withdraw"
 - Enters amount and payment method
 - Confirms withdrawal
 - System processes and updates balance
-

9. Security & Compliance

9.1 Authentication & Authorization

- Secure password hashing (bcrypt)
- JWT tokens for session management
- Role-based access control (User, Admin)
- API key encryption for Kraken integration

9.2 Data Security

- SSL/TLS for all communications
- Database encryption at rest
- Regular automated backups
- Secure API credential storage

9.3 Trading Safety Features

- Maximum trade size limits
 - Daily loss limits (circuit breaker)
 - Minimum balance requirements
 - Trade confirmation logs
 - Audit trail for all transactions
-

10. Testing Requirements

10.1 Testing Strategy

- Unit tests for cycle detection algorithm
- Integration tests for trading engine
- End-to-end tests for user workflows
- Load testing for multiple concurrent users
- Backtesting on historical data

10.2 Paper Trading Mode

- Users can test strategies without real money
 - Uses same logic as live trading
 - Tracks simulated P&L
 - Helps users optimize TP/SL parameters
-

11. Deployment & Monitoring

11.1 Deployment

- Docker containers for easy deployment
- CI/CD pipeline for updates
- Database migration scripts
- Environment-based configuration (dev, staging, prod)

11.2 Monitoring & Alerts

- System uptime monitoring
- Trading engine health checks
- API connection status
- Error rate tracking
- Email/SMS alerts for critical issues

11.3 Logging

- Trade execution logs
 - Cycle detection logs
 - Error logs
 - User activity logs
 - API request/response logs
-

12. Success Metrics

12.1 User Metrics

- Number of active traders
- Average trades per user per day
- User retention rate
- Average profit per user

12.2 System Metrics

- Trade execution success rate
- Cycle detection accuracy
- System uptime percentage
- Average response time

12.3 Trading Performance Metrics

- Overall win rate across all users

- Average P&L per trade
 - Number of TP hits vs SL hits
 - Cycle detection reliability
-

Appendix: Kraken API Integration

Data Endpoints Required

- OHLC data (Open, High, Low, Close) - All timeframes
- Order book data
- Recent trades
- Ticker information

Indicator Calculations

- Calculate in-house using fetched OHLC data:
 - SMA (9, 21 periods)
 - EMA (9, 21 periods)
 - RSI (14 periods)
 - MACD (12, 26, 9)
 - Bollinger Bands (20, 2)
 - Store calculated indicators in database tables
-

Document Version: 2.0 - Simplified for Automated Trading

Last Updated: [Date]

Status: Ready for Implementation