Alert Trigger Monitoring Documentation

This document describes the alert trigger monitoring system that automatically detects when price levels hit alert conditions and provides real-time notifications.

Overview

The alert trigger system provides:

- **Real-time Monitoring** - Checks alert conditions every second
- **Automatic Trigger Detection** - Detects when price levels are hit
- **Visual Notifications** - Shows popup notifications for triggered alerts
- **Status Tracking** - Tracks trigger count and last trigger time
- **Database Updates** - Updates alert status in real-time

How It Works

1. **Continuous Monitoring**

- Prices are refreshed every 1 second
- Alert conditions are checked against current prices
- Triggers are detected automatically

2. **Trigger Detection Logic**

```
# Example trigger conditions:
if operator == '>=' and current_price >= target_value:
    # Alert triggered
elif operator == '<=' and current_price <= target_value:
    # Alert triggered
```

3. **Status Updates**

- Alert count is incremented when triggered
- Last trigger time is recorded
- Last trigger price is stored

API Endpoints

Check Alert Triggers

Endpoint: `GET /alerts/check-triggers`

Description: Checks all active alerts against current prices and returns triggered alerts.

Authentication: Required

Response:

```
{
    "triggered_alerts": [
        {
            "uuid": "b88f3994-4d51-4266-b7d4-85ac2e2f7212",
            "name": "NIFTY 50 Breakout",
            "symbol": "NIFTY 50",
            "current_price": 24500.50,
            "target_price": 24400.00,
            "operator": ">=",
            "triggered_at": "2025-01-19T10:30:45.123456"
        }
    ],
    "count": 1,
    "success": true
}
```

Database Schema Updates

New Fields Added

Field	Type	Description
`last_triggered_at`	TEXT	Timestamp of last trigger
`last_triggered_price`	REAL	Price when last triggered
`alert_count`	INTEGER	Total number of triggers

Database Migration

The system automatically adds new columns to existing databases:

```
ALTER TABLE alerts ADD COLUMN last_triggered_at TEXT;
ALTER TABLE alerts ADD COLUMN last_triggered_price REAL;
```

User Interface Features

1. **Real-time Notifications**

When an alert is triggered, a notification appears:

- **Position**: Top-right corner of the page
- **Content**: Alert name, condition, current price
- **Duration**: Auto-removes after 10 seconds
- **Animation**: Slides in from the right

2. **Alert Status Display**

Alerts show trigger information:

- **Trigger Badge**: "🚨 Triggered X time(s)"
- **Last Trigger Time**: When it was last triggered
- **Visual Highlighting**: Triggered alerts have special styling

3. **Automatic Updates**

- **Price Updates**: Every 1 second
- **Alert Checking**: Every 1 second
- **Status Refresh**: When triggers are detected

Visual Indicators

Alert Notifications

```
.alert-notification {
    position: fixed;
    top: 20px;
    right: 20px;
    background: linear-gradient(135deg, #ff6b6b, #ee5a24);
    color: white;
    padding: 1rem;
    border-radius: 8px;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.3);
    animation: slideInRight 0.3s ease-out;
}
```

Trigger Badges

```
.trigger-badge {
    background: linear-gradient(135deg, #ff6b6b, #ee5a24);
    color: white;
    padding: 0.25rem 0.5rem;
    border-radius: 4px;
    font-size: 0.75rem;
    font-weight: bold;
}
```

Usage Examples

1. **Creating Alerts for Testing**

```
// Create an alert that's likely to trigger
const alertData = {
    "name": "NIFTY 50 Test",
    "lhs_exchange": "INDICES",
    "lhs_tradingsymbol": "NIFTY 50",
    "lhs_attribute": "LastTradedPrice",
    "operator": ">=",
    "rhs_type": "constant",
    "type": "simple",
    "rhs_constant": "24400"  // Set below current price
};
```

2. **Monitoring Triggers**

```
// Check for triggered alerts
fetch('/alerts/check-triggers')
    .then(response => response.json())
    .then(data => {
        if (data.success && data.triggered_alerts.length > 0) {
            data.triggered_alerts.forEach(alert => {
                console.log(`Alert triggered: ${alert.name}`);
            });
        }
    });
```

3. **Viewing Trigger History**

```
// Get stored alerts with trigger info
fetch('/alerts/stored')
    .then(response => response.json())
    .then(data => {
        data.alerts.forEach(alert => {
            if (alert.alert_count > 0) {
                console.log(`${alert.name}: Triggered
${alert.alert_count} times`);
            }
        });
    });
```

Testing

Test Script

Run the comprehensive test:

```
python test_alert_triggers.py
```

This tests:

- Alert trigger API endpoint
- Stored alerts with trigger information
- Test alert creation and triggering
- Continuous monitoring capability

Manual Testing

1. **Create Test Alerts**:
    - Set targets close to current prices
    - Use "Above" or "Below" conditions
    - Set realistic target values
2. **Monitor Triggers**:
    - Watch the prices page
    - Look for notification popups
    - Check alert status updates
3. **Verify Database**:
    - Check `alert_count` increments
    - Verify `last_triggered_at` timestamps
    - Confirm `last_triggered_price` values

Performance Considerations

1. **Efficient Monitoring**

- Only checks enabled alerts
- Uses cached price data when possible
- Minimal database queries

2. **Rate Limiting**

- Checks every 1 second (not more frequent)
- Prevents excessive API calls
- Balances responsiveness with performance

3. **Error Handling**

- Graceful failure handling
- Continues monitoring on errors
- Logs issues for debugging

Troubleshooting

Common Issues

1. **No Triggers Detected**
    - Check if alerts are enabled
    - Verify target prices are realistic
    - Ensure price data is updating
2. **Notifications Not Showing**
    - Check browser console for errors
    - Verify JavaScript is enabled
    - Check if notifications are blocked
3. **Database Not Updating**
    - Check database permissions
    - Verify database file exists
    - Check for SQL errors in logs

Debug Steps

1. **Check Alert Status**:

```
curl http://localhost:5001/alerts/stored
```

2. **Test Trigger API**:

```
curl http://localhost:5001/alerts/check-triggers
```

3. **Check Current Prices**:

```
curl http://localhost:5001/stocks/fetch-price
```

Best Practices

1. **Alert Creation**

- Set realistic target prices
- Use appropriate operators
- Give descriptive names

2. **Monitoring**

- Don't create too many alerts
- Use percentage-based alerts for flexibility
- Regularly review and clean up old alerts

3. **Performance**

- Monitor system performance
- Check database size
- Clean up old trigger data if needed

Future Enhancements

Potential Features

- **Email Notifications** - Send alerts via email
- **SMS Alerts** - Text message notifications
- **Sound Alerts** - Audio notifications
- **Alert Groups** - Organize alerts by category
- **Trigger History** - Detailed trigger logs
- **Custom Notifications** - User-defined notification styles

The alert trigger monitoring system provides real-time price monitoring with immediate notifications when alert conditions are met, giving users instant awareness of important price movements.