Telegram Error Monitoring Setup Guide

📋 Daftar Isi

🎯 Overview

Sistem monitoring error dengan Telegram bot memberikan notifikasi real-time untuk:

- ✅ Critical errors dan exceptions
- ✅ Database errors
- ✅ API errors
- ✅ Security alerts
- ✅ Celery task failures
- ✅ WebSocket errors
- ✅ HTTP 5xx errors
- ✅ Suspicious request patterns

🤖 Setup Telegram Bot

Langkah 1: Buat Bot Telegram

1. Buka Telegram dan cari `@BotFather`
2. Kirim perintah `/newbot`
3. Ikuti instruksi untuk memberikan nama dan username bot
4. BotFather akan memberikan **Bot Token** (simpan untuk konfigurasi)

Contoh Bot Token:

```
123456789:ABCdefGHIjklMNOpqrsTUVwxyz
```

Langkah 2: Dapatkan Chat ID

Opsi 1: Menggunakan Bot @userinfobot

1. Cari `@userinfobot` di Telegram
2. Kirim pesan apa saja
3. Bot akan membalas dengan informasi termasuk Chat ID

Opsi 2: Menggunakan API Telegram

1. Kirim pesan ke bot Anda yang baru dibuat
2. Buka browser dan akses:

```
https://api.telegram.org/bot<YOUR_BOT_TOKEN>/getUpdates
```

3. Cari field `"chat":{"id":123456789}` - angka tersebut adalah Chat ID Anda

Opsi 3: Untuk Group Chat

1. Tambahkan bot ke group
2. Kirim pesan di group
3. Akses URL yang sama seperti Opsi 2
4. Cari Chat ID (untuk group, biasanya dimulai dengan tanda minus, contoh: `-987654321`)

⚙️ Konfigurasi Aplikasi

1. Update File `.env`

Buka file `.env` di root project dan tambahkan:

```
# Telegram Error Monitoring Configuration
TELEGRAM_ERROR_LOGGING_ENABLED=True
TELEGRAM_BOT_TOKEN=123456789:ABCdefGHIjklMNOpqrsTUVwxyz
TELEGRAM_CHAT_ID=987654321
ENVIRONMENT=production
```

Keterangan:

- `TELEGRAM_ERROR_LOGGING_ENABLED`: Set `True` untuk mengaktifkan, `False` untuk menonaktifkan
- `TELEGRAM_BOT_TOKEN`: Token yang didapat dari BotFather
- `TELEGRAM_CHAT_ID`: Chat ID Anda atau Group Chat ID
- `ENVIRONMENT`: Environment aplikasi (development/staging/production)

2. Install Dependencies

```
pip install -r requirements.txt
```

3. Restart Aplikasi

```
# Development
python manage.py runserver
```

```
# Production (Gunicorn)
gunicorn face_app.wsgi:application --bind 0.0.0.0:8000

# Production (uWSGI)
uwsgi --http :8000 --module face_app.wsgi

# Docker
docker-compose restart
```

🔍 Fitur Monitoring

1. Automatic Error Monitoring

Middleware secara otomatis mendeteksi dan melaporkan:

Exception Monitoring

```
# Otomatis menangkap semua unhandled exceptions
# Tidak perlu kode tambahan
```

Response Monitoring

```
# Otomatis melaporkan HTTP 5xx errors
# Status code 500, 502, 503, 504, dll.
```

Security Monitoring

```
# Otomatis mendeteksi:
# - SQL injection attempts
# - XSS attempts
# - Path traversal attempts
# - Suspicious patterns
```

2. Manual Error Logging

Anda bisa log error secara manual di kode:

```
from core.telegram_logger import telegram_logger, log_to_telegram

# Method 1: Using telegram_logger directly
try:
```

```
    # Your code here
    risky_operation()
except Exception as e:
    telegram_logger.log_critical_error(
        message="Failed to process user enrollment",
        exception=e,
        additional_context={
            'user_id': user.id,
            'operation': 'enrollment'
        }
    )

# Method 2: Using convenience function
try:
    # Your code here
    process_face_recognition()
except Exception as e:
    log_to_telegram(
        error_type="🔴 FACE RECOGNITION ERROR",
        message="Face recognition failed",
        exception=e,
        request=request,  # Optional: pass Django request object
        user=request.user,  # Optional: pass user object
        face_id=face.id,
        confidence_score=0.45
    )
```

3. Specific Error Types

```
from core.telegram_logger import telegram_logger

# Critical Error
telegram_logger.log_critical_error(
    message="System critical failure",
    exception=exception,
    additional_context={'system': 'database'}
)

# Database Error
telegram_logger.log_database_error(
    message="Database connection failed",
    exception=exception
)

# API Error
telegram_logger.log_api_error(
    message="External API call failed",
    exception=exception,
    additional_context={'api': 'face_recognition'}
)
```

```
# Security Alert
telegram_logger.log_security_alert(
    message="Unauthorized access attempt",
    exception=exception,
    request_data={
        'ip': '192.168.1.100',
        'path': '/admin/',
        'method': 'POST'
    }
)

# Celery Task Error
telegram_logger.log_celery_error(
    message="Background task failed",
    exception=exception,
    task_name='process_face_embeddings'
)

# WebSocket Error
telegram_logger.log_websocket_error(
    message="WebSocket connection error",
    exception=exception
)
```

🧪 Testing

Test Manual

Buat file test script `test_telegram.py`:

```
from core.telegram_logger import telegram_logger

# Test 1: Simple message
success = telegram_logger.log_critical_error(
    message="Test error notification from Face Recognition System",
    additional_context={
        'test': True,
        'timestamp': '2026-01-02 10:30:00'
    }
)
print(f"Message sent: {success}")

# Test 2: With exception
try:
    raise ValueError("This is a test exception")
except Exception as e:
    telegram_logger.log_exception(
        message="Testing exception logging",
        exception=e,
```

```
        additional_context={'test_type': 'exception'}
    )
```

Jalankan:

```
python manage.py shell < test_telegram.py
```

Test dengan Django Shell

```
python manage.py shell
```

```
from core.telegram_logger import telegram_logger

# Test notification
telegram_logger.log_critical_error(
    message="Testing Telegram notification",
    additional_context={'source': 'django_shell'}
)
```

Test dengan View

Tambahkan test endpoint (hanya untuk development):

```
# In core/views.py
from rest_framework.decorators import api_view, permission_classes
from rest_framework.permissions import AllowAny
from rest_framework.response import Response
from core.telegram_logger import telegram_logger

@api_view(['GET'])
@permission_classes([AllowAny])  # Remove in production
def test_telegram(request):
    """Test endpoint for Telegram notifications"""
    success = telegram_logger.log_critical_error(
        message="Test notification from API endpoint",
        request_data={
            'method': request.method,
            'path': request.path,
            'ip': request.META.get('REMOTE_ADDR')
        }
    )
    return Response({'success': success})
```

Access: `http://localhost:8000/api/test-telegram/`

🎨 Format Notifikasi

Notifikasi Telegram akan menampilkan:

```
🚨 CRITICAL ERROR
Environment: `production`
Time: `2026-01-02 14:30:45 UTC`

Message:
```

Database connection timeout

```

Exception Type: `OperationalError`
Exception Details:
```

could not connect to server: Connection refused

```

Traceback:
```

File "/app/core/database.py", line 45, in connect
conn = psycopg2.connect(...)

```
Request Info:
• Method: `POST`
• Path: `/api/auth/face-recognition/`
• IP: `192.168.1.100`
• User Agent: `Mozilla/5.0...`

User Info:
• ID: `12345`
• Username: `john_doe`
• Email: `john@example.com`

Additional Context:
• operation: `face_verification`
• attempt: `3`
```

🚨 Troubleshooting

Bot tidak mengirim notifikasi

1. Cek konfigurasi:

```
# Django shell
from core.telegram_logger import telegram_logger

print(f"Enabled: {telegram_logger.enabled}")
print(f"Bot Token: {telegram_logger.bot_token[:10]}...")
print(f"Chat ID: {telegram_logger.chat_id}")
print(f"Configured: {telegram_logger.is_configured()}")
```

2. Test koneksi:

```
# Django shell
import requests

bot_token = "YOUR_BOT_TOKEN"
url = f"https://api.telegram.org/bot{bot_token}/getMe"
response = requests.get(url)
print(response.json())
```

3. Cek log aplikasi:

```
# Check Django logs
tail -f logs/django.log

# Check for Telegram-related errors
grep -i "telegram" logs/django.log
```

Error "Unauthorized"

- Bot token salah atau expired
- Regenerate bot token dari @BotFather dengan `/token`

Error "Chat not found"

- Chat ID salah
- Bot belum pernah menerima pesan dari chat tersebut
- Untuk group: pastikan bot adalah member group

Notifikasi tidak muncul di Group

1. Pastikan bot sudah ditambahkan ke group
2. Kirim pesan di group terlebih dahulu

3. Gunakan Chat ID yang benar (biasanya dimulai dengan `-`)

Rate Limiting

Telegram API memiliki rate limit:

- Maximum 30 messages per second
- Maximum 20 messages per minute per chat

Jika terlalu banyak error, pertimbangkan:

```
# In settings.py
TELEGRAM_ERROR_THROTTLE = True
TELEGRAM_ERROR_THROTTLE_RATE = '10/minute'  # Max 10 errors per minute
```

🔐 Security Best Practices

1. **Jangan commit Bot Token ke repository**

   ```
   # Add to .gitignore
   .env
   .env.local
   .env.production
   ```

2. **Gunakan environment variables yang berbeda per environment**

   ```
   # Development
   TELEGRAM_CHAT_ID=123456789

   # Production
   TELEGRAM_CHAT_ID=-987654321  # Group chat
   ```

3. **Batasi informasi sensitif dalam notifikasi**

   - Jangan log password
   - Jangan log API keys/tokens
   - Sanitize user input

4. **Disable di development jika tidak perlu**

   ```
   TELEGRAM_ERROR_LOGGING_ENABLED=False
   ```

📊 Monitoring Dashboard

Untuk monitoring yang lebih advanced, pertimbangkan integrasi dengan:

- Sentry
- Datadog
- New Relic
- Prometheus + Grafana

Telegram bot sebaiknya digunakan untuk:

- Critical alerts
- Security events
- Real-time notifications
- Supplement to primary monitoring

🔄 Update dan Maintenance

Update Bot Token

1. Revoke old token via @BotFather: `/revoke`
2. Generate new token: `/token`
3. Update `.env` file
4. Restart aplikasi

Menambah Chat ID Baru

Untuk mengirim ke multiple chats, modify `telegram_logger.py`:

```
class TelegramLogger:
    def __init__(self):
        # Support multiple chat IDs
        chat_ids = getattr(settings, 'TELEGRAM_CHAT_ID', '')
        self.chat_ids = [id.strip() for id in chat_ids.split(',')]

    def send_message(self, message: str):
        for chat_id in self.chat_ids:
            # Send to each chat
            ...
```

Then in `.env`:

```
TELEGRAM_CHAT_ID=123456789,-987654321,456789123
```

📞 Support

Jika ada masalah:

1. Check logs: `logs/django.log`

2. Test bot manually via @BotFather
3. Verify network connectivity
4. Check Telegram API status: https://t.me/BotNews

📝 Changelog

Version 1.0.0 (2026-01-02)

- ✅ Initial release
- ✅ Error monitoring middleware
- ✅ Multiple error type support
- ✅ Security alert detection
- ✅ Automatic exception handling
- ✅ Manual logging support