Content Research 94% Hang Fix

Datum: 2 november 2025
Status: ✅ Opgelost en gedeployed

Probleem

De Content Research tool hing vast op 94% en gaf een “network error”. Keywords werden wel gevonden (backend werkte), maar de frontend kreeg de resultaten niet binnen en de statusbar werd niet afgewerkt.

Symptomen:

- ✅ Backend genereerde wel content ideeën
- ❌ Frontend hing op 94%
- ❌ “network error” verscheen in de UI
- ❌ Resultaten werden niet getoond

Root Cause

De API probeerde **alle content ideeën** (50+ items met veel data per item) in één Server-Sent Event (SSE) te sturen. Dit veroorzaakte:

1. **SSE Payload Limit**: SSE events hebben een maximale grootte, en onze data was te groot
2. **Network Buffer Overflow**: De browser kon de grote SSE event niet correct verwerken
3. **Stream Interruption**: De verbinding werd vroegtijdig verbroken

Oplossing

Backend Changes (`/api/client/content-research/route.ts`)

VOOR:

```
// Stuurde alle data via SSE (TE GROOT!)
sendData({
  type: 'complete',
  success: true,
  plan: contentPlan,              // ❌ Grote object
  articleIdeas: articleIdeasData, // ❌ Array van 50+ items
  message: '...',
});
```

NA:

```
// Stuurt alleen een completion signal (KLEIN!)
sendData({
  type: 'complete',
  success: true,
  totalIdeas: contentPlan.summary.totalIdeas, // ✅ Alleen getal
  message: '...',
  // ✅ Geen grote data - frontend haalt dit uit database
});
```

Frontend Changes (`/client-portal/content-research/page.tsx`)

VOOR:

```
// Probeerde data uit SSE response te gebruiken (ONBETROUWBAAR!)
setArticleIdeas(data.articleIdeas || []);
```

NA:

```
// Haalt altijd data uit database (BETROUWBAAR!)
if (selectedProject) {
  setTimeout(() => {
    loadContentPlan(selectedProject.id); // ✅ Vanuit database
    alert(`✅ ${data.message}`);
  }, 1000);
}
```

Verbeterde Error Handling

```
} catch (error: any) {
  if (error.message?.includes('network')) {
    alert('Network fout. De research is mogelijk wel voltooid. Ververs de pagina.');
    // ✅ Probeer automatisch te herladen
    if (selectedProject) {
      setTimeout(() => {
        loadContentPlan(selectedProject.id);
      }, 2000);
    }
  }
}
```

Benefits

Betrouwbaarheid

- ✅ **Geen SSE payload limits meer**: Alleen kleine progress updates via SSE
- ✅ **Database als source of truth**: Data wordt altijd vanuit database geladen
- ✅ **Betere error recovery**: Automatische reload bij network problemen

Performance

- ✅ **Snellere completion**: Geen grote data payloads meer via network
- ✅ **Minder memory gebruik**: Browser hoeft geen grote SSE events te bufferen

User Experience

- ✅ **Duidelijke feedback**: Gebruiker ziet wat er gebeurt
- ✅ **Automatische recovery**: Bij problemen probeert systeem automatisch te herladen
- ✅ **Betere foutmeldingen**: Gebruiker weet precies wat er mis ging

Technische Details

SSE Message Flow

VOOR (FOUT):

```
1. Frontend: POST /api/client/content-research
2. Backend: [SSE] 0% - Start...
3. Backend: [SSE] 20% - Website analyseren...
4. Backend: [SSE] 80% - Content genereren...
5. Backend: [SSE] 94% - Laatste checks...
6. Backend: [SSE] 100% + ALLE DATA (50+ ideeën) ❌ TE GROOT!
   └─> Network Error! Stream interrupted!
```

NA (CORRECT):

```
1. Frontend: POST /api/client/content-research
2. Backend: [SSE] 0% - Start...
3. Backend: [SSE] 20% - Website analyseren...
4. Backend: [SSE] 80% - Content genereren...
5. Backend: [SSE] 94% - Laatste checks...
6. Backend: [Database] Content ideeën opslaan ✅
7. Backend: [SSE] 100% + Completion signal (KLEIN!) ✅
8. Frontend: Reload data from database ✅
```

Database Flow

```
// 1. Backend slaat data op in database
await Promise.all(
  articleIdeasData.map(ideaData =>
    prisma.articleIdea.upsert({
      where: { clientId_slug: { clientId, slug } },
      update: { /* update bestaande */ },
      create: ideaData,
    })
  )
);

// 2. Backend stuurt alleen completion signal
sendData({
  type: 'complete',
  totalIdeas: contentPlan.summary.totalIdeas,
});

// 3. Frontend haalt data uit database
setTimeout(() => {
  loadContentPlan(selectedProject.id);
}, 1000);
```

Testing

Scenario 1: Normal Flow

```
✅ Start research
✅ Progress 0% → 100%
✅ Data wordt opgeslagen in database
✅ Frontend laadt data uit database
✅ Resultaten worden getoond
```

Scenario 2: Network Error

```
✅ Start research
✅ Progress 0% → 94%
❌ Network error tijdens SSE
✅ Automatische reload poging
✅ Data uit database wordt geladen (als opgeslagen)
✅ Gebruiker ziet resultaten
```

Scenario 3: Timeout

```
✅ Start research
⏰ Request timeout na 10 minuten
✅ Automatische reload poging
✅ Data uit database wordt geladen (als opgeslagen)
```

Code References

Backend

- **File**: `/nextjs_space/app/api/client/content-research/route.ts`
- **Lines**: 271-294 (SSE completion logic)
- **Key Change**: Removed large data from SSE, added completion signal only

Frontend

- **File**: `/nextjs_space/app/client-portal/content-research/page.tsx`
- **Lines**: 316-340 (SSE completion handling)
- **Lines**: 370-401 (Error handling)
- **Key Change**: Always reload from database, added auto-recovery

Deployment

```bash
# Build
cd /home/ubuntu/writgo_planning_app/nextjs_space
yarn build

# Deploy
build_and_save_nextjs_project_checkpoint
```

Live URL: https://writgoai.nl/client-portal/content-research

Monitoring

Monitor deze metrics:
- ✅ **Completion rate**: % van research requests die succesvol afronden
- ✅ **Error rate**: Aantal network errors / totaal requests
- ✅ **Average completion time**: Tijd tot resultaten zichtbaar zijn
- ✅ **Database writes**: Alle ideeën correct opgeslagen?

Conclusie

Het probleem is volledig opgelost door:
1. **SSE alleen voor progress**: Kleine updates, geen grote data
2. **Database als source of truth**: Betrouwbare data opslag en retrieval
3. **Betere error handling**: Automatische recovery bij problemen

De tool is nu veel betrouwbaarder en kan grote hoeveelheden content ideeën (50+) zonder problemen verwerken.