



WritgoAI Chatbot - Grondige Analyse & Verbeterplan



Huidige Status Analyse



WAT GOED WERKT

1. Sterke Backend Architectuur

- ✓ Native AIML tool calling met 12+ tools
- ✓ Multi-model intelligentie (GPT-4o, Gemini 2.5, Claude, etc.)
- ✓ Streaming responses met word-by-word display
- ✓ Heartbeat systeem voor lange requests (voorkomt timeouts)
- ✓ Credit systeem en rate limiting
- ✓ Project context integratie

2. Goede Features

- ✓ File upload support (images, documents)
- ✓ Web search tool
- ✓ Image generation
- ✓ Video generation
- ✓ Blog canvas (zoals Gemini Artifacts)
- ✓ Status terminal met real-time logs
- ✓ WordPress integratie
- ✓ Social media integratie

3. Goede UX Elementen

- ✓ Dark theme design
- ✓ Markdown rendering met syntax highlighting
- ✓ Auto-scroll tijdens generatie
- ✓ File upload interface
- ✓ Progress indicators



KRITIEKE VERBETERPUNTEN (om niveau ChatGPT/ Gemini te bereiken)

1. ✗ Conversatie Context & Geheugen (URGENT)

Probleem:

```
// Huidige implementatie: slechts 10 messages context
conversationHistory: messages.slice(-10).map(m => ({
  role: m.role,
  content: m.content
}))
```

ChatGPT/Gemini niveau:

- 📝 **Volledige conversatie geschiedenis** - niet limiteren tot 10 berichten
- 🧠 **Persistent geheugen** - onthoudt voorkeuren, context over sessies heen
- 🔄 **Intelligent context compressie** - samenvat oude berichten, behoudt belangrijke info
- 📊 **Token management** - slim prioriteren welke berichten mee te sturen

Oplossing:

```
// NIEUW: Intelligente context management
async function prepareConversationContext(
  messages: Message[],
  maxTokens: number = 100000
): Promise<Message[]> {
  // 1. Altijd laatste 20 berichten volledig meesturen
  const recentMessages = messages.slice(-20);

  // 2. Oudere berichten samenvatten als ze te lang zijn
  const olderMessages = messages.slice(0, -20);
  const summarized = await summarizeOldMessages(olderMessages);

  // 3. Belangrijke system prompts & user voorkeuren altijd meesturen
  const systemContext = await getSystemContext(clientId);

  return [
    ...systemContext,
    ...summarized,
    ...recentMessages
  ];
}
```

2. ❌ Response Kwaliteit & Natuurlijkheid

Probleem:

- System prompt is te instructief/robotachtig
- Te veel focus op planning in plaats van direct antwoorden
- Niet genoeg persoonlijkheid

ChatGPT niveau:

- 💬 **Natuurlijke conversatie flow** - praat als mens, niet als robot
- 🗣️ **Persoonlijkheid** - vriendelijk, behulpzaam, soms humor
- 🧠 **Intelligente vraagstelling** - alleen vragen om details als echt nodig
- ⚡ **Directe antwoorden** - geen onnodige planning/structuur

Huidige system prompt problemen:

```
// TE STRIKT / TE INSTRUCTIEF:
`Je bent WritgoAI - een snelle, slimme AI assistent die meedenkt.

=====
🧠 CREATIEVE VERZOEKEN - EERST VERDUIDELIJKEN
=====

Bij verzoeken om afbeeldingen, video's of creatieve content:
**ALTIJD EERST VRAGEN OM DETAILS!**`
```

Betere aanpak:

```
// NATUURLIJKE & FLEXIBELER:
`Je bent WritgoAI, een slimme en vriendelijke AI assistent.

Gedragsregels:
- Praat natuurlijk en menselijk, niet robotachtig
- Geef direct antwoorden waar mogelijk
- Vraag alleen om details als het écht nodig is voor kwaliteit
- Gebruik tools automatisch zonder dit aan te kondigen
- Wees creatief en behulpzaam

Voorbeeld natuurlijke gesprekken:
User: "Wat is SEO?"
Assistant: "SEO staat voor Search Engine Optimization. Kort gezegd help je Google (en andere zoekmachines) je website beter te begrijpen en hoger te ranken. De belangrijkste factoren zijn: goede content, technische snelheid, en relevante backlinks. Wil je tips voor jouw website?"

User: "Maak een afbeelding van een hond"
Assistant: "Leuk! Ik ga een afbeelding van een hond voor je maken. Welke stijl wil je? Realistisch, cartoon, of iets anders? En heb je voorkeur voor een bepaald ras?"
```

3. Tool Calling Transparantie

Probleem:

- Gebruiker ziet té veel "AI is bezig..." berichten
- Tool execution is te zichtbaar/storend
- Status terminal opent automatisch (verwarrend)

ChatGPT niveau:

- 🔍 **Subtiele tool gebruik** - "Researching..." kort boven input
- ⚡ **Geen aggressive popups** - status canvas optioneel
- 📊 **Tool details on-demand** - alleen tonen als user erom vraagt

Verbeterpunt:

```
// SUBTIELE STATUS UPDATES:
<div className="fixed bottom-20 left-1/2 -translate-x-1/2 z-10">
  {isGenerating && (
    <div className="bg-gray-800 px-4 py-2 rounded-full shadow-lg flex items-center gap-2">
      <Loader2 className="w-4 h-4 animate-spin" />
      <span className="text-sm">{currentStep}</span>
    </div>
  )}
</div>

// Status canvas NIET automatisch openen, maar kleine indicator:
{toolsUsed.length > 0 && (
  <button onClick={() => setShowStatusCanvas(true)}>
    <Badge variant="outline">
      {toolsUsed.length} tools gebruikt
    </Badge>
  </button>
)}
})
```

4. ❌ Model Selectie & Transparantie

Probleem:

- Model selectie logica is té complex en zichtbaar
- Te veel tier/reasoning info naar gebruiker
- Gebruiker wordt gestoord met model keuzes

ChatGPT niveau:

- 🎯 **Invisible model switching** - gebeurt automatisch, user ziet niks
- 💡 **Alleen tonen op verzoek** - "Welk model gebruik je?" → dan pas vertellen
- ⚡ **Simpele defaults** - 95% van queries op 1 model

Verbeterpunt:

```
// SIMPELER MODEL SELECTIE:
function selectModel(message: string): string {
  // Simpele heuristic: lang/complex → premium, rest → balanced
  const isComplex = message.length > 500 ||
    message.includes('schrijf') ||
    message.includes('maak');

  return isComplex ? 'gpt-4o' : 'gpt-4o-mini';
}

// Model info NIET automatisch tonen, alleen op verzoek:
{showDebugInfo && (
  <Badge variant="ghost" className="text-xs">
    Model: {message.model}
  </Badge>
)}
```

5. ❌ Conversatie Management

Probleem:

- Bij reload start automatisch nieuwe chat (verlies oude conversaties)
- Geen goede conversation history sidebar
- Moeilijk om oude chats terug te vinden

ChatGPT niveau:

- 📄 **Persistent conversations** - automatisch opslaan en terugladen
- 🔍 **Zoeken in geschiedenis** - find old conversations
- 📌 **Pin belangrijke chats**
- 📁 **Folders/tags** - organiseer conversations

Verbeterpunt:

```
// AUTO-SAVE CONVERSATIONS:
useEffect(() => {
  if (messages.length > 0) {
    const debounced = setTimeout(async () => {
      await saveConversation(currentConversation?.id, messages);
    }, 2000); // Save every 2 seconds

    return () => clearTimeout(debounced);
  }
}, [messages]);

// SIDEBAR MET HISTORY:
<div className="w-64 bg-gray-900 border-r border-gray-800">
  <Input placeholder="Zoek in geschiedenis..." />
  <ScrollArea className="h-full">
    {conversations.map(conv => (
      <ConversationItem
        key={conv.id}
        conversation={conv}
        active={conv.id === currentConversation?.id}
        onClick={() => loadConversation(conv.id)}
      />
    ))}
  </ScrollArea>
</div>
```

6. ❌ Response Speed & Performance

Probleem:

- Word-by-word streaming is té traag (vertraging per woord)
- Heartbeat system verstoort flow (te veel interrupts)
- Max 12 iterations is laag voor complexe taken

ChatGPT niveau:

- ⚡ **Snelle streaming** - 10-20 woorden per seconde
- 🎯 **Geen artificial delays** - stream zo snel als API data binnenkomt
- 🔄 **Hoger iteration limit** - 20-30 voor complexe redeneringen

Verbeterpunt:

```
// SNELLERE STREAMING (geen artificial delay):
for (let i = 0; i < words.length; i++) {
  const word = words[i];
  streamedContent += (i > 0 ? ' ' : '') + word;

  controller.enqueue(encoder.encode(createStreamUpdate('word', {
    word: word,
    content: streamedContent,
  })));

  // GEEN DELAY - stream direct door!
  // await new Promise(resolve => setTimeout(resolve, delay)); // VERWIJDEREN
}

// Of nog beter: stream character-by-character zoals API stuurt:
const reader = apiResponse.body.getReader();
while (true) {
  const { done, value } = await reader.read();
  if (done) break;

  // Direct doorsturen zonder processing
  controller.enqueue(value);
}
```

7. ❌ Error Handling & Recovery

Probleem:

- Cryptische error messages
- Geen graceful degradation
- Te veel crashes op edge cases

ChatGPT niveau:

- 🛡️ **Friendly error messages** - begrijpbaar voor gebruiker
- 🔄 **Automatic retry** - probeer opnieuw met fallback model
- 💡 **Suggesties bij errors** - help user om vraag anders te formuleren

Verbeterpunt:

```
// BETERE ERROR HANDLING:
async function handleChatError(error: any, retryCount: number = 0) {
  console.error('Chat error:', error);

  // 1. Try to recover with fallback model
  if (retryCount < 2) {
    toast.info('Proberen met ander model...');
    return await sendMessage(input, {
      model: getFallbackModel(),
      retryCount: retryCount + 1
    });
  }




  // 2. User-friendly error message
  const friendlyMessage = error.message.includes('timeout')
    ? 'Je vraag was te complex. Probeer het in kleinere stappen?'
    : error.message.includes('credits')
    ? 'Je hebt niet genoeg credits. Wil je er bijkopen?'
    : 'Er ging iets mis. Probeer het opnieuw';

  toast.error(friendlyMessage);





  // 3. Offer suggestions
  setSuggestions([
    'Probeer een kortere vraag',
    'Splits je vraag in delen',
    'Vraag om hulp'
  ]);
}
```

PRIORITEITEN voor Implementatie





FASE 1: Foundational Fixes (Week 1)

1.  **Fix conversatie context** - verhoog van 10 naar volledige geschiedenis
2.  **Verbeter system prompt** - natuurlijker, minder robotachtig
3.  **Remove artificial delays** - snellere streaming
4.  **Beter error handling** - user-friendly messages

FASE 2: UX Improvements (Week 2)

1.  **Subtiele status updates** - geen aggressive popups
2.  **Invisible model switching** - gebeurt automatisch
3.  **Persistent conversations** - auto-save & reload
4.  **Search in history** - find old chats

FASE 3: Advanced Features (Week 3)

1.  **Conversation folders** - organiseer chats
2.  **Follow-up questions** - AI stelt zelf vragen
3.  **Multi-turn reasoning** - complexe problemen oplossen
4.  **Voice input/output** - spraak interface



CONCRETE CHANGES NEEDED

1. app/api/client/chat/route.ts

```
// CHANGES:
- Line 264: conversationHistory verwijderen van 10-limit
+ Gebruik volledige geschiedenis met smart filtering
- Line 51-224: Simpelere model selectie logica
+ Invisible switching, minder tiers/reasoning
- Line 1011: Verwijder artificial delay in streaming
+ Stream direct zonder setTimeout
```

2. components/writgo-deep-agent.tsx

```
// CHANGES:
- Line 910-923: Auto-open terminal verwijderen
+ Subtiële status bar onder input
- Line 1000-1013: Word-by-word delay verwijderen
+ Character-by-character streaming zonder delay
- Line 735-770: Conversatie auto-save toevoegen
+ Save every 2 seconds automatically
```

3. System Prompt Rewrite

```
// BEFORE:
"Je bent WritgoAI - een snelle, slimme AI assistent..."
[300+ regels strikte instructies]

// AFTER:
"Je bent WritgoAI, een vriendelijke en slimme AI assistent.

Gedrag:
- Praat natuurlijk, niet robotachtig
- Geef directe antwoorden
- Gebruik tools automatisch
- Wees creatief en behulpzaam

[50 regels essentiële regels]"
```



SUCCESS METRICS

Na implementatie moet de chatbot:

- ⚡ **Response Time:** <2 seconden voor eerste woord
- 💬 **Conversation Quality:** 9/10 naturalness rating
- ↺ **Context Retention:** 100% van conversatie geschiedenis behouden
- 🛡️ **Error Rate:** <1% crashes, 100% graceful failures
- ★ **User Satisfaction:** Minimaal 8.5/10 rating

REFERENTIES

ChatGPT Best Practices:

- Invisible tool gebruik
- Natural conversation flow
- Smart context management
- Fast streaming responses

Gemini Best Practices:

- Artifacts/Canvas voor lange content
- Multi-turn reasoning
- Code execution
- File analysis

Conclusie:

De chatbot heeft een sterke basis, maar mist de natuurlijkheid en user experience van ChatGPT/Gemini. Met bovenstaande wijzigingen komen we op hetzelfde niveau!