**Already Covered Requirements**

1. **Core Interaction Logic**  
   User input → Display message → AI response (Thinking → Generating → Complete)
2. **Experience Optimization**  
   Streaming responses + Dynamic status + Auto-scrolling
3. **Content Rendering**  
   Supports Markdown formatting (better text display)
4. **Error Handling**  
   Friendly error prompts
5. **Streaming Processing**  
   Frontend-backend streaming integration for smooth responses
6. **Simplified Backend API**  
   Direct POST to GPT streaming API
7. **Basic Auth Framework**  
   JWT authentication (placeholder)
8. **Lightweight Frontend**  
   Vue3 + Composition API + TailwindCSS structure
9. **Thinking/Generating Indicators**  
   Human-like AI feedback with clear status updates

✅ **Integrated [Frontend + Backend Core Code] Example**  
(Ready for direct use or further development)

Frontend Code (Vue3 + TailwindCSS)

<template>

<div class="h-screen flex flex-col p-4 bg-gray-100">

<!-- Chat content -->

<div class="flex-1 overflow-y-auto space-y-4" ref="chatBox">

<div v-for="(msg, index) in messages" :key="index" class="flex flex-col">

<div :class="msg.role === 'user' ? 'text-right' : 'text-left'">

<div class="inline-block px-4 py-2 rounded"

:class="msg.role === 'user' ? 'bg-blue-500 text-white' : 'bg-white text-black border'">

<div v-if="msg.status === 'thinking'">💡 AI is thinking...</div>

<div v-else-if="msg.status === 'generating'">

<div class="flex space-x-1">

<div class="w-2 h-2 bg-blue-400 rounded-full animate-bounce"></div>

<div class="w-2 h-2 bg-blue-400 rounded-full animate-bounce delay-150"></div>

<div class="w-2 h-2 bg-blue-400 rounded-full animate-bounce delay-300"></div>

</div>

</div>

<div v-html="markedContent(msg.content)"></div>

</div>

</div>

</div>

</div>

<!-- Input box -->

<div class="mt-4 flex">

<textarea v-model="input" rows="2"

class="flex-1 border rounded p-2 focus:outline-none"

placeholder="Type your question, press Enter to send"

@keydown.enter.exact.prevent="sendMessage" />

<button @click="sendMessage"

class="ml-2 px-4 py-2 bg-blue-600 text-white rounded hover:bg-blue-700">

Send

</button>

</div>

</div>

</template>

<script setup>

import { ref, nextTick } from 'vue';

import { marked } from 'marked';

const messages = ref([]);

const input = ref('');

const sendMessage = async () => {

if (!input.value.trim()) return;

// Add user message

const userMsg = { role: 'user', content: input.value, status: 'complete' };

messages.value.push(userMsg);

// AI placeholder

const aiMsg = { role: 'assistant', content: '', status: 'thinking' };

messages.value.push(aiMsg);

input.value = '';

try {

const response = await fetch('/api/chat', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

'Authorization': `Bearer ${localStorage.getItem('token')}`

},

body: JSON.stringify({ message: userMsg.content })

});

aiMsg.status = 'generating';

const reader = response.body.getReader();

const decoder = new TextDecoder();

while (true) {

const { done, value } = await reader.read();

if (done) break;

aiMsg.content += decoder.decode(value);

await nextTick(() => {

const el = document.querySelector('.flex-1');

el.scrollTop = el.scrollHeight;

});

}

aiMsg.status = 'complete';

} catch (e) {

aiMsg.content = '⚠️ AI service unavailable';

aiMsg.status = 'error';

}

};

const markedContent = (text) => marked.parse(text, { breaks: true });

</script>

Backend Code (Laravel API /chat with Streaming Support)

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Http;

Route::post('/chat', function (Request $request) {

$prompt = $request->input('message');

return response()->stream(function () use ($prompt) {

$client = new \GuzzleHttp\Client();

$response = $client->post('https://api.openai.com/v1/chat/completions', [

'headers' => [

'Authorization' => 'Bearer ' . env('OPENAI\_API\_KEY'),

'Content-Type' => 'application/json',

],

'json' => [

'model' => 'gpt-4',

'messages' => [['role' => 'user', 'content' => $prompt]],

'stream' => true

],

'stream' => true

]);

$body = $response->getBody();

foreach (\GuzzleHttp\Psr7\StreamWrapper::getResource($body) as $chunk) {

$data = substr($chunk, 6);

if ($data === "[DONE]") break;

$json = json\_decode($data, true);

$content = $json['choices'][0]['delta']['content'] ?? '';

echo $content;

ob\_flush();

flush();

}

}, 200, [

'Content-Type' => 'text/event-stream',

'Cache-Control' => 'no-cache',

'X-Accel-Buffering' => 'no'

]);

});

**Integrated Feature Checklist (Frontend + Backend)**

✅ **Additional Suggestions (Next Steps)**

1. **Frontend Extensions**
   * User login / Profile (Upload history / Search records)
   * Form interactions (e.g., property listings)
2. **Backend Extensions**
   * User authentication (JWT/Session)
   * Database integration (Property listings / Chat history)
   * Payment / Permission controls
   * Multi-language / Multi-model support

**Submission**

Send your GitHub repo/ZIP file to: 62 812 9657 7544  
Subject: **Laravel Test Submission - [Your Name]**