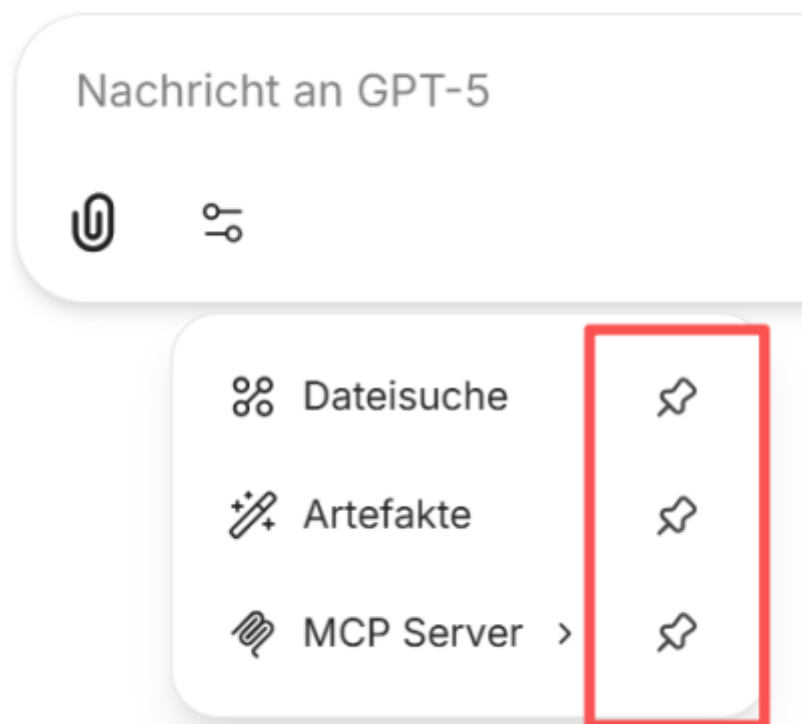


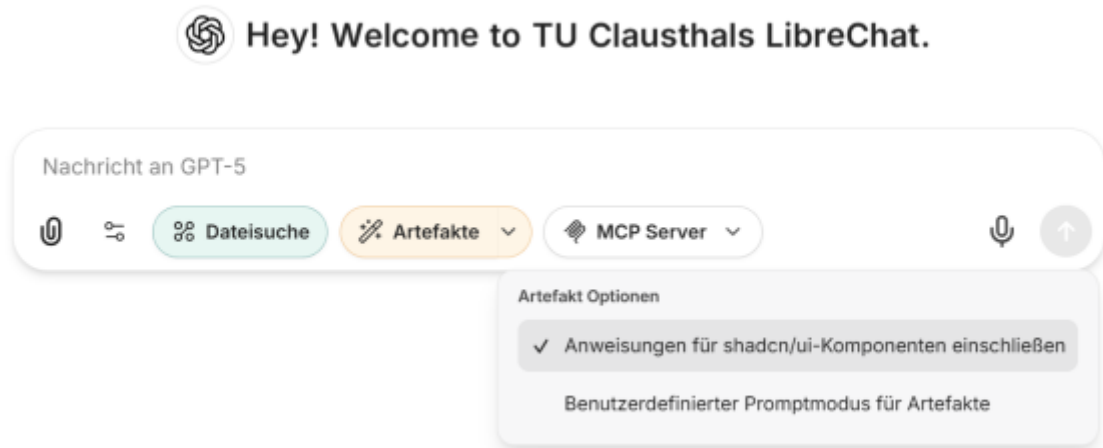
Empfohlener Ablauf zur Nutzung

Nutzung von Artifacts

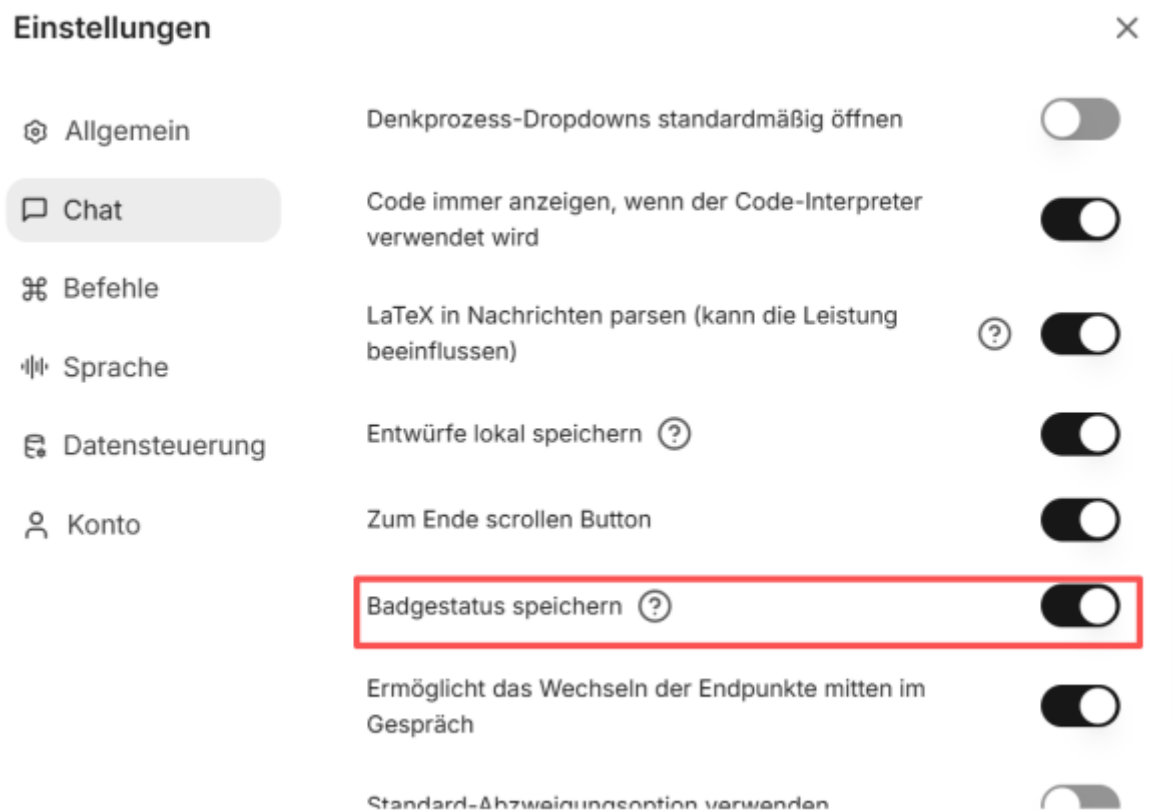
Verwenden Sie Artifacts, um die Anzeige und den Download von Dateien zu optimieren. Die gezeigten Screenshots illustrieren den empfohlenen Ablauf zur Nutzung von Dateien und Artifacts in LibreChat:



- Oben ist zu sehen, wie im Chatfenster verschiedene Upload- und Anzeigemethoden zur Auswahl stehen – insbesondere „Dateisuche“ und „Artefakte“. Diese Optionen lassen sich im Nachrichtenbereich von LibreChat auswählen und sind für die Weiterverarbeitung und Darstellung von Dateien gedacht.



- Im mittleren Ausschnitt wird unter den Einstellungen verdeutlicht, dass die Artifact-Funktion aktiviert ist und weitere Upload-Möglichkeiten bereitstehen. Die rote Markierung hilft Nutzern, die relevanten Schalter und Bereiche im Interface schnell zu finden.



- Durch das Speichern des Badgestatus werden die ausgewählten Einstellungen standardmäßig beibehalten, sodass die Auswahl nicht jedes Mal erneut getroffen werden muss.

The screenshot displays a web-based chat interface (LibreChat) with a GPT-5 model. The chat history shows a user prompt: "generate me a simple python script and show in a file". The model's response includes a detailed explanation of the script's functionality and the full Python code. The code defines functions for greeting, factorial calculation, listing files, and report generation. The interface also features a file explorer and a download button.

```
#!/usr/bin/env python3
"""
simple_script.py

A simple, reusable Python script that:
- Greets the user
- Computes the factorial of a number
- Lists .py files in a directory
- Optionally saves a short report to a file

Usage examples:
python simple_script.py --name Alice --number 5 --path .
python simple_script.py --name Bob --number 10 --path ./src --save report.txt
"""
import argparse
import logging
import os
from typing import List

def greet(name: str) -> str:
    """Return a greeting for the given name."""
    return f"Hello, {name}!"

def factorial(n: int) -> int:
    """Compute factorial iteratively to avoid recursion depth issues."""
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers.")
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

def list_python_files(path: str) -> List[str]:
    """Return a list of .py files in the given directory (non-recursive)."""
    try:
        entries = os.listdir(path)
    except OSError as e:
        logging.error("Failed to list directory '%s': %s", path, e)
        return []
    return [f for f in entries if f.endswith(".py") and os.path.isfile(os.path.join(path, f))]

def make_report(name: str, number: int, path: str) -> str:
    """Create a small report string describing operations performed."""
    files = list_python_files(path)
    fact = factorial(number)
    lines = [
        greet(name),
        f"Computed factorial({number}) = {fact}",
        f"Found {len(files)} Python file(s) in {path}.",
    ]
    lines.extend(f" - {fn}" for fn in files)
    return "\n".join(lines)

def parse_args():
    p = argparse.ArgumentParser(description="Simple Python utility script")
    p.add_argument("--name", "-n", default="World", help="Name to greet")
    p.add_argument("--number", "-k", type=int, default=5, help="Number to compute factorial for")
    p.add_argument("--path", "-p", default=".", help="Directory to list .py files from")
    p.add_argument("--save", "-s", help="Optional path to save the report")
    p.add_argument("--verbose", "-v", action="store_true", help="Enable verbose logging")
    return p.parse_args()

def main():
    args = parse_args()
    logging.basicConfig(level=logging.DEBUG if args.verbose else logging.INFO, format="%(levelname)s: %(message)s")
    try:
        report = make_report(args.name, args.number, args.path)
    except Exception as e:
        logging.error("An error occurred: %s", e)
```

- Der große Screenshot zeigt einen typischen Anwendungsfall. Die Dateien, die über diese Schaltfläche heruntergeladen werden, weisen derzeit Formatierungsprobleme auf, die jedoch in Zukunft behoben werden. Vorläufig werden Standarddateien weiterhin als „index.html“ gespeichert. Daher ist eine manuelle Umbenennung der Datei weiterhin erforderlich, bsp. als demo.py.

Nutzung von Text mit OCR

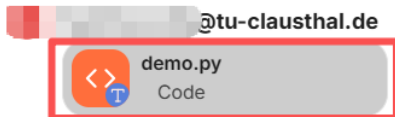


Diese Methode eignet sich daher besonders für kurzfristig wichtige Dateien, wie z.B. Textkorrekturen oder Übersetzungen, die nach ihrer Verwendung nicht erneut benötigt werden. Aufgrund der begrenzten Serverressourcen trägt das Hochladen von Dateien im Textformat für den kurzfristigen Gebrauch ebenfalls dazu bei, unseren Ressourcenverbrauch zu reduzieren, zusätzlich zur regelmäßigen Bereinigung nicht benötigter persönlicher Dateien in LibreChat.

Dateien, die im Textformat hochgeladen werden, werden im Klartextformat an das Sprachmodell

gesendet. Die spezifischen Unterschiede sind wie folgt:

- Ein Anwendungsfall:



Was ist der Zweck dieses Codes?



GPT-4.1

Der Zweck des bereitgestellten Python-Skripts (simple_script.py) ist es, eine einfache, vielseitige Befehlszeilen-Utility zur Verfügung zu stellen, die die folgenden Aufgaben erledigt:

1. Begrüßung des Benutzers:

Das Skript begrüßt einen Benutzer mit einem über die Kommandozeile übergebenen Namen (z.B. "Hello, Alice!").

2. Fakultätsberechnung:

Es berechnet die Fakultät einer angegebenen Zahl (z.B. $5! = 120$).

3. Auflisten von Python-Dateien:

Im angegebenen Verzeichnis (Ordner) listet das Skript alle dort vorhandenen Python-Dateien (*.py) auf (nicht rekursiv).

4. Erstellung eines Berichts:

Aus den Informationen (Begrüßung, Fakultät, Dateiliste) erstellt das Skript einen kurzen Bericht und gibt ihn auf der Konsole aus.

5. Optional: Speichern des Berichts in eine Datei:

Mit dem Parameter --save kann der Bericht in eine Textdatei geschrieben werden.

6. Optional: Ausführliche Protokollierung:

Mit --verbose werden zusätzlich ausführliche Protokollinformationen im Terminal ausgegeben.

Zusammengefasst:

Das Skript ermöglicht es, mit einfachen Kommandozeilenbefehlen statistische und explorative Aufgaben (Begrüßung, mathematische Berechnung, Dateisystem-Inspektion) schnell durchzuführen und die Ergebnisse optional in einer Datei zu speichern.

Typische Anwendungsbeispiele:

- Einen einfachen Bericht über Python-Dateien in einem Projektordner erstellen
- Die Fakultät einer beliebigen Zahl berechnen
- Skripte testen, Python-Utility-Funktionen demonstrieren

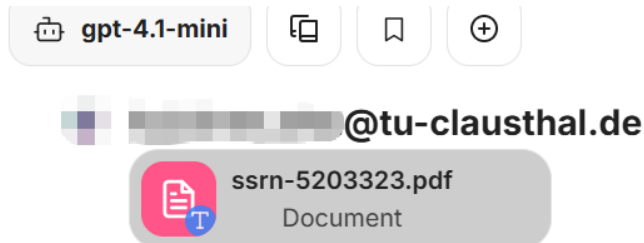
Wenn du möchtest, kann ich die Funktionsweise auch Zeile für Zeile erklären oder einzelne Bereiche detailliert aufschlüsseln!



- Beim Hochladen als reiner Text verwendet das Sprachmodell keine zusätzlichen Werkzeuge, wie

zum Beispiel das Tool „file search“.

- Hochgeladene Dateien sind in der rechten unteren Ecke mit einem „T“-Symbol markiert, das anzeigt, dass es sich um Textinhalt und nicht um eine Datei handelt.
- Der Textinhalt existiert nicht in Form einer „Datei“ und kann daher bei längerem Gebrauch nicht mehrfach wiederverwendet werden. Selbst für das Sprachmodell ist es schwierig, in großen Mengen unstrukturierter historischer Daten spezifische Informationen schnell und präzise zu finden.



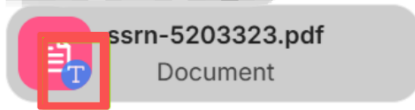
Zusammenfassung des obenstehenden Textes

GPT-4.1

Das Dokument beschreibt ein neues Framework namens Fourier Feature-Embedded Physics-Informed Neural Network-based Topology Optimization (FF-PINNTO) für die Topologieoptimierung von geometrisch nichtlinearen Strukturen. Dieses Framework nutzt physik-informierte neuronale Netzwerke (PINNs) mit Fourier-Feature-Einbettung und periodischen Aktivierungsfunktionen, um die Modellierung nichtlinearer partieller Differentialgleichungen zu verbessern.

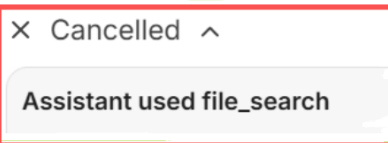


 i-clausthal.de



Zusammenfassung der oben genannten Dokumente

GPT-4.1



```
{
  "query": "Zusammenfassung Fourier feature embedded physics-informed neural network-based topology optimization nonlinear structures"
}
```

Result

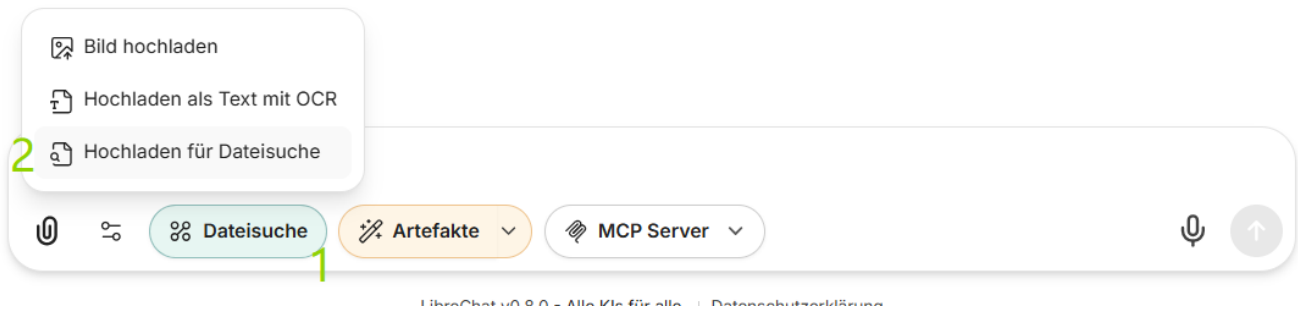
Error processing tool: Tool response format is "content_and_artifact" but the output was not a two-tuple.
Result: "No files to search. Instruct the user to add files for the search."

Ich habe momentan keine Dokumente geladen, um eine Zusammenfassung zu erstellen. Bitte laden Sie die entsprechenden Dokumente hoch, damit ich Ihnen eine Zusammenfassung liefern kann.

Nutzung von Dateisuche



Beim Hochladen von Dokumenten im Dateiformat werden diese von einem voreingestellten Modell (derzeit GPT-5-Nano) in mehrere Textabschnitte mit unterschiedlichen Schlüsselwörtern unterteilt. Dabei erfolgt eine klare Kennzeichnung, welcher Abschnitt zu welcher Datei gehört. Diese Art der Dateiuploads ermöglicht es den großen Sprachmodellen (LLMs), den exakten Inhalt jeder einzelnen Datei eindeutig zuzuordnen. Daher ist diese Methode besonders gut für langfristige Aufgaben geeignet.



Das Bild zeigt einen Ausschnitt aus der Dateiupload- und Funktionsleiste in LibreChat. Zunächst sieht man die Schaltfläche „Dateisuche“ (Nummer 1). Erst wenn man auf diese Schaltfläche klickt, öffnet sich das oben angezeigte Auswahlménü (Nummer 2), in dem verschiedene Optionen für den Datei-Upload erscheinen. Dazu gehören „Bild hochladen“, „Hochladen als Text mit OCR“ und „Hochladen für Dateisuche“.

- Ein Anwendungsfall:



GPT-4.1

Dieser Code ist ein einfaches, wiederverwendbares Python-Skript, das folgende Aufgaben erfüllt (siehe Datei demo.py):

1. Begrüßt den Benutzer mit einem personalisierten Gruß.
2. Berechnet die Fakultät (factorial) einer angegebenen Zahl.
3. Listet alle .py-Dateien in einem angegebenen Verzeichnis (nicht rekursiv) auf.
4. Kann optional einen kurzen Bericht über diese Operationen in eine Datei speichern.
5. Unterstützt optionale Parameter und eine Verbose-Option für detaillierte Logausgaben.

Das Skript kann mit verschiedenen Argumenten über die Kommandozeile ausgeführt werden, z.B.:

```
css
Code kopieren

python simple_script.py --name Alice --number 5 --path .
```


Zweck: Es dient als Beispiel für grundlegende Aufgaben wie Benutzerinteraktion, mathematische Berechnung, Dateioperationen und Berichtserstellung in Python.



- Das Dateisymbol für hochgeladene Dokumente im Datei-Format verfügt nicht über das „T“-Symbol in der unteren rechten Ecke. Dies ist auch im persönlichen Dateimanagement direkt erkennbar.



 u-clausthal.de

 ssrn-5203323.pdf
Document

Zusammenfassung der oben genannten Dokumente

GPT-4.1

✓ Ran file_search ^

Assistant used file_search

```
{
  "query": "Zusammenfassung"
}
```

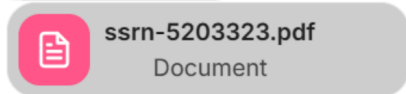
Result

```
File: ssrn-5203323.pdf
Anchor: \ue202turn0file0 (ssrn-5203323.pdf)
Relevance: 0.2288
Content: mization 63 (5) (2021) 2571-2588.962
[80] D. Jakubovitz, R. Giryes, M. R. Rodrigues, Generalization error in deep963
learning, in: Compressed sensing and its applications: third interna-964
tional MATHEOn conference 2017, Springer, 2019, pp. 153-193.965
[81] M. Nazem, M. H. Moavenian, Alternative remeshing techniques for large966
deformation analysis of geotechnical problems, Computers and Geotech-967
nics 138 (2021) 104344.968
40
This preprint research paper has not been peer reviewed. Electronic copy available at:
https://ssrn.com/abstract=5203323
Preprint not peer reviewed
```

Das Dokument ist ein wissenschaftliches Preprint-Papier, das sich mit der Entwicklung eines robusten Ansatzes für ein bestimmtes technisches oder ingenieurwissenschaftliches Problem beschäftigt. Es beinhaltet Algorithmen und Optimierungsmethoden, möglicherweise im Bereich der Topologieoptimierung oder numerischer Simulationen, unterstützt durch CAD-Modell-Rekonstruktionen. Das Papier umfasst Referenzen zu relevanter Literatur, darunter Arbeiten zu Tiefe Lernalgorithmen (Deep Learning), Finite-Elemente-Methoden und Topologieoptimierung in MATLAB. Die Simulationen im Dokument verwenden eine Finite-



 **tu-clausthal.de**




Der oben genannte **Inhalt**





GPT-4.1

Ich sehe, dass Sie "Der oben genannte Inhalt" erwähnt haben. Könnten Sie bitte genauer erläutern, **auf welchen Inhalt** Sie sich beziehen oder welche Informationen Sie genau benötigen? So kann ich Ihnen besser weiterhelfen.

 **Dateien anhängen**

Dateien filtern...

Name	↑↓	Datum	↑↓
 demo.py		6 Oct 2025	
 demo.py		6 Oct 2025	

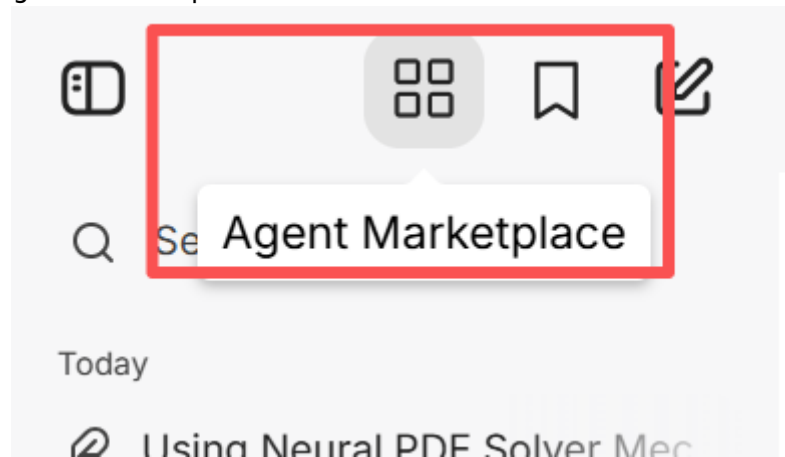
Nutzung von Agenten



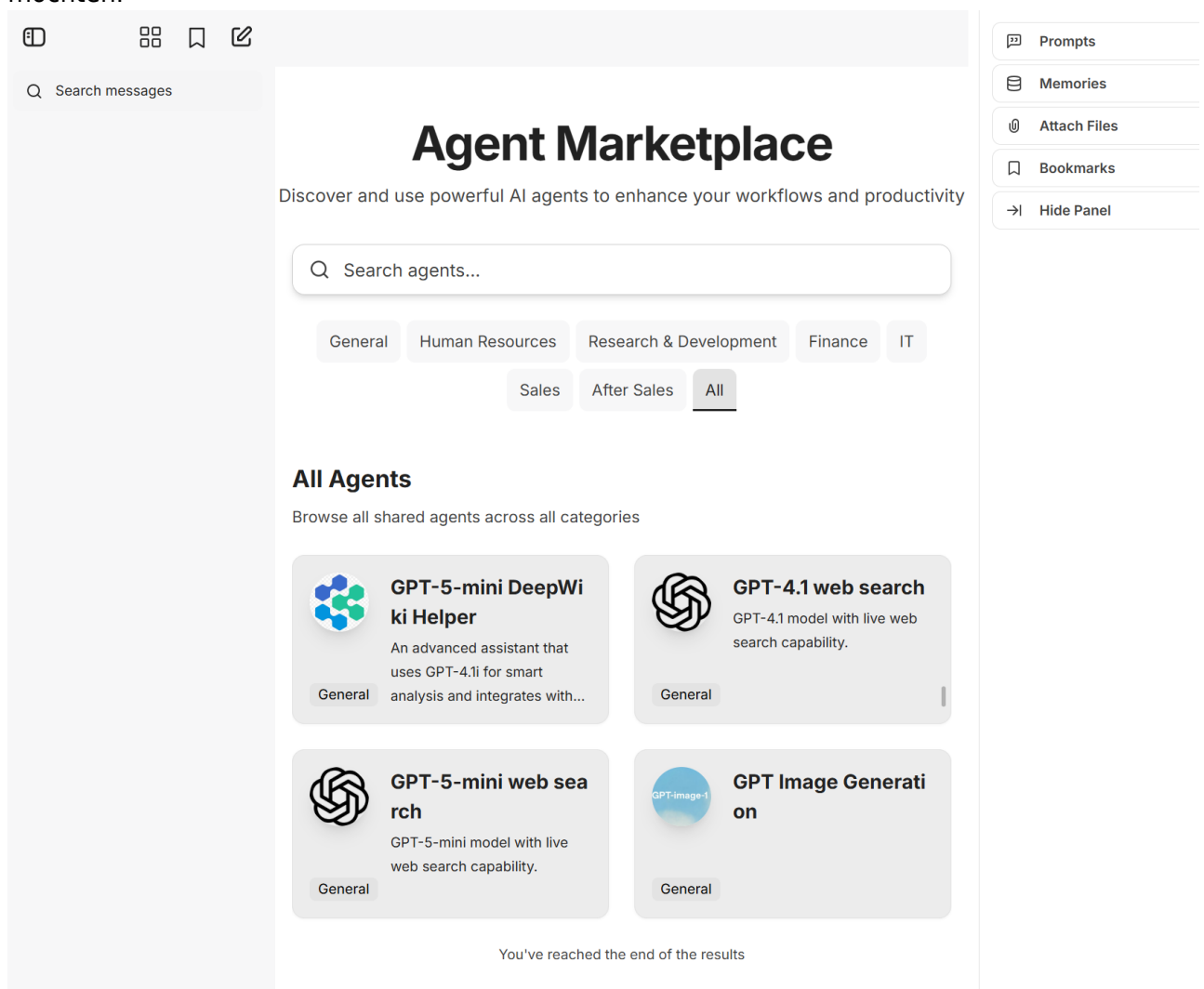
In der neuen LibreChat Version 0.8.0 werden die bereitgestellten Agenten-Möglichkeiten im Agenten-Marktplatz angeboten. Nur wenn Sie einen Agenten aus dem Marktplatz auswählen und verwenden, wird dieser Agent zu Ihrem persönlichen Agenten.

Agenten Anwendungsschritte für die Agenten:

1. Betreten Sie den Agenten-Marktplatz.



2. Wählen Sie einen Agenten aus, den Sie zuvor noch nicht verwendet haben, aber diesmal nutzen möchten.



3. Überprüfen Sie in der Endpunktliste „Meine Agenten“.


Agent Marketplace

Discover and use powerful AI agents to enhance your workflows and productivity


General Human Resources Research & Development Finance IT Sales After Sales All

All Agents


Browse all shared agents



General



General

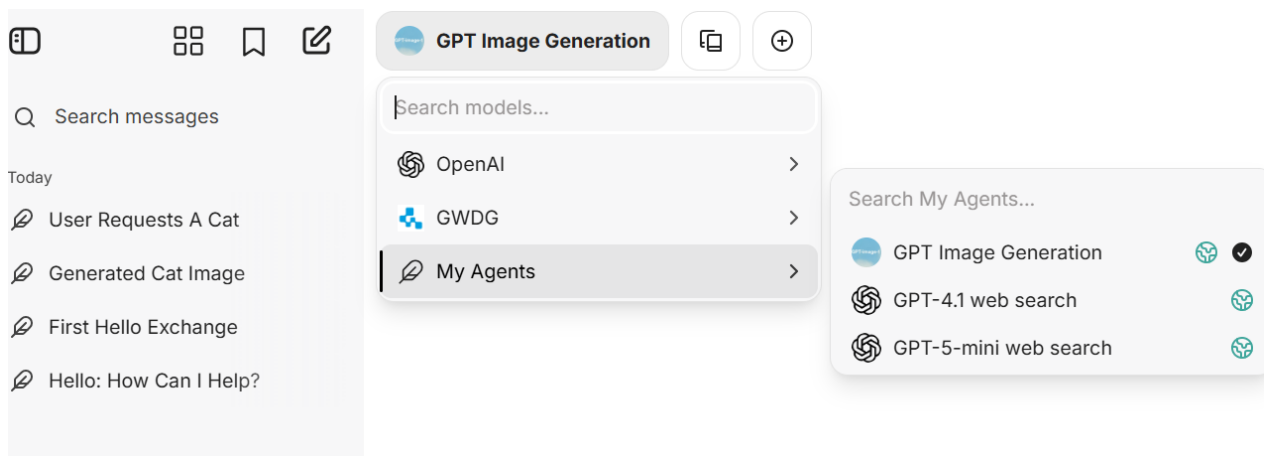


GPT-5-mini DeepWiki Helper

An advanced assistant that uses GPT-4.1i for smart analysis and integrates with the DeepWiki MCP server to fetch and summarize technical documentation, libraries, and code resources from deepwiki.com for highly relevant answers in development, engineering, or AI contexts.

[Start Chat](#)

You've reached the end of the results



Direkt-Link:

https://doku.tu-clausthal.de/doku.php?id=sonstige_dienste:ki-dienste:librechat:empfohlener_ablauf_zur_nutzung&rev=1759907273

Letzte Aktualisierung: **07:07 08. October 2025**

