OmniVault AI Transfer - Technical Implementation Plan

1. The Architecture Stack

Frontend: Chrome Extension (Manifest V3)
- Language: JavaScript (Vanilla), HTML, CSS
- Role: Scrapes text from ChatGPT, Automates pasting into Gemini.

Backend: Local Server
- Language: Python 3.10+
- Framework: FastAPI (Fast, modern, handles async)
- Role: Receives JSON data, cleans it, converts to Markdown.

2. Step-by-Step Implementation

Phase 1: Setup
- Create folders: /omnivault-extension and /omnivault-server
- Install libraries: pip install fastapi uvicorn beautifulsoup4

Phase 2: The Scraper (Extension)
- Detect Active Chat: Listen for URL changes (chatgpt.com/c/...). Save URL.
- Scrape DOM: Find <div class='text-message'>. Extract text to JSON.
- Send to Python: POST request to localhost:8000.

Phase 3: The Converter (Python)
- API Endpoint: /process-history
- Logic: Loop through JSON. Convert User/AI roles to Markdown (**You**: ...).

Phase 4: The Injector (Automation)
- Open Gemini: Extension opens chrome.tabs.create({url: 'gemini.google.com'}).
- Paste & Send: Inject script to find contenteditable div, insert text, click Send.

3. Backend Code (server.py)

Save this as server.py and run with: uvicorn server:app --reload

```
from fastapi import FastAPI
from pydantic import BaseModel
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

# Fix CORS errors (Mandatory for extensions)
```

OmniVault AI Transfer - Technical Implementation Plan

```
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_methods=["*"],
    allow_headers=["*"],
)

class ChatData(BaseModel):
    messages: list

@app.post("/process-history")
def process_chat(data: ChatData):
    formatted_text = "# Imported History\n\n"
    for msg in data.messages:
        role = msg.get('role', 'Unknown')
        content = msg.get('content', '')
        formatted_text += f"**{role.upper()}:** {content}\n\n---\n\n"

    return {"clean_text": formatted_text}
```

4. Extension Config (manifest.json)

Save this in your extension folder.

```
{
  "manifest_version": 3,
  "name": "OmniVault AI Transfer",
  "version": "1.0",
  "permissions": ["activeTab", "scripting", "storage"],
  "host_permissions": [
    "https://chatgpt.com/*",
    "https://gemini.google.com/*",
    "http://localhost:8000/*"
  ],
  "background": { "service_worker": "background.js" },
  "content_scripts": [
    { "matches": ["https://chatgpt.com/*"], "js": ["content_gpt.js"] },
    { "matches": ["https://gemini.google.com/*"], "js": ["content_gemini.js"] }
  ],
  "action": { "default_popup": "popup.html" }
}
```

5. Critical Gotchas

1. CORS Errors: Browsers block extensions from talking to localhost unless you configure CORS in FastAPI.
2. Class Names: ChatGPT changes CSS class names often. Use generic selectors or update your scraper

regularly.
3. Input Box: Gemini uses a rich-text editor, not a simple input. You may need to use document.execCommand or ClipboardEvent to paste text programmatically.