Point & Click Software Engineer Take-Home Assignment

I - Introduction & Context

About This Assignment

At Point & Click, we believe the best way to evaluate someone's ability is seeing them work in real conditions. Our process includes a take-home technical assignment that requires non-negligible time investment, but reflects our philosophy: **we prioritize evaluating actual job skills over theoretical off topic performance.**

This assignment focuses on building a simplified version of what we do at Point & Click - creating an AI Agent that lives in your browser to take care of menial tasks for you. It gives you space for deep thinking and technical work similar to what you would do with us.

II - The Challenge

Problem Statement

Your goal is to build a minimal version of Claude Computer Use Agent that can pilot a Chrome Extension.

Your deliverable should takes as input:

- natural language tasks such as “Go through my recent Gmail and find email lists or promotional emails that I haven’t opened in the last 3 months” or “Find the latest paper on Hugging Face Daily Paper about UI Agents”

output:

- Nothing, but the Chrome Extension should be manipulated to execute the task using Claude Computer Use Agent.

The focus of this assignment is not on the UX/UI but the ability to understand how Computer Use works, how Chrome Extensions work and how to combine both.

We suggest splitting the code in two parts:

- Chrome adapter bridge that can be piloted by a client
- Client (e.g. Python) that orchestrates the Computer Use logic and communicates with the Chrome adapter to pilot the browser

You are free to use whatever language you are most comfortable with as long as you respect the assignment.

Bonus:

- If you finish early, you can add a custom tool to the Chrome adapter and adapt the Agent accordingly to support:
    - download of files
    - upload of files
    - switch tab

Technical Requirements

- **Framework:** App built with your preferred stack but must be able to pilot a Chrome Extension
- **Input:** Natural language query
- **Expected behavior:** pilots the browser to perform the task
- **Execution model:** Client-only. Any LLM calls happen from the Extension / Client with a user-supplied API key. (Store locally; do not ship your key.)

Deliverable:

- Working application (code + deployment instructions)
- **3-5 minutes Loom video demonstrating functionality**
- A short **README** explaining setup, how the API key is used, and any limitations.

III - Evaluation Criteria

Your deliverable is judged

- first on its adherence to the provided instructions
- then, if functional, on the code quality, organization, and architecture

We're not just evaluating the final result but also your technical decision-making and your interactions with us to iterate and solve the problem with the most elegant and functional solution.

IV - Constraints & Guidelines

- **Time**: 72 hours from receipt to submission
- **Code Standards:**
    - ➔ You MUST be able to explain and justify every line of code
    - ➔ If using AI assistance, ensure full understanding; avoid code that feels auto-generated or unexplained.
    - ➔ Keep the codebase clean, readable, and documented (comments where it matters, concise README).

V - Submission

You will send your deliverable to the following email: contact@conception.dev

Good luck!