Interview Task

Senior Tools Software Engineer @ **Croud**

For the second stage interview for the **Senior Tools Software Engineer** role in the **Croud Engineering Team**, we'd like you to **prepare for the following task and take us through your response during the panel interview.**

You will have **1 week since receiving this assignment to solve the task**. Feel free to return it earlier if you feel like you have finished. Any inconvenience related to this timing, please contact paul.watkins@croud.com.

After you have delivered the solution to us, our engineering team will review it and invite you for a technical interview round where we will discuss it.

Subject to time and the natural flow of conversation we may elect to break this technical round into two stages:

1. Discussing the results of this challenge, walking through the design and decisions (1-1.5 hrs)
2. A practical code based and SDLC round, to evaluate competency in a pair programming approach with a view of determining how you would function day to day with key members of the team. (1-1.5 hrs)

Both sessions will be handled by Lead Engineers in the team, who are daily touch points for all projects.

Task Description

Croud account management teams manage social media accounts on behalf of clients, and proactively review the feedback and impact of any changes to those accounts.

We plan on building an internal tool to support those teams that analyses comments added to social media posts and videos on Instagram, YouTube and TikTok. The intention is to determine the sentiment of any comments retrieved for a list of post URLs to determine how positive the feedback is on a given post.

We want a high level solution design that outlines key challenges that we would face with undertaking this project. Some key considerations are:

- This is an opportunistic project. We have seen a way to make our team's lives easier and more productive. However, the ROI is initially uncertain, so effort should be spent wisely. We want to get a solution in front of users as soon as possible to validate value and provide an iterative and incremental roadmap for change. This should be a factor in your decisions. Equally, should this be adopted and used at increasing scales, what are ongoing considerations?
- Retrieving post information from social platforms manually is time consuming and subjective. We aim to assist with this by optimistically retrieving post and comment information programmatically, avoiding the manual effort in collating this source information.

  The intention is that a user can provide a list of platform URLs, which are then queried to provide a list of comments for each URL on the respective platform.

  We have no pre-conceived opinions on how this should be approached - external APIs, scraping or a combination of them, and you have a free remit to suggest approaches. The use of paid or third party service is acceptable, but the consequences of doing so (costs, reliability etc) should be identified and justified.

  This demonstrates your ability to research and evaluate options to solve an implementation detail.

- Similarly, evaluating comment data is also very time consuming. This may be complicated with variations in language, colloquialisms and the use of contextual use of emojis. Again we want to provide a consistent means of evaluating the sentiment of the each comment obtained in the previous step that is distilled down to two metrics:
  - A decimal value between -1 and +1 representing the positivity or negativity of the comment. -1 represents very negative, +1 very positive and a small decimal value represents relatively neutral.
  - A string value - NEGATIVE / |NEUTRAL / POSITIVE - to label the sentiment score.

  Note: Again, no fixed opinion on this. The use of natural language libraries is acceptable, as is the use of third party services or LLMs to perform this action. Again, justify your choices and provide any evidence you can to support your decision (example prompts and outputs for LLM usage, for example)

  Emojis are prevalent in social media comments. How do we evaluate these to determine context and sentiment? The same could be said about other factors like interpreting sarcasm.

- Validation of evaluated results is variable and subjective. What could be done to govern this, and provide measurability of this valuation? What kind of problem do you anticipate?
- Data isolation is not a concern with this project, though responsible storage is.
- The system output will be stored in a database and should be accessible via a REST API to a frontend that consumes said API. Results shall be provided as a CSV to download via the API.
- A lightweight UI will be built to display executions and results for a given user. Implementation choices about the UI are not required for this exercise.
- Authentication to the UI and API will use existing Croud OIDC and access is determined with the use of a valid JWT.
- There are no predefined usage patterns for this tool and no fixed SLI/SLOs. However, it is expected that the number of users is small (< 10 users) and usage will be in bursts of activity rather than consistent. It's accepted that results will not be immediate, and users will need to revisit the application to retrieve results.

  What performance and reliability considerations do you have? If the audience grows, what factors of your initial approach may need to change? How do we ensure visibility on system health - both internal resource usage and any third party dependencies?

There are three aspects to this challenge.

1 . **Solution design** What we'd like you to do is to share your thoughts on how you'd go about designing this tool. So please treat this as a system design task. You can find the scope for it below.

2. **Code example** This is also a hands-on role, so we'd like to see an implementation on how a fragment of this could be implemented in python to evaluate your coding standards. We do not expect the entire application to be built, but would like to see an implementation of the following

- Implement a SentimentAnalyzer class that: -
  - Accepts a string input and returns a sentiment score (-1 to +1) and label
  - Includes proper error handling and configuration management
  - Demonstrates your chosen approach (LLM, NLP library, or API service)
  - Also demonstrates the coding standards that you hold yourself to.
- Test coverage for the above implementation. We want you to think about what we need to test - not just the happy path - and how tests can be used to explain and describe intended usage. We like tests that are informative and not just there to achieve code coverage thresholds.

3. **Your feedback** This is intentionally ambiguous to test you and give you room to suggest your own approach. There is not a single correct way to do this. The role will generally be implementing more concrete and defined specifications, but working alongside marketing teams, there is always the chance to see opportunities and volunteer solutions. Some of our most useful solutions have emerged out of observing and talking with stakeholders.

For this challenge though - what would you do if presented with this in a real situation. What would be your approach to ensure you're confident in the expectations? Where would you need support and how would you go about achieving this?

We do not want you to spend too much time on this - it's acceptable to be fairly high level and provide talking points to go into more details during the interview stage. The interaction with stakeholders is also a key part of this exercise. We want to know that we can all work together!

Task Scope

The overarching objective of this project is to fulfil the following requirements:

- The application accepts a list or URLs string for Instagram, Youtube or TikTok posts. This actions is considered to be a single 'job' that is executed
- For each post obtain a list of comments
- For each project, evaluate the comment text to determine the sentiment.
- Once all comments are processed the output results can be retrieved by CSV.
- An API is implemented to allow for:
    - The creation of a `job` execution
    - Retrieval of the current status of the job (ie: is it in progress or has it completed).
    - Once all processing is complete, the csv may be retrieved via a download API route..

These are some guidelines to kickstart your thinking process and provide a rough outline of what we expect to receive in the material you hand over:

- **Application design:** Within the limited information provided - and the expectation that python 3.12 is used - describe toolchain and development experience that is appropriate for this project
    - What standards (if any) would you develop to?
    - Provide an overview of a proposed application structure that is maintainable and intuitive for collaborative team development.
    - Provide example implementations as described above.

- **System architecture**: At a high-level, what does the architecture and dependencies look like? Broad strokes are acceptable here - we can talk through the details and you do not need to account for every resource that could be deployed as part of this solution.
- **Security & authentication**: For cost attribution purposes, different teams can bring in different API keys to use the LLM services. How can you make sure different users have access to the right keys?
- **Performance**: We'd expect to have multiple users (to begin with probably <10) . How would you go about designing a tool that scales for performance, but is not prematurely optimised given the unknown nature of adoption?
    - If it is helpful, you may comment on: latency, reliability, availability, load balancing, caching, event-handling, queues, pub-sub, etc.
    - Do not feel you have to comment on everything or limit yourself to the above. *These are merely suggestions*.
- **CICD**: How would you go about Continuous Integration / Continuous Deployment? Which platforms would you use and how (high-level idea)?
- **Maintenance**: Considerations on maintainability, deployment, monitoring, error reporting, etc.

Task Presentation

We are open to your choice of presentation for the material. Please feel free to use diagrams / sketches / slides, etc. **Note that you will deliver all the materials to us prior to the final panel; please use open source formats (PDF preferably)**.

Code examples can be provided as attached assets or as a link to a public code repository (preferred).

Do not worry about prettifying the materials. We do not expect candidates to spend hours after hours on this, so any materials that enable you to show off your experience and skills will work for us. (If it helps: You can even hand-draw an annotated diagram rather than create a polished gliffy diagram for instance).

Please be specific about the technologies you use (e.g. saying Django for frontend isn't sufficient, we'd want to hear further details on how you'd use Django to deliver which functionalities for what purpose).

This is an opportunity for you to showcase your software engineering skills and familiarity with the cloud computing platform you are most comfortable with currently.

We'd like to see your approach, technology choices, and assumptions made amongst other things.

Please feel free to reach out to paul.watkins@croud.com for questions regarding this assignment.