Welcome to The Standardized Introductory Task for Software Engineers at Algorithmic Sciences.

THIS DOCUMENT IS CONSIDERED PROPRIETARY AND CONFIDENTIAL. DO NOT SHARE IT OR ANY PART OF IT WITHOUT WRITTEN PERMISSION FROM THE COMPANY.

WARNING: Skimmers and people who do not read this document a few times carefully are virtually *guaranteed* to fail this test.

Prelude

This part of our Assessment Center is a real-life exercise which demonstrates exactly what it's like to work with this company, what you are required to deliver when given a task, and how you're being constantly evaluated as an engineer while working for us. The task was taken as one of our company's stand-alone, well-defined projects that was completed in the past, and was standardized into this Introductory Task. Your output based on this test is a very real decision point in terms of whether we continue our relationship with you as an engineer: it does not only show whether your skills, experience and ability to follow instructions make you a good fit for the company, but also whether you're the right cultural fit for our team.

In addition to this, executing the task correctly is going to give you a good overview of what we expect in terms of standards and evaluation criteria when you submit any repository to us in the future. When you work for us as an engineer, you are required to deliver all code to the same exact standards - so this Assessment Center also acts not only as a filter of diligence and focus, but also as a kind of educational tool for the candidates of our engineering department.

Introduction

Congratulations on passing a difficult selection process so far. If you could do that, you can give yourself a pat on the back - you're no doubt a smart person with a lot of potential, high intelligence, and good problem-solving skills.

But before we assign you to a team, we have to test your “experience” level, to make sure you’re aware of the basic subjects and best practices related to the work that we are doing, and that you’re going to be a good fit for our team professionally.

This test is a small development project that you should be able to do within 3-5 days when working on it full-time, and should demonstrate your set of skills when it comes to the basic components of software engineering at our firm. During these 3-5 days (or more) of solving this test we are also checking whether you are capable of delivering 8 hours / day according to our expectation, until the task is completed.

We ask you to track your time on toggl for this project and report every day according to instructions given to you later in this document. If you have contracts signed with us, you’re going to get paid for the task for up to 5 days. If you’re during the interview phase then the test is unpaid. After 5 days, however, if you still haven’t completed it, you will need to do what is left on your own time even if you’ve signed contracts with us already. Just for emphasis, let us repeat: you’re doing this task for the purpose of assessing your software engineering skills and communication, so unless you have a signed contract with us, this company will not owe you anything for work you do on this task.

This company already has a working version of the code, which you can use to benchmark your performance and your proximity to meeting the requirements. You can reach it by connecting to 135.181.96.160 port 44445 (with SSL = False). You can search for any one-line string in the 200k.txt and you should be able to get a “String found” response from the server. Any other query should result in a “string not found” response.

You will find a text file called 200k.txt with sample search strings here:
https://tests.quantitative-analysis.com/200k.txt

Specification

WARNING: Please note that if you post a public repository on any public website or server (including Github, GitLab or anywhere else) with any “work in progress” or final version of this project, thus opening your work up to being plagiarized, or if you try to plagiarize even the smallest detail of any other candidate’s work, your application will be automatically and immediately disqualified.

1. Write a server script that binds a port and responds to connections (an unlimited amount of concurrent connections),

2. Receives "String" in the connection in clear text,

3. The path to find the file comes from the configuration file. The configuration file can have a large number of elements that are not relevant to your server script. The line with the path in it will start with “linuxpath=” and will have the path after it, for example “linuxpath=/root/200k.txt”.

4. Opens the file found in the path, and searches for a full match of the string in the file. Please note: partial matches of the search query string in a line do not count as matches. You should only respond with STRING EXISTS if you can find a match for the whole string as a stand-alone line in the file.

5. REREAD_ON_QUERY option: When set to True, this checks whether "String" exists in the file, considering that the file in the path COULD change every few microseconds. In this case, the code should re-read the contents of the file on every search query sent from the client. When the option is set to False, the file is not expected to change and it's enough to read it one time, on load.

6. The maximum payload size is 1024 bytes. The server strips any \x00 characters from the end of the payload it receives,

7. Responds on the TCP port saying "STRING EXISTS" or "STRING NOT FOUND" (with a newline character at the end).

8. Uses multithreading to accept a large amount of requests in parallel,

9. The script will work on Linux and needs to work with files up to 250,000 rows, with an average of 40 millisecond execution time per file in case REREAD_ON_QUERY is TRUE, and 0.5 ms if it's FALSE.

10. On the TCP output, also show the "logs" including the search query, the requesting IP, the measured execution time, timestamps etc., marked as "DEBUG: ",

11. You are allowed to use native Linux scripts to execute the task (wrapped in Python), core Python, or any libraries. Since speed is a critical part of the specification, do some research and write a small, few-page, well-formatted report in human-readable format on the speeds of the different options benchmarked against each other. Your speed report should cover at least 5 different file-search options and algorithms as you have tried, with the performance of each one, with an extra bonus point available for every file-search option beyond 5. Your report should have a table that shows all the algorithms you have tested against each other sorted based on performance, and at least one chart that compares all the performances of the algorithms against each other as a function of the number of lines in the file. Your final submission should have the code implemented with the fastest option. You should deliver the report in a PDF format.

12. The code needs to run as a Linux daemon or service, with clear installation instructions on how to run it as such,

13. The code needs to be protected from attackers. Please handle your buffer overflows and other potential security issues, and implement SSL authentication between the server and the client, using either a self-signed certificate or PSK as the authentication method. Make the SSL authentication configurable and easy to turn off from the configuration file (True/False).

14. The code needs to be PEP8 and PEP20 compliant, statically typed, docstring'd, and documented on delivery, neatly and professionally packaged. Remember: you're here to demonstrate your level of attention to detail and your diligence. So if you're going to use a tool that promises to simply process your code and make it PEP8 compliant, then beware: if it does not work perfectly, then you're going to get penalized for submitting non-PEP8-compliant code, and in addition to that, you're also going to make the wrong impression on your evaluators, which is going to impact the offer you get. Please always check your PEP8 compliance manually, and do it with the eye of a perfectionist, instead of trusting third-party tools to do a good enough job for you.

15. Robust exception handling and error messages are required, covered with unit tests for:

A. Showing different execution times for different file sizes from 10,000 to 1,000,000 with a client you write for testing purposes and cover these in your speed testing report,

B. Showing different execution times for file sizes vs. number of queries per second, up to the point that the server can not handle it anymore (document the limitations of the software),

C. Demonstrating correct workflow in all paths and cases,

D. Unit testing all exceptions and edge cases is required (use pytest!)

16. To submit your repository, use the one-time registration link that has been provided to you to create an account on our Introductory Task review website. Make sure your repository is in a zip file when it's done, complete with tests, speed report, client.py, and everything else, ready to deploy. Ensure that the zip file does not contain any other zip files, any virtual environment folder or its contents, any git folder, or any other unnecessary files that are not in the specifications. Make sure you have everything else in there, and make sure you check the Evaluation Sheet below to score yourself and see that you are making a passing submission. Also, please make sure not to include any absolute paths in the tests and other places of the repository because you will get penalized for submitting a repo with unit tests that fail. Make the package stand alone and complete, including all test data in the tests directory, and name your files and tests in a way that pytest can collect them properly and run them fast. Also, don't forget to include a requirements.txt and everything else that would normally be required for a professional submission.

Evaluation criterion

Please read this section very carefully, including the scoring sheet below! Note: you will be required to fill out a part of the scoring sheet for yourself when you submit your repository. Failure to understand and follow procedure will result in penalties. Please do your best to read the instructions carefully.

In your decisions, consider as though the software you're working on would be mission-critical for a large and costly process, deployed at a large client. You need to make a good impression and follow all of the policies and coding best practices.

Our criteria for rating your results are going to be the following:

1.) Is every point of the specification met **exactly as defined**? There is not even the **slightest** flexibility of interpretation in any of the specification instructions. If not, then please do not submit a work product until it is: you only have *five* chances to make submissions on our review website, and the primary criterion is whether every point of the specification above is met. If your submission was "close enough" there can be an extra feedback round offered at the discretion of our evaluators (and at the cost of penalty points) - but this is not guaranteed.

2.) How fast were you? It's OK to be slower - the penalties for delayed submission are not very large, as you can see on the evaluation sheet… but if you give up, the company will not continue its relationship with you. In case of delayed - or on-time - submission, please note that if we see you track less than 8 hours a day into the task, that is a red flag: the level of commitment for this company is at an 8 hours/day

minimum, which means less time tracked during this test is not acceptable unless you have other full-time commitments and you have not signed contracts with us yet.

3.) Two of the most common reasons for a candidate to fail this test are either not paying enough attention to detail and the evaluation criterion, or underestimating the amount of time the task requires, and not dedicating enough time to it in general. To give yourself a chance of succeeding, please set aside a consecutive chunk of 5 days for this task, with possibly a few extra days for fixes and delays. Please note the following:
    a.) In order to submit a passing submission, even the best candidates have used at least 4-5 days to get it right,
    b.) We want to see consistency in your delivery: "Missed" / "Skipped" days without reports are penalized by our evaluation system very severely (as you can see on the evaluation sheet below),
    c.) If you have other full-time commitments and have not signed contracts with us, you are still expected to submit a daily report even if you did nothing that day, or it will draw a penalty,

4.) Was your communication on point during the task, and according to specification point 16, did you submit a daily report on every day you're working? You could be doing everything else right except sending daily reports, and your score will be penalized for not following instructions carefully. The daily reporting template is below.

5.) It's OK to ask questions about the specification - but you're expected to figure out the technical details yourself. We will be evaluating your communication while working on the project.

6.) Quality, **simplicity**, straightforwardness, readability, cleanliness, comments, PEP8 compliance, pythonic quality, static typing and programming best practices of the code you submitted. The snippet about coding best practices taken out of our policies document is below.

7.) It's not in the specification, but we do expect you to submit your client.py along with the server, so we have an easier way to test your code. This should pretty much go without saying, though!

Our evaluation system (based on the sheet attached to this document) is completely **objective, pragmatic and meritocratic**, and does not allow our evaluators to make any subjective decisions or exceptions. That means that whoever you are, regardless of location, skin color, age or gender, you will have to pass the exact same bar that everyone else in this company has passed before, and you will receive the exact same offer as anyone else would have received based on the repository you have submitted and our experience working with you up to that point.

Whether you pass or fail this test, if you have been "close enough but not quite there", we will happily disclose the source code of the proper, good implementation of this server solution to you after the test is done - the only condition of this is that you sign a Non-Disclosure Agreement with us before we share it with you. Please note: the decision to disclose the code is discretionary, disclosure is not guaranteed - but feel free to ask.

Coding best practices

Even though we do not expect you to have experience in quantitative finance, as for the quality of the work you'll deliver, it is critical for you to understand the nature of quantitative finance as an industry. There is a big reason why we are explicitly looking for perfectionists, "First class honors" graduates and people with Obsessive Compulsive Disorders: even small coding mistakes in our industry can cost a small fortune for our investors who trade our systems on the markets.

In order for us to achieve these requirements, certain quality controls are in place. Please remember:

1.) **Every function you write needs to be covered with a suite of unit tests. Code that is not covered is as good as code that does not exist.**
    a.) All of the test data need to be part of the test suite (do not hardcode paths!),
    b.) All of the exception-handling mechanisms and edge cases need to be covered by the test suite.
    c.) This one is worth repeating because so many people seem to be making this mistake: if you hardcode paths in your test suite or anywhere else, you will attract a quite hefty penalty for it! Not just that, but you will also look kind of dumb when you submit your repo.

2.) **There are no bugs in properly tested code.**

    If there is a bug report about anything, anywhere, that means there was a test missing. Don't just fix the bug: create the missing tests and cover similar cases to make sure that similar bugs are not hiding anywhere!

3.) **Every function needs to have proper, granular exception handling with intelligent error messages**.

    You need to think of all of the inputs and outputs of all of the functions, handle all of the corner cases with proper exception handling, and give intelligible, helpful error messages to the user that explain to them what they did wrong, and what they need to change in their input to get the output they're expecting.

4.) **Every single line of code needs to be easy to read and PEP8 and PEP20 compliant, statically typed, well documented, human-readable, and covered with docstrings and #comments throughout the code.**

    Again, let's repeat this. Please treat this as a checklist.

    All code in this company, produced by you (or going through your hands) should be:

    A. Easy to read and understand for a human,
    B. PEP8 compliant,
    C. PEP20 compliant,
    D. With static typing (use mypy),
    E. Well documented, where the functional documentation should have examples,
    F. Covered with detailed, verbose docstrings,
    G. With #comments throughout the code.

5.) **If there are multiple ways of doing it, and you're not sure what's the fastest, then the only way to decide is to program and test all the ways.**

    When making certain technical decisions, and writing code that affects the speed of a certain process or application by more than just a few seconds, and you're not quite sure what is the fastest and best way to write that code, then the only way to decide properly is by coding it both ways and generating evidence. Please be careful, conscious, and reasonable with this guideline so as not to get stuck in an infinite loop of

trying to optimize something that is already well done - try to balance the value of your time with the benefit the company is getting from better-performing software. When unsure - just ask.

6.) **English is a beautiful language. Let's use it correctly!**

Sometimes you will find bad English, punctuation mistakes, grammatical mistakes or typos. Sometimes it's even in the function names, and you have to refactor everything in order to fix it! No problem: *always* fix issues like this, when you catch them.

7.) **Using the assistance of Artificial Intelligence in your work**

Keep in mind that while the AI might provide a solution, it's absolutely crucial that you re-engineer, validate and verify all advice and code snippets it generates. Despite these models' complexity and capabilities, they are not infallible and can generate incorrect or insecure code. This will help mitigate any potential vulnerabilities and other unintended consequences.

In order to ensure full transparency and accountability, all interactions with these AI models must be explicitly logged. If a piece of code or a solution is derived from an AI model, that information should be clearly documented and attributed. As part of your daily report, if you used AI for work that day, include a full conversation log of the entire collection of conversations with the AI. In addition to that, attach an AI log with your daily report, with a short list of specifying the problems at hand, the AI's generated final solution, and a brief description of how that solution was applied in your work.

These AI models can be a significant boon, they should not replace human expertise. They are tools that can enhance your work, not do it for you. As a software engineer, you should continue to exercise critical thinking, problem-solving, and utilize your domain expertise alongside AI-generated suggestions. The goal is to strike a balance between human input and AI assistance, to leverage the best of both worlds for the benefit of the company.

Daily Reporting Template

On top of the time tracking, please answer the following questions in email every day. It should really not take more than 5-10 minutes, and it's going to give us important insight into your progress and involvement with the company.

1.) How many hours in total did you track, and which task(s) have you been working on today for the company exactly, and what have you achieved for it? Who gave you the task you're working on? How many hours/days is the task estimated to take according to its title, and how many hours have you spent on it so far?

2.) If you are working on a task that you haven't answered before: Please describe your detailed understanding and interpretation of the task, and your plan of what you intend to do, and when. How are you going to solve the problem? How many hours do you think it should take? When exactly should we expect the issue to be solved?

3.) Are you stuck with anything, or do you have any questions or obstacles that stop you from proceeding or being productive in any way?

4.) Show us a screenshot of some of the documentation or code snippets that you're most proud of - as a result of today's work. Was there any difficulty figuring it out? What was your thought process, and what did you go through when doing it? (Please note: do not copy & paste code, as it'll break the formatting and can make the rest of your email difficult to read. When we say a screenshot, we mean a screenshot).

5.) Issues and questions solved with AI assistance: Provide a short list of problems or questions and final solutions that you have used in your work, and attach a full log of the conversation with the AI. In addition, create a document that contains 3 sections for every issue: the first part should have a brief description of the issue you were dealing with, Section 2 should detail the solution AI gave you, and Section 3 should show the solution you have implemented after consulting AI. Keep the report brief and do not copy & paste more than necessary from the code base to show your implementation.

Please note: it's very important that you honestly and verbosely go through these 4 questions with us in email every day, so we have a deeper understanding of where you are and what you're doing. It is also extremely important that you do not miss a single day of sending this report to us.

If you have tracked even just one hour that day, please take a minute to write the daily report, but also note that according to the Scoring Sheet below, your "consistency and availability" are both being evaluated, which means that if you deliver less-than-8-hour-days, then you're going to get penalized for the number of hours that would complete your days to 8 hours. This also includes weekends, holidays, breaks, accidents and other circumstances that might affect the way we perceive your reliability. The only case we might make an exception to this rule is if you're admitted to a hospital while completing the Assessment, which will be a fact that we will expect you to prove with hospital admission and release papers and medical findings (an expression of our company having an evidence-based culture).

In addition to all of this, please do not expect us to respond to your daily reports. If you have sent them to the right address, then we will have them in our inboxes, and when you've submitted a final repository, we are going to go back to them and evaluate your reporting and consistency.

We look forward to getting your daily reports. Please address them to the following email: candidate_reports@algosciences.com.

In terms of choosing your subject lines: please just use the "Reply All" function of your email client as you go through the entire process, including all of your daily reports. Do NOT change the subject line of your original email that you have received from us.

For your final code submission, use the one-time registration link sent to your email (when you were assigned the task) for creating your user account, or visit our task review website and log in if you have already created an account. After a successful authentication process, you will be able to make your submission for a review (which will most likely be reviewed within 24 hours if you have not exhausted your chances for submission).

If you miss submitting your daily reports, or if you do not follow the instructions related to the final submission given in the feedback to you, then you most probably will not get a response to your final submission from the company at all.

Asking questions & Reporting problems

As you go through this process with us, you are going to notice that we are not very responsive when it comes to your daily reports and your questions related to the task. The reason for this is two-fold.

The first reason is that we regard this document as self-sufficient, stand-alone and complete, covering every angle and every question in an absolutely unambiguous way. That means it probably has *more* information than what you could expect to receive when being on the job in a live context, and if this information is not enough for you to do a perfect job of delivering this repository, then it follows that you are not the right fit to work for this company and you would probably require more oversight, help, mentorship and coaching from our senior staff who are busy doing their jobs and leading complex projects and large teams assigned to them.

The second reason is that we run a lean and bootstrapped organization, where all of our time is managed very efficiently - and we are simply too busy to respond to every email, report, or question by every candidate, or provide candidates with detailed feedback, oversight, help, mentorship or coaching.

For these reasons, in order to take the assessment, please consider the following:

1.) Whenever you decide to go through the Assessment Center, you are welcome to take part in it, and if you were invited to this test then this requires no further approvals or communication from us or from you.
2.) Schedule a chunk of at least 5 consecutive days as soon as possible, but not further than 3 weeks away from receiving the invitation to start the Assessment Center, and let us know the dates you have scheduled. Expect no response to this from us, but know that we have read your email and we know that you're participating in the Assessment.
3.) Submit your reports on time (not more than 24 hours after the reported day ended), track your time, and follow the template and the guidelines and instructions provided precisely. We are not looking for approximations here - any divergence from the reporting template and other procedures will be penalized as it is required by this procedure. Throughout your work process, expect no response from us. Again, as far as questions go, it is assumed that this document contains everything you need to know. The reporting process was not created for you to ask questions at this stage, but for you to be able to show us your diligence, consistency, reliability, and thinking process as you work towards the result.
4.) After submitting your final repository and the relevant parts of the Scoring Sheet below, don't worry, you will receive feedback from one of our Evaluators, even if we haven't communicated with you throughout points 1, 2, or 3 in this paragraph.

With all of the above said, from the point where we send you the URL of this document, up to the point where you have "final submission" of a repository that you believe should pass all of the evaluation criteria, please be self-sufficient. Do not expect any responses, "encouragements", approvals or feedback from us while participating in the test. If you want to receive an offer from us then your job now is to start generating enough of the right kind of evidence to the company in the ways described in this document that show the company that you are consistent, reliable, can write solid code, and are able to follow procedure.

If you have submitted a "final submission" and have not received an evaluation within 72 hours, please follow up. Send an email to the address provided above with the subject line: "Urgent: No response to final submission."

About the submission and feedback process

Before submitting your repository, make sure that it fits *all* of the specification points precisely, and double-check your submission based on the specification points above, the evaluation points above, and the black parts of the scoring sheet below.

If you are going to submit on a "best effort basis", and "see if it will pass", then please be aware that you are wasting both your time, and ours. Passing with this submission is a *binary*, you either pass, or you don't. There's no subjective element to our evaluation system - it is entirely mechanical, and if your repository does not follow *all* of the specification points, and *all* of the evaluation criteria, then instead of just letting you move on to the next stage and receive an offer, we're going to need to give you feedback about it.

The first feedback round is "free", and the rest are penalized. If a candidate receives a round of feedback, makes a new submission, and they did not do a good job of incorporating the feedback into the new repository and the same feedback needs to be repeated the second time, then the evaluators are required to stop giving feedback and will have no choice but evaluate one last submission, and see if it passes or fails.

Incorporating feedback sometimes can take several hours or even days for the candidate, depending on the quality of the first submission; however, please consider that a senior member of our team takes their time and goes through your code to evaluate it against the criteria in this document, and you should respect their time and efforts and do your best to work with them in order to get your repository up to par. They are not trying to be difficult… trying to work with you in order for you to be able to submit a passing repository so that the company can hire you!

For many people who participate in this process, this is actually a valuable learning opportunity and experience, and even if they don't get hired, they walk away being better developers.

With all of this said, it's worth noting that 3 of the most common subjects of first-round feedback that our candidates receive are the following:

1.) PEP8 compliance-related feedback: you may have used some kind of tool, instead of familiarizing yourself with PEP8 and going through your code to make it compliant. Tools are great - it's not a problem… The problem is if you take them for granted, and don't double-check their performance.
2.) Speed-test report-related feedback: a lot of people seem to skip or misinterpret the specification point that requires them to create a speed testing report and benchmark the speed-related performance of different algorithms against each other.
3.) In the majority of submissions, we see the REREAD_ON_QUERY functionality misinterpreted. Please go back to this specification point and invest time and attention into making sure that you have interpreted this specification point correctly.

Please make an extra effort and try to submit a passing submission on the first try in order to save both your time, and your Evaluator's time, with special attention to detail especially when it comes to the 3 points above. The advantage of doing this is that once you start working for the company, you're going to have a

much easier time, since we actually use the same exact standards to evaluate all of the code that anyone ever submits, before we merge it into our repositories!
For more instructions about how to submit your final repository for review, visit the link here.

Please spare our space

Due to the large volume of applications we receive day in and day out, we ask you to kindly refrain from attaching files and repositories to your emails and instead provide us with links (web, Dropbox, Google Drive, or any other services) to those files to avoid our email boxes from filling up unnecessarily.

Scoring sheet

Note: You get one round of feedback without penalty. The rest of the rounds, especially if some element of feedback has to be repeated, will be penalized.

Evaluation points marked in green are to be filled out by the candidate and sent via email (to the same email addresses above), right after the final repository is initially submitted to the review website. Please make a copy of this document and submit either a docx, or a PDF file as a link (web, Dropbox, Google Drive, or any other services) for the filled version of this document. Please do not zip this scoring sheet up along with the repository submitted to the review website. The scoring instructions explain how exactly to calculate the value of the "Score reached" column. Enter numbers into the Score Reached column. Do not edit anything or enter any words into this column or anywhere else, other than the correctly calculated scores.

	Scoring instructions	Score reached
PROCEDURAL SECTION		
Did the candidate need to be reminded to fill out the green questions in this sheet for the purpose of evaluating themselves according to the instructions in the green paragraph above?	If not: 0 points, if yes: **immediate disqualification.** Lack of attention to detail, no culture fit. We will not be able to proceed with the application of this candidate at this time.	
Did the candidate mention "GitHub" in their communication or otherwise fail to follow the submission instructions detailed above?	If not: 0 points, if yes: -5 points (lack of attention to detail, no culture fit)	
Was the project delivered within 5 days or less?	If yes: 0 points. If not: -1 point for each day of delay (lack of experience)	
Did the candidate track at least 8 hours / day on every day working on the task?	If yes: 0 points. if not, -1 point for every hour missed (lack of diligence, no culture fit)	
Did the candidate complain "the server is not online", or some variation thereof?	If no: 0 points, if yes: -5 points, they did not understand the specification in detail (lack of attention to detail, no culture fit)	
Did the candidate submit a valid "speed testing report" with a table, a graph and at least 5 different algorithms benchmarked on both reread_on_query options in a PDF format?	If yes, and the rest of the specification is completed: 4 points, if not: -10 points (missing one of the main objectives). For every algorithm tested over 5: +1 point, up to +10 points. Delivery in wrong format: -2 points.	
Did the candidate submit the client.py along with the server?	If yes: 0 points. If not: -2 points (lack of attention to detail, no culture fit. it should, logically, be a given that we shouldn't be expected to code our own client.py for evaluating the candidate's code.)	

Did the candidate submit a valid daily report for every day of work within 12 hours of finishing the workday, and is the report according to the template?	If yes: 0 points, if not: -5 points for every day missed (not following instructions, no culture fit)	
Did the candidate submit time tracking logs from hubstaff or toggl along with their daily reports?	If yes: 0 points, if not: -5 points for every day missed (not following instructions, no culture fit)	
Did the candidate deliver the project within less than 40 hours?	If yes: 1 point for every hour up to 12 extra points. If no: the penalty for delayed delivery is 1 point per day according to a previous evaluation criterion, as long as the candidate tracks 8h/day minimum up to the point of delivery.	
Did the candidate send the daily reports to <u>all</u> of the specified email addresses for evaluation?	If yes: 0 points, if not: please do not be surprised, if you do not get a response to your final submission at all… the right person might simply have not seen it.	
Did the candidate adhere to the procedural requirements related to subject lines of reporting & submission emails throughout the process?	If yes: 0 points, if not: -10 points for every email that does not adhere to the requirement.	
Did the candidate disclose and prepare appropriate reports on their use of AI while completing the test task?	If yes: 0 points. If report is missing: reviewer needs to conduct an AI-related fraud investigation if candidate passes.	
TECHNICAL SECTION		
Are 1 to 16 points of the specification met with functional, working code?	If yes: 16 points. If 9 is not met: -11 points. If any other point is not met, candidate receives 0 points	
Did the candidate submit code that only matches the whole line in the text file, and does not give false positives for partial matches?	If yes: 0 points, if not: -5 points (lack of attention to detail)	
Did the candidate submit full installation instructions on how to install the code as a Linux daemon that always runs?	If yes: 0 points, if not: -5 points (lack of attention to detail)	
Is the SSL authentication implemented correctly?	If yes: 0 points, if not: candidate receives 0 points, specification is not met.	
Is the code pythonic, simple, straightforward, well-documented, readable, clean, commented, pep8 compliant, and using the best practices outlined?	If yes: 0 points, if not: -6 points for every offense (lack of attention to detail, no culture fit)	
Is the code statically typed using mypy?	If yes: 0 points, if not: -6 points (lack of attention to detail, no culture fit)	
How many rounds of feedback did the project require to get up to par after submission?	If perfect on the first try: 0 points, otherwise -1 point for every round of feedback.	
Do the unit tests the candidate wrote cover 100% of the functionality in the specification with all the corner cases?	If yes: 0 points, if not: -6 points for every 10% of coverage missing (lack of experience)	
Is the exception handling of the server robust and covering edge cases with intelligent and useful error messages?	If yes: 0 points. If the evaluator's client.py crashes the server: -10 points for every error that triggers a crash (lack of experience, missing objective, no culture fit). Otherwise, -10 points for lack of robust exception handling.	

Did the test suite run flawlessly in a stand-alone fashion at first try *before the first feedback round* or did the candidate hardcode any paths in the repository either in the test suite or anywhere else or made some other mistakes?	If it ran: 0 points. If test suite fails for whatever reason on the Evaluator's computer due to hardcoded paths or other issues: -5 points for every failing test.	
Did the candidate communicate with bad English spelling or grammar with team members, comments, documentation or code?	If no: 0 points, if yes: -1 point for every mistake, typo or grammatical error (no culture fit)	
Was communication on behalf of the candidate accurate, verbose in terms of technical details, courteous and on point in general?	If yes: 0 points. If too terse or otherwise not on point: penalty of up to -10 points.	
Did the candidate fail to type our company name correctly or introduce "changes" and "alternate versions" to the company name at any time throughout the application process?	If no: 0 points. If yes: -5 point penalty for every occurrence.	

Max score: 20.
Passing score: any score above 0.