Zedulo Technical Test

It's a pleasure to see that **you** decided to take us up on this test. The purpose with this is to give you a chance to demonstrate your capabilities in the best possible way. It is therefore not a test of the type that you have done at SHS/Uni but rather intended for you to show what you are capable of accomplishing within a given time frame, just like you would at work.

The test contains a mandatory base section that you should start with. After that you select between different add-on topics that you want to focus on. You can select a few and spend all time on them but do them very precise, or you can be more generic and do many but more coarse grained. It is up to you.

The rules are simple but **important**:

0) If you have technical problems with the setup or need additional ports to be opened, programs to be installed etc. then you can always get help from support@zedulo.com. Requesting help will not be seen as negative for you. However, you will not get more time as compensation for the delay in requesting help.

1) No help from anyone else is allowed. If we notice or suspect that you have taken help from anyone else at any level then you will fail the test instantly and you will also not receive the compensation. This is done so we can see *your* capability, not anyone else.

2) You are allowed to copy/paste from e.g. stackoverflow or other dev communities, but only smaller section (<50 lines per sections) and you then have to mark that section as copied and reference exactly where you took it from. You can of course then change that code, but you should keep the reference and that you initially copied the code.

3) You are not allowed to take any substantial material from outside, this include work that you have done yourself earlier. If in doubt then just as if you can copy code.

4) You are allowed to copy pictures / art, sound clips etc. that is not code. We do not evaluate your artistic capabilities. However, make a note of what you have copied.

We request that you email support@zedulo.com at the end of each day with a short summary of your activity that day including a rough estimate of how many hours you worked that day. Do also remember to take backups of your work on a daily basis to prevent any loss of data.

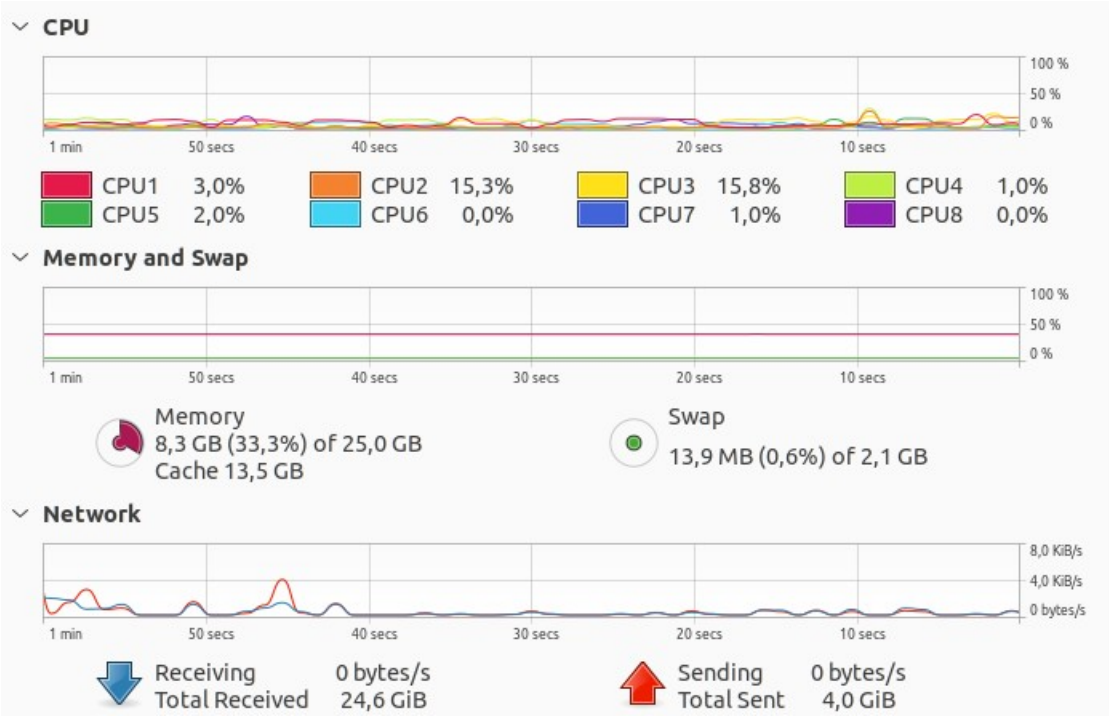
Your access to the compute instance will be removed exactly a week after the test starts. We will then schedule a call to walk through your work together with you. We then expect you to know all the code that you have produced, code that you have "borrowed" and discuss how you came up with the solutions you did.

Good luck!

Baseline

Server monitoring is an essential part of running services in a data-center. The figure below shows one such example for linux that monitors cores, memory usage and network throughput. In this test you will develop a monitoring system for the server compute-instance that you are running on. This can include CPUs core usage, memory as well as power consumption and temperatures.

The data should be fetched from the server instance and then displayed in a user interface. You can yourself decide if you want to develop a web based interface or a mobile app interface. What we want to see in the end is a functioning website or app that can be accesses on the server from internet.



It is up to you to decide how the data should be displayed, it could simply be text output, a pie chart, most recent data only or running a graph with historic data as the linux example above uses. It is also up to you how to enable/disable display of different parameters. Making it look nice is not required but a later optional target.

The input data can be any of the following, with the simplest one at the top. The solution should preferably use real-time data but it is not mandatory:

a) Hard-coded or random data that you make yourself

b) Hard-coded data in a local file

c) Real-time data from your compute instance using linux /proc/stat file, this is updated every 1 second. You can access this on the local instance by typing the command “cat /proc/stat” and you can place a local soft-reference to the file in any directory by typing “ln -s /proc/stat statistics.txt”.

See links below for how to interpret the file, as you can see total core usage, idle, system, etc. is available.

https://rosettacode.org/wiki/Linux_CPU_utilization

https://support.checkpoint.com/results/sk/sk65143

d) In addition to the /proc/stat above you can also add data for the full servers temperature and power consumption. This is can be fetched from URL below, updated every 5 seconds:

http://zedulo.com:9999/sensors.txt

http://zedulo.com:9999/sensors.json

Add-ons

The baseline function should be further enhanced in the following areas. Pick one or multiple areas and make your solution as good as possible.

1) Authentication

Add authentication mechanisms to access the interface. This should likely include login/password but can also include additional functionality such as setting/changing the password, storage of the password in a file, verification of the set password strength, prevention mechanisms against brute-force attacks on the password, 2-step- authentication, etc.

2) Security

Safety and security, this includes both to ensure that the communication between client and server is secure but also to check for incorrect input that could cause be used as an attack. Buffer overflow, and other type of standard attacks needs to be handled.

3) Test and functional validation

Make an automated test mechanism that allows as much functionality to be validated without user-interaction. The intent is to run this from e.g. different browsers or on a regular basis when updates has been made.

4) Performance validation

Design a setup that validates functionality under load from many users or users with high level of interaction. How do you prevent overload?

5) Mobile web interface

Or desktop / tablet interface if you initially did Mobile. The point is that the interface should adapt to the screen size. All features should be available in all interfaces with the same overall feel.

6) Mobile app

Develop a mobile app (android) that has a limited feature-set compared to the web interface. Or, if you initially developed a mobile app then you develop a web-interface with limited functionality vs the initial app.

7) User Interface look-and-feel

Make the user-interface impressive from a look and feel perspective. How cool can you make it look?

8) Language flexibility

Make the interface flexible from a language aspect where all text is stored in a separate database/file to allow for easy translation and support for multiple language. The interface should support at least two languages to demonstrate the flexibility and have an easy way to switch between them.

9) Export

Support export of information to e.g. pdf, excel, word or similar.

10) External API

Provide an external API that allows external services to access the back-end and request information.

11) Server notifications

Allow for background services with external notifications when certain key parameters are reach, for instance max temperature, max CPU load etc. This can trigger a message on the display or an email to be sent.

You will need some additional information to be able to send external emails but let support know and you will be given this additional information.

12) Database

Feed data into a database and allow for long-term history to be retrieved and displayed.