Stage 1: Design and implementation of a 'vanilla' version of client-side simulator with a simple job dispatcher (DUE 5pm Monday, Week 7)

This assessment item accounts for 20% of the total marks (100) of the unit. As this simulator design and implementation is to be a **group** work, each group shall make **only one submission** (no individual submissions). The marking will be conducted in two steps: (1) assessing the working of code at the demo (at your allocated workshop in Week 7), and (2) offline marking on the report, code and individual contributions.

Your main task in this stage is to design and implement a plain/vanilla version of client-side simulator that acts as a simple job dispatcher. This version of client-side simulator shall implement:

1. The ds-sim simulation protocol as described in ds-sim User Guide. Briefly, it connects the server-side simulator, receives jobs and schedules them
2. A simple job dispatcher, called 'allToLargest' that sends all jobs to the first one of largest* server type (e.g., ID 0 of x*large* server type in `ds-config02.xml` in ds-sim User Guide).

   Appendix G in ds-sim User Guide is a sample simulation log of Stage 1 implementation.
   Note that your log might be different, for example, if you schedule jobs (SCHD) without requesting resource information (GETS). Your client-side simulator should work for *any simulation configuration*. The client-side simulator (`ds-client`) provided in the ds-sim project repository (https://github.com/distsys-MQ/ds-sim) can be used as reference implementation. In particular, when running the server with '`-v brief`', your simulation logs should exactly** match those generated using the reference implementation.

** The largest server type is defined to be the one with the largest number of CPU cores.*

*** The format and the order of messages are of importance. For example, the order of 'JOBN', 'SCHD' and the actual execution of job. Job details, such as submission time, runtime and resource requirements should also match. If you run the server with '-v all', ds-server might show log differently depending on how the client makes scheduling decisions.*

Deliverable

The submission is to be **a single compressed file** of your project git repository ('Download ZIP' in GitHub and 'Download repository' in Bitbucket).

Your submission shall contain at least:

- The **source code** of your client-side simulator
- The **report**.

** This source code will be the one to be used in your demo (NO changes allowed after the submission).*

Your submission may include other files, such as `makefile` or a shell script for installation and a user guide.

In addition to your submission, **your group needs to run a demo** at an allocated workshop in Week 7. The actual selection and allocation of a workshop for your demo will be done prior to Week 7.

Suggested headings for Stage 1 report (STRICTLY NO TITLE PAGE!)
(**max. 5 pages** including everything; the entire submission is 5 pages or fewer)

- Project title: no more than 100 characters (approximately 14 words), e.g., Cloud job scheduler or Cost-efficient resource allocator for distributed systems.

- Group members: full names (in the order of given name and surname) and student numbers, e.g., Young Choon Lee (12345678).

- Introduction (½ page): What this project (focusing on Stage 1) is about, including the goal of the project and Stage 1.

- System overview (½ page): high-level description of the system (both client-side simulator and server-side simulator with the focus being your client-side simulator), preferably, with a figure (your own, not one in ds-sim User Guide) showing the workflow/working of the system.

- Design (1 page): design philosophy, considerations and constraints, functionalities of each simulator component focusing on the client-side simulator.

- Implementation (2 pages): brief description of any implementation specific information including technologies, techniques, software libraries and data structures used. How each of components/functions of your simulator is implemented including who is in charge of which function(s) and how they have led the design and development.

- References including project git repository/wiki, e.g., GitHub and Bitbucket.

* The numbers of pages for each section are also a suggestion.

Marking rubric (in percentage)

85 to 100

Work in this band presents a full and comprehensive account of all requirements outlined above. In particular, the efficient and elegant design and implementation of fully functional client-side simulator with the correctly working allToLargest policy should be evident in (1) the source code, (2) report and (3) demo. The clear demonstration of teamwork should be evident in all these three components, e.g., git commit history and log of other project management tool.

75 to 84

Work in this band presents a clear account of all requirements outlined above. In particular, the good design and implementation of fully functional client-side simulator with the correctly working allToLargest policy should be evident in (1) the source code, (2) report and (3) demo. The sufficient demonstration of teamwork should be evident in all these three components, e.g., git commit history and log of other project management tool.

65 to 74

Work in this band presents an adequate design and implementation of working client-side simulator. The appropriate demonstration of teamwork in all three assessment components should be evident.

50 to 64

Work in this band presents a decent design and implementation of working client-side simulator. The appropriate demonstration of teamwork should be evident.