**Practical project 2018**

**(due 11:59pm Friday 25th May - end of week 12)**

See also: [**Clarifications**](http://teaching.csse.uwa.edu.au/units/CITS3002/project2018/clarifications.php) and [**Resources**](http://teaching.csse.uwa.edu.au/units/CITS3002/project2018/resources.php).

**The aim** of the project is to develop a network-based application to test students through their ability to correctly answer multi-choice questions and to complete short programming challenges written in one of Java, C, C++, or Python. Authenticated students interact with the testing system using a standard web-browser.

The web-browser will communicate with a *Testing-Server* (software), which first authenticates a student's session and then navigates the student through a sequence of multi-choice questions and programming challenges, receiving the student's attempts, and totalling marks awarded for correct attempts.

The *Testing-Server*, in turn, will communicate with a *Question-Server* (software) which generates a sequence of multi-choice questions and programming challenges for each student (communicated via the *Testing-Server*), and marks and executes students' attempts.

The web-browser, *Testing-Server*, and *Question-Server*, all execute on different computers (hardware) and, thus, must communicate using network protocols. Because students communiate with the testing system using a web-browser, communication between the web-browser and *Testing-Server* will require the HTTP(S) and HTML protocols, while the design and implementation of a protocol between your *Testing-Server* and *Question-Server* is up to you.

Pre-registered students must first login to the *Testing-Server* using a text-based name and password. They may then navigate, forwards and backwards, through a sequence of, say, 10 questions which are either multi-choice questions, or ones setting a short programming challenge (see Clarifications for some examples). If the student's first attempt at a question is correct, they receive 3 marks, 2 marks if their second attempt is correct, and so. Once they have attempted all questions (either correctly or after 3 attempts) they may no longer submit answers. At any time, a student may login, see their progress and total mark to-date, or logout.

The *Testing-Server* only manages the testing of students, and has no 'understanding' of the questions or how they are marked. Similarly, the *Question-Server* has no 'understanding' of the students and their marks, and only generates and marks questions. In an effort to reduce collusion amongst students, the *Question-Server* should choose its questions randomly from a question-bank (simple approach), or generate repeatably randomized questions (more complex approach). The marking of multi-choice hopefully requires no explanation. The marking of the programming challenges will require the *Question-Server* to *execute* the student's attempt, and to compare any output against the anticipated answer (which may, itself, have to be generated by executing a standard solution).

**Important dates and project submission**

The project contributes **30%** of your mark in CITS3002 this year, and has a number of important dates and deadlines:

* by **5pm Wednesday 23rd May** (week 12)   
  complete the project demonstration booking sheet, which will be pinned up outside of CSSE Lab 2.3.
* by **11:59pm Friday 25th May** (end of week 12)   
  submit via [cssubmit](https://secure.csse.uwa.edu.au/run/cssubmit) your team's project submission. Only one team member needs to submit the work. By this deadline submit all source code, digital certificates, and scripts that you wish to be assessed.
* during **the week commencing Monday 28th May** (week 13)   
  project demonstrations.

**Constraints**

The **constraints** of the project require that:

1. the *Testing-Server* and *Question-Server* must execute on different computers (hardware), and must not assume access to any shared (networked) files.
2. the *Testing-Server* and *Question-Server* must be written in different programming languages (selected from Java, C, C++, or Python).
3. the *Question-Server* must generate and assess questions written in Java, C, C++, or Python (only one needs to be supported).
4. your project may be developed for Linux, Windows, or macOS (or a combination).
5. your software must support two or more students simultaneously attempting questions.
6. all network traffic must be encrypted using SSL.
7. your *Testing-Server* does not need to produce a fancy web-interface - basic HTML involving forms, a *textarea*, and *radio*-buttons will be sufficient.
8. the *Testing-Server* does not need to support full user or password management - just logging in and out. Assume that all (named) students and their unchangeable passwords are pre-recorded (there is no need to use UWA student numbers or Pheme authentication).
9. neither the *Testing-Server* nor *Question-Server* require a sophisticated database to manage the students, marks, and questions - simple text files will be sufficient.
10. you should employ the core networking functions (classes, methods, libraries,...) of your chosen programming languages and not employ specific 3rd-party frameworks or resources to complete large parts of the project. If in doubt, please ask.

**Project demonstration**

Your team must also arrange a demonstration of your software, for **up to 30 minutes, in week 13 (or possibly week 14)**. The role of the demonstration is for you to demonstrate how much of the project's goals you have met, and to answer questions to demonstrate your understanding. It is not essential for all team members to be at your demonstration, but ensure that those attending will be able to represent the team.

During the demonstration, you'll be logging into your account on your chosen operating systems. A booking sheet will be provided closer to the deadline. During the demonstration, your team should:

* briefly describe design decisions and assumptions that you have made in your project. You must clearly explain how it works, identify what is being protected by your system, how that protection is provided, and identify any known weaknesses with your approach or its implementation.
* re-compile your programs, and initialize and invoke the two server programs. Describe the contents of necessary directories and files.
* demonstrate, through a small number of examples, how someone may use your software system.
* do not prepare a PowerPoint-style presentation.

**Working in teams of up to 3**

The project may to be undertaken in teams of up to 3 students (no, not 4). The motivation for working in small teams is to enhance communication skills amongst students, and to enable you to attempt a project considered of greater difficulty than would normally be reasonable for the time available. It is anticipated that this project will require **20-30 hours** of study by each member of the 3-person teams.

The project is worth **30%** of your mark in CITS3002 this year, and the distribution of marks within your team (typically one-third each) must be agreed to by all members of your team.

Only one team member needs submit files using [cssubmit](https://secure.csse.uwa.edu.au/run/cssubmit). Ensure that all students' names and student numbers appear in all submitted materials.

Anyone needing to find a project partner should read [*Students seeking project partners*](http://undergraduate.csse.uwa.edu.au/units/CITS3002/partners.php) as soon as possible.

**Clarifications**

Please post requests for clarification about any aspect of the project to [*help3002*](https://secure.csse.uwa.edu.au/run/help3002) so that all students may remain equally informed.   
Clarifications will be also added to the [**project clarifications**](http://teaching.csse.uwa.edu.au/units/CITS3002/project2018/clarifications.php) webpage, and additional materials may be addded to the [**project resources**](http://teaching.csse.uwa.edu.au/units/CITS3002/project2018/resources.php) webpage.

Good luck,