COMP3030J Server FAQ

# What tech stack should we use?

It is strongly recommended to use a tech stack that you are familiar with. We assume the following have been taught to you:

* Flask (backend framework) **OR**
* Spring boot (backend framework)
* Jinja/Spring MVC + HTML 5
* Bootstrap (CSS framework)
* SQLite3 (database)
  + Flask-SQLAlchemy to connect Flask to SQLite3
  + sqlite-jdbc to connect Spring to SQLite3
* jQuery + AJAX
* Linux

You can install Flask or Spring extensions like Flask-Dotenv, Flask-Babel, Flask-WTF, etc.

You may use raw SQL rather than SQLAlchemy if desired.

Note: **it is not necessary to use a frontend framework like Vue or React.**

# Are we allowed to use Django/Nuxt/other frameworks?

(see below)

# Are we allowed to use Docker/microservices?

(see below)

# Can we use whatever framework/library/database we want?

Since the focus on COMP3030J is about project management and soft skills, **it is strongly recommended that you** **use tools you already know** instead of learning a new framework for your project. The project can be successfully completed using tools that have already been taught to you such as Flask, SQLite, and jQuery. There is no need to learn any new frameworks.

# Do we need to learn a new backend framework to complete this project?

No, you should use frameworks you already know or are familiar with. There might be some things your preferred framework cannot do well — you might need to install a few new libraries. However, there will be very few instances of when it is required for you to learn a completely new framework to successfully complete your project.

# How do I ssh into the server?

Use the ssh command using the hostname that we have provided for you.

Here is an example of using the ssh command:

$ ssh student@csi123-42-vm0.ucd.ie

**Remember to set a strong password when you first login to the server**.

# How do I create a strong password for the server?

Use the pwgen application (available online: <https://8-p.info/pwgen/>) to generate a **long**, memorable password. The password:

* must be **at least 16 characters** long
* must **NOT** be derived from your name or your group name
* must **NOT** be a word in the dictionary (either English or Chinese)
* must **NOT** be on the list of the [most common passwords](https://en.wikipedia.org/wiki/List_of_the_most_common_passwords)

Then use the [passwd](https://www.cyberciti.biz/faq/linux-set-change-password-how-to/) command in Linux to change your password.

# How do I deploy the project to the server?

Follow the web server instructions on Moodle.

# Can we use port 8000? Port 5000? Port 8080? Port 3036? Port X?

**No**, you may not use these ports outside of the server. The only **publicly-accessible ports** that your application can use on the are **port 80 for HTTP traffic**, and (optionally) **port 443 for HTTPS/WebSocket/WebRTC traffic**. It is possible to complete your project using only port 80 for HTTP traffic. Port 22 will also be open on your server so that you can manage it with SSH.

**No other publically-accessible ports will be made available**.

Note: applications may use additional ports *internally* -- that is, only available to the VM and its services. For example, a database engine (e.g., MySQL, Redis) might use a different port *inside the VM*, and this is normal and perfectly acceptable. However, a database server **must never listen to connections on the public internet** since doing so would be a major security vulnerability.

**DO NOT touch the firewall settings!**

**DO NOT install an additional firewall like firewalld!**

# How do I make port 5000/port 8000/port 8080 available to the public?

If you are using this port for web traffic, **use a reverse proxy like Nginx** to tunnel outside traffic from port 80 to your internal port. See the web server instructions on Moodle for more details.

# What is a reverse proxy?

A reverse proxy is an application that listens to web traffic (HTTP/HTTPS) and **reroutes it to a backend application** (e.g., your Flask application). Advantages of using a reverse proxy include:

* better, easier security — the reverse proxy can do *TLS termination*, which means it can handle HTTPS requests and translates them to simpler HTTP requests to your backend.
* performance — you can deploy multiple instances of your backend application, and the reverse proxy will perform *load balancing*, which distributes requests to your backend instances to make the best use of your available resources.

**The main reason you will use a reverse proxy in COMP3030J is security.** Listening to port 80 and port 443 requires superuser privileges (i.e., sudo), and **it is forbidden to run your backend server using sudo** because minor bugs in your code can lead to a hacker completely taking over your server. To avoid this risk, backend applications must be run without sudo, which requires a reverse proxy to be placed in front of it to handle incoming requests and reroute them to your application.

Nginx is a popular, well-supported reverse proxy, and static file hosting server.

# How do I use Nginx?

The basics for how to use Nginx are provided in the web server instructions:

<https://csmoodle.ucd.ie/moodle/pluginfile.php/178762/mod_resource/content/0/COMP3030J%20Web%20Server%20instructions%20%282023%29.pdf>

# How do I access the website running on our server?

Once you have followed the steps in the web server instructions, your website should be available at the *hostname* we gave you, for example, http://csi123-42-vm0.ucd.ie. Flask might print a message stating that the server is accessible at one or more IP addresses and ports, but **this is false**. Always use the full hostname for the website, just like any other website on the internet. **Do not connect to the server using its raw IP address and/or port number**.

# How do we enable English and Chinese translations?

If you are using Flask (recommended) use Flask-Babel. Follow online guides for more information: <https://python-babel.github.io/flask-babel/>