Part 0 (0 Points): Set up your local machine

You will have a local working environment on your personal machine and a live remote server to deploy the web application on. Similar to professional environments, you will do all development locally, and then deploy to the live server. *Be sure to test on the server well ahead of the deadline.* This semester, we will be using Vagrant as the main development environment to make setup easier (trust us, it can be a pain). Vagrant uses a config file provided by us to set up an Ubuntu virtual machine (VM) on your laptop, and you will use this VM for all projects in EECS 485.

Part I: General Overview

At the end of this guide, your virtual machine will support a MySQL server and an isolated Python web server. Both these items will also be required on a remote, live server, but that comes a little later. The Python web server is set up using virtualenv, which creates isolated working environments with independent libraries. We also use pip, which is a package installer for Python (used to install virtualenv).

Part II: Vagrant Setup

First, we will install Vagrant and setup a VM on your machine. At the end of this section, you will be able to SSH into an Ubuntu VM that has all system requirements already installed.

- Install VirtualBox
- Install Vagrant
- If you are on *Windows*, you must install git bash. This is a shell (like command prompt) and you must open this before continuing. Type the following command in git bash itself.
    - **During the install, make sure you chose "Use Git and optional Unix tools from the Windows Command Prompt"**
- Download the starter files from Google Drive and cd into this folder
    - i.e. `cd /path/to/vagrant` in terminal
- Run `vagrant up` to start the Virtual Machine (VM), while in your current working folder.
    - The first time you do this, Vagrant downloads a base image for Ubuntu, which may take a while, but it only does this once. It uses the `Vagrantfile` as a config file to see what to install (check out the file to understand it).
    - The first time you run a VM, Vagrant also runs `vagrant.sh` to set up the environment (read the file and the comments). This also sets the username/password of the MySQL server to be `root` for both.
- After this process, the VM is ready for work (way easier than the traditional setup).
    - The root project folder on your local computer is synced with the `/vagrant` folder on the VM but be sure to use Git to stay in sync with your teammates.

- You can login in to the VM using `vagrant ssh`. You are now inside the Ubuntu VM. Do `cd /vagrant` to go to the main working directory.
- Once you're done working, shutdown the VM with `vagrant halt` or it will keep using your RAM.
- Or run `vagrant destroy` to completely destroy the VM if you want to recover your disk space as well. Your local files won't be affected.
    - Note that the next time you run `vagrant up`, it will re run vagrant.sh since you destroyed the whole VM.

Part III: Python Flask Setup

Now that everyone has a consistent dev environment, we will set up the Python Flask web server to develop the project in.

- Go into your working directory inside the VM. So run `vagrant up` if you haven't already, then do `vagrant ssh` to go into the VM. Lastly, type `cd /vagrant` to go to your main directory.
- Type `virtualenv venv --distribute`
    - This creates an isolated virtual environment so you can install local packages only for your project without messing with global, system wide settings.
- Type `source venv/bin/activate`
    - This command will activate the virtual environment we just created.
    - You MUST do this every time you go into a fresh terminal window (because the packages we install will only be present in this isolated environment).
    - You should see something like `(venv)your-computer:python username$` (varies)
- Next, type `pip install -r requirements.txt`
    - We give you an initial set of packages that are required for Project 1. This command will install all those packages in the virtual env.
    - In the isolated virtual environment, you can install Python packages using pip. For example, you will be building the app using Flask, a Python web framework. You can install it by running `pip install Flask`, which puts the package in the venv folder that you can reference from your app.
    - You want your teammates to stay in sync so run `pip freeze > requirements.txt` when you add new packages so your group can do the same.
- Now to start the web server in the starter files, run the following commands:
    - `cd p1`
        - **This is the folder where your Git repo should lie. You should also duplicate this folder for each new project but connect to a new repo.**
    - `python app.py`
        - Go to localhost:3000 in your native browser (not VM) to see it working!

Optional Part:

Vagrant is essentially creating an Ubuntu VM on your laptop so we can separate all 485 development. However, if you already have an Ubuntu machine or don't want to install a VM, you can do it all on your native OS. Follow this:

- Install Python and Pip (follow the instructions for your OS, but python may already be installed by default on your machine):
    - Mac OS X
    - Windows
    - Linux
- Install virtualenv: `sudo pip install virtualenv`
- Install MySQL (follow for your OS). During this process, you will be asked for a password. It's best to do `root` for both username/password in **local environments**.
    - Mac OS X (Section 2.4.2 works best)
    - Windows (Follow the simple installation method)
    - Linux
        - If you encounter an error, try `apt-get install build-essential python-dev libmysqlclient-dev` and then try Google.
- Go back to Part III and continue there.

If this is your first time doing package installation, you will undoubtedly encounter errors. Google and StackOverflow will be your best friends and you should do that before reaching out to staff.