SYSC 4806 – Lab 5: Continuous Delivery of your AddressBook App

If you've made it this far, congratulations: you have turned your AddressBook program into a very simple web app! In this lab you are going to set up the continuous delivery pipeline for your app, so that it is tested and built on every commit, deployed, and easy to update. Here are the steps to follow:

1- First, if you haven't already done so (should be second nature by now!) enable git version control for your project, and add a corresponding remote GitHub repo from IntelliJ. You should now be able to commit your changes and push them to GitHub.

2- Let's write some tests, to replace and automate the manual creation of HTTP GET or POST requests, and the verification of their response. This tutorial shows you how you can run an integration test of your web app, and also a lighter way, using a mock, to run those same tests without having to start up a web browser every time. Look into RestTemplate for the programmatic composition of REST calls. If your tests pass, move to the next step.

3- Now set up continuous integration on GitHub. For this we are going to use GitHub Actions (on a free GitHub account, you get 2,000 minutes free per month). First, read this to acquaint yourself with GitHub Actions, then follow this to set up the Java with Maven CI workflow for your repository. The "Actions" tab in your GitHub repo page should normally also recommend the "Java with Maven" starter workflow to you, and choosing it will generate the YAML file and store it in the right directory for you. But make sure the content of the YAML file it generates is what you want! For example, check that the JDK version it uses is the same as the one you have set in your pom.xml file. Also, it might include additional steps that might require GitHub permission tokens, and that we don't need to worry about here. Now, if you push your code along with the tests you wrote on step 2 to GitHub, your GitHub Actions workflow should automatically run those tests and build your app if the tests pass. Check if that's the case by going to the Actions tab of your GitHub repo. You can even include a badge that displays the status of your build in your repo's README.md file! Figure out how to do it!

4- Unfortunately, Heroku no longer offers a free tier for hosting web apps, and the other free options require a credit card, so for this year at least, the following is OPTIONAL (and the TAs have not been asked to offer support for this): see if you can create another GitHub Actions workflow to deploy your application to a cloud hosting service! For Java applications, currently Google Cloud and Azure offer free hosting (but both require credit card information). Other free hosting services such as render.com require that you build a Docker container first. This link documents some of your options. Now try adding a simple feature to your address book, and test it. For example, add an address field to your BuddyInfo and write the corresponding test(s). Push your changes and see if your deploy shows the updated version

Success? Show your work to the TA, i.e., show your repo with a successful build involving a few integration tests. If you can't, invite your dedicated TA as a member of your GitHub repo (the GitHub handles for the TAs can be found on Brightspace in the "Course Details" section). The TA will check that you have correctly set up the GitHub Action for CI, and that there are a few integration tests that are being run by CI.