Assignment 5: RESTful Web Service Implementation + Docker

** This assignment will combine a few things that you have learned in this class and will require a little learning on your own. Do your best and be creative. If you need help, ask sooner rather than later in Slack. Myself and your classmates are here to help and do not wait until the last minute to do this assignment.
** You may work in two person teams, if you plan to do so please email me to let me know who you are working with. You may NOT work in multiple teams.
** You may use Python, JAVA or node.js (javascript) for this assignment.
** I have listed some tutorials below, but you may need to Google some on your own.

What You Will Do:

- You will create a RESTful web service that runs in a docker container.
    - **Video: What is docker? -** https://www.youtube.com/watch?v=dz5_IsWlfTU
    - **Review Lecture Slides + Materials:**
        - Week 9 (Part 2 of 1) - Microservices Sections
        - Whiteboard Examples:
            - REST vs RPC Example Calls: Customers + Orders
            - REST Route Examples
            - More Rest Endpoint Examples (Orders & Products)
- Your web service will contain four GET routes:
    - One that displays a collection of records (i.e /customers returns a collection of customer records)
    - One that displays a single record that corresponds to an ID
        - **Example:** If I created two routes; /customers and /customers/35 (note that 35 is the ID of a given customer in my database)
    - On that displays a collection of records for a given entity
        - **Example:** /customers/35/orders (note that 35 is the ID of a given customer in my database)
    - One that displays a single record from a collection of a given entity
        - **Example:** /customers/35/orders/13 (note that 13 is the ID of a given order submitted by customer with ID 35 in my database)
- The data returned from your web service routes MUST be in JSON or XML form.
- You MUST create a simple client application (i.e. web page) to load your results from every service endpoint you have created.

- **The Database:**
    - You will create a hard coded JSON file based database **OR** a relational database (i.e. Postgres or MySQL) as the backing datastore for your web service routes.
        - **[IMPORTANT]** If you choose to use a real database, it must run in a separate docker container and your service must communicate with the container for database access. You must demonstrate this in your screencast video.
        - **[IMPORTANT]** Also note, your data model is something you make up. Meaning you can store a collection of cars, customers, food items, restaurants, video games, sports teams, etc. Be creative :)
        - **[IMPORTANT]** If you use a JSON file based database, note this is similar to what the presenter did in the GraphQL video we watched in our last class. He used a JSON file as a database for his demo.
- You are required to present your work in a screencast video. This is not optional!

Tutorials:

- Docker
    - What is docker: https://www.youtube.com/watch?v=dz5_IsWlfTU
    - Installing Docker: Windows - https://www.youtube.com/watch?v=wCTTHhehJbU
    - Docker Tutorial (Step by Step) - https://www.youtube.com/watch?v=Vyp5_F42NGs
    - https://blog.talpor.com/2015/01/docker-beginners-tutorial/
    - https://docs.docker.com/engine/getstarted/
    - https://hackr.io/tutorials/learn-docker
- Python RESTful services using Flask:
    - https://code.tutsplus.com/tutorials/building-restful-apis-with-flask-diy--cms-26625
    - https://impythonist.wordpress.com/2015/07/12/build-an-api-under-30-lines-of-code-with-python-and-flask/
- Node + Express REST API Example
    - https://closebrace.com/tutorials/2017-03-02/creating-a-simple-restful-web-app-with-nodejs-express-and-mongodb
- Node Simple RESTful API (shows using json file as DB)
    - https://www.tutorialspoint.com/nodejs/nodejs_restful_api.htm
- Dockerize your Flask App
    - https://www.smartfile.com/blog/dockerizing-a-python-flask-application/
    - http://containertutorials.com/docker-compose/flask-simple-app.html
- Docker + Spring Boot (JAVA)
    - https://spring.io/guides/gs/spring-boot-docker/
- **Freeware screencast-o-matic to use for screencasting**
    - Website: https://screencast-o-matic.com/
    - Tutorials: http://help.screencast-o-matic.com/