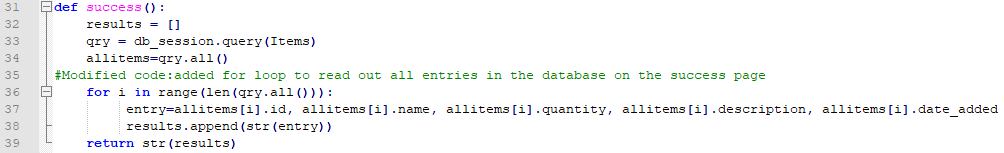
**List of issues found and changes made (in order that they were found)**

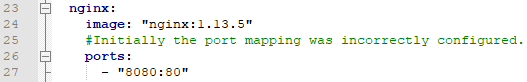
1) Improper flask code that lead to unreadable database on success page

Changes made: Modified app.py file to make database readable on success page(Line 34-38:added for loop and changed results=qry.all() to allitems=qry.all())



2) Connectivity Issues between nginx and host

Changes made: Modified docker-compose.yml to correct port configuration for nginx (Line 27: 80:8080 to 8080:80)



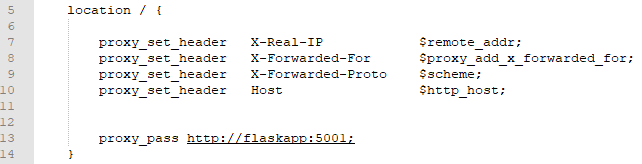
3) Connectivity Issues between nginx and flask

Changes made: Modfied app.py by adding the right port to connect to ngnix(Line 45: added port=’5001’ ). Note: I also added the debug=True line but that does not affect the websites functionality.



4) Improper coding in nginx that lead to failure of loading success page properly

Changes made: Modified flaskapp.conf file to allow the success page to run successfully (Deleted: proxy\_set\_header Host $host;)



**In depth notes on my approach in troubleshooting the code**

**Intro prompt**

Imagine you're on an engineering team that is building an eCommerce site where users can buy and sell items (similar to Etsy or eBay). **One of the developers on your team has put together a very simple prototype for a system that writes and reads to a database.** **The developer is using Postgres for the backend database, the Python Flask framework as an application server, and nginx as a web server. All of this is developed with the Docker Engine, and put together with Docker Compose.**

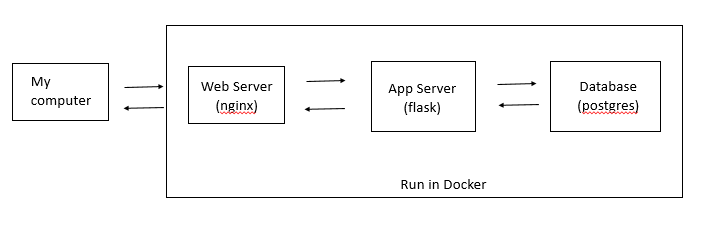
Unfortunately, the developer is new to many of these tools, and is having a number of issues. The developer needs your help debugging the system and getting it to work properly.

**Notes on the prompt**

The goal of this exercise is to create a functional website in which you add items into a database and can also read what is in the database. In addition, the introductory prompt shows me what software was used to create the code for the puzzle (POSTgres, nginx, FLASK, python, DOCKER) and what they were being used for. Below are the steps that I took to solve the puzzle.

**PART 1: Exploratory analysis**

Since I am completely unfamiliar with app/web development, the first thing I did was figure out on a high level what the components of an app/website are, what they do, and how they communicate with each other. From here I was able to sketch out a diagram of the architecture (Figure 1). Without knowing what was inside the code, I posited that the issues could be divided into code that were either related to or not related to communication issues.



**Figure 1. Architecture of website**

I have experience with coding in python but was unfamiliar with flask, postgres, nginx, and docker. So I did some research to get a high level overview what each component does—especially docker since it was harder to grasp. From my initial research, I realized that I could run everything on docker and did not have to download and install postgres and nginx.

**Part 2: Troubleshooting the code**

**Step 1) Initial run of all code through docker**

I ran the three lines of codes without trouble on the windows command prompt1 to run the system.2 After running the system and error message popped up for nginx (…Cannot start service nginx: driver failed programming external connectivity on endpoint systems….) which appeared to be an indication of connectivity issues.

1 Initially the second line would not work as I was using git bash to run the code. I did some research and realized this was a bug with git bash in windows and switched to command prompt.

2 Instead of running “docker-compose up –d” I ran “docker-compose up”. This allowed me to see error messages.

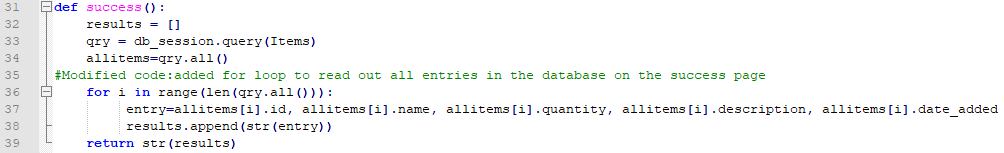
**Step 2) Running Flask and postgres only**

I isolated the process by running only flask and postgres on docker. I did this for two reasons: (1) to confirm my suspicion that the problem was coming from nginx and (2) make sure that the website was functional and that there were no errors unrelated to connectivity in the flask/postgres code itself.

I ran the code “docker-compose up flaskapp” and got a message that the code was running on <http://0.0.0.0:5000>/, however, I could not access the website through that specific address. After some additional research on networking and ports, I realized that in order to look at the webpage on my computer, I had to map the port that flask was running on to my localhost. I temporarily modified the flask app ports in the compose file (ports: - ”5000:5000”), and I was able to load the root page on my web browser (localhost:5000).

The root page worked, however, I noticed on the success webpage that I was getting an empty list. One of the goals of the exercise was to be able to create an app that could read the database as well as write into it. I figured out that the code was creating a list of “each entry” but not the entrys’ specific columns and associated data. Therefore, I modified the code below to read out of the database on the success page. I could now say that the flask and postgres part of the pipeline worked in docker. 1

Changes Made: Modified app.py file to make database readable on success page(Line 35-38:added for loop to retrieve data from database and on Line 34 changes results=qry.all() to allitems=qry.all())

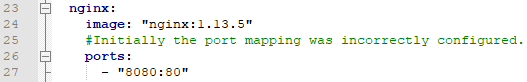


1Even though the code worked, I still took a deeper dive of the python code associated with running flask and postgres, figured out how each module interacts with each other, and commented the code accordingly. I also took note that flask connects with postgres through the database.py module.

**Step 3) Identifying connectivity issues of nginx**

From step 1, I posited that the suspected issue with the code was a connectivity issue either between the localhost and ngnix or nginx and flask or perhaps both. I found that nginx was connected to both the localhost and flask in the docker-compose file and through the .conf file. The .conf file was configuring nginx to communicate through port 80. I realized that the nginx ports were configured incorrectly in the compose file. Ports: [HOST]:[CONTAINER]

Changes Made: Modified docker-compose.yml to correct port configuration for nginx (Line 27: 80:8080 to 8080:80)



From step 2, I knew that Flask’s port was defaulting to 5000. However nginx through the .conf file was communicating with nginx through port 5001 “proxy\_pass http://flaskapp:5001;”. In addition, the docker file was configuring flaskapp to communicate through port 5001 “EXPOSE 5001”.

Changes Made: Modified app.py by adding the right port to connect to ngnix (Line 45: added port=’5001’ ). Note: I also added debug=True as well but that does not affect the websites functionality.



**Step 4) Final check and further troubleshooting of nginx**

The code finally loaded the website locally through <http://localhost/8080>. I double checked and made sure that the code was fully functional. While the root page worked, upon adding a database entry, I was redirected onto a page that could not be found: Localhost%2Clocalhost:8080/success

From Step 2, I knew that postgres and flask were working correctly, therefore I figured the issue was because of nginx. In addition when I manually typed in to my web Localhost:8080/success, I found that the success page was working as intended, again indicating the issue was with nginx.

Changes Made: Modified flaskapp.conf file to allow the success page to run successfully (Deleted: proxy\_set\_header Host $host;)

