This independent study has been a success and taught me valuable skills for web development. The final product is a personal portfolio LAMP stack web application, hosted through Heroku which is accessible via masonbrill.me and the files can be accessed here https://github.com/Mason-Brill/Portfolio there exists another repo for the phpmyadmin docker container that is also hosted but cannot make public due to sensitive information within it. Starting from the beginning, I created three different docker files. One is for a phpmyadmin container to make changes to the database which is also accessible online but will only be up and running when need be in the interest of cost and security. The second docker container is the MySQL database itself containing the projects, more specifically each project's image number, title, skills, and description. This container is accessed by the frontend via PHP. This leads me into the third and final container which is an apache/PHP container. This container is the frontend of the website. It contains an index.php file which is the file that the website loads for the user to see. The file index.php consists of html code alongside php code. The php code utilizes an environment variable which is the url of the database. From here it conducts a mysqli query of the database which searches for the "projects" table. Then the php performs a while loop to iterate through the database and grab each project's details returning it in html. I was able to easily apply css classes to these html elements and style them to my liking. There also exists a ruleset that is enabled in case the user's viewport is under 1000 pixels, i.e. on a phone. You can also see this ruleset enabled if the user shrinks the size of their browser window on their computer.

There existed numerous hurdles throughout the development lifecycle that I needed to discover a solution to. There existed little to no challenges when developing the application in a local environment. The most prominent number of issues occurred during deployment to Heroku, this is due to all the changes that need to take place when you convert from a local environment to an online one. The first problem that occurred during deployment was that the docker container would be running, but none of my website's files would be in it. I needed to add a line of code inside the docker file that copied all of my files from the directory where the docker container was being instantiated, to inside the container itself. The second problem that I encountered was the most challenging and time consuming, it was querying the database to get the information onto the website itself. I continuously had the same error which was an invalid url for the database. The first step in solving this issue was getting the MySQL database container up and running properly. Inside the dockerfile for the database, I needed to provide it with the environment variables for the database which consisted of the name, user, password, root password, and allow_empty_password. I also had to expose port 3306 and run the docker-entrypoint.sh script. The final addition to the dockerfile was running a command that starts the mysql server. From here the MySQL database was up and running accordingly. I now had to make sure that my PHP code that connects to the database was correct as I was still getting the same error. To start, I used a getenv() function to get the database url. Then I parsed this url accordingly using builtin functions and storing them to variables which then allowed me to perform a mysqli query and get each project's information from the database. Now I was

getting a new error which is the third problem I encountered and it stated that the database was empty. This was strange to me because I had an init.sql script that was supposed to run when the container was initialized to populate the database. Nevertheless, my solution was to run the phpmyadmin container to manually populate the database myself. This was not too difficult as I just had to run the container on another web address, which is ultimately a separate deployment from the web application itself. Once I was able to do this I encountered yet another error. The fourth major problem was that phpmyadmin was telling me my login credentials were incorrect. This made no sense as I was the one to create the credentials. I noticed that within the url environment variable for the database, heroku was outsourcing my application to AWS(Amazon Web Services), and within this url I was able to see that my username and password were being hashed because the database url follows a specific format that contains the username and password for the database. I plugged these hashed values into phpmyadmin and I was able to login, and therefore populate the database. Thus resulting in a complete full stack web application working harmoniously.