Gaby Masak

D602 – Deployment

Task 3: Program Deployment

10/26/2024

Report: API Creation and Deployment

**Requirement A: Gitlab Subgroup and Project**

Link to GitLab Repository: <https://gitlab.com/wgu-gitlab-environment/student-repos/gmasak/d602-deployment-task-3/-/tree/working_branch?ref_type=heads>

Repository Branch History:

A screenshot of a computer

Description automatically generated

Figure 1: Screenshot of GitLab Branch History

**Requirement B: FastAPI Package in Python**

Gitlab Link to Version 1: <https://gitlab.com/wgu-gitlab-environment/student-repos/gmasak/d602-deployment-task-3/-/commit/f88d695008d4054ba60bf3d6d7712540eae1ab5b>

GitLab Link to Version 2: [https://gitlab.com/wgu-gitlab-environment/student-repos/gmasak/d602-deployment-task-3/-/commit/577d5aa0c19bdb4ed187806129b8d7083e85f79a](https://gitlab.com/wgu-gitlab-environment/student-repos/gmasak/d602-deployment-task-2/-/commit/118a3154358916df5b5bab272db9bf17de6575e4)

GitLab Link to Final Version: [[Troubleshooting 500 Server Error -- Resolved (d6e5c481) · Commits · WGU GitLab Environment / Student Repos / gmasak / D602 Deployment Task 3 · GitLab](https://gitlab.com/wgu-gitlab-environment/student-repos/gmasak/d602-deployment-task-3/-/commit/d6e5c481016551ce529922ddc26e473b27e02d02)](https://gitlab.com/wgu-gitlab-environment/student-repos/gmasak/d602-deployment-task-2/-/commit/e904c6003152c1ff5087734f31ab9b86f522677d#d406fb1394122462ad4731706bd8d2a7d4a7725d)

Screenshot of History:

A screenshot of a computer error

Description automatically generated

Figure 2: Screenshot of Final Commit for Requirement B

**Requirement C: Unit Tests**

Gitlab Link to Version 1: <https://gitlab.com/wgu-gitlab-environment/student-repos/gmasak/d602-deployment-task-3/-/commit/3b4a4a960d2fd37d0870d720e0b3038603039ed4>

GitLab Link to Version 2: [https://gitlab.com/wgu-gitlab-environment/student-repos/gmasak/d602-deployment-task-3/-/commit/0aa874308700a7ffe57bfd4b33524f6cb1f15cc2](https://gitlab.com/wgu-gitlab-environment/student-repos/gmasak/d602-deployment-task-2/-/commit/e904c6003152c1ff5087734f31ab9b86f522677d#d406fb1394122462ad4731706bd8d2a7d4a7725d)

Screenshot of History:

A screenshot of a computer

Description automatically generated

Figure 3: Screenshot of Final Commit for Requirement C

**Requirement D: Dockerfile**

Gitlab Link to Version 1: <https://gitlab.com/wgu-gitlab-environment/student-repos/gmasak/d602-deployment-task-3/-/commit/37310c570c760ba5a3194965a05dd74a102665a1>

GitLab Link to Version 2: [https://gitlab.com/wgu-gitlab-environment/student-repos/gmasak/d602-deployment-task-3/-/commit/104350a6249eb1235b46d5f0cba225b5a1999f34](https://gitlab.com/wgu-gitlab-environment/student-repos/gmasak/d602-deployment-task-2/-/commit/118a3154358916df5b5bab272db9bf17de6575e4)

Screenshot of History:

A screenshot of a computer

Description automatically generated

Figure 4: Screenshot of Final Commit for Requirement D

**Requirement E: Explanation of Code and Challenges**

There were three components that were addressed to write the relevant code for task 3. First, I cloned the GitHub project to my IDE. I had some issues with the pipeline as seen in Figure 5 below. This was resolved and allowed me to edit the API\_Python\_1.0.0.py file.

A screenshot of a computer

Description automatically generated

Figure 5: Screenshot of Pipeline Errors

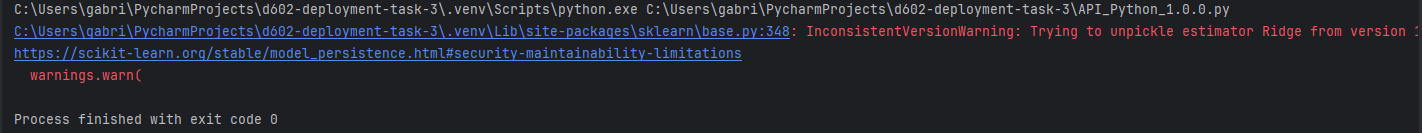


Figure 6: Screenshot of Console Errors

A screen shot of a computer

Description automatically generated

Figure 7: Screenshot of Terminal Errors

I ran into several issues, as seen above in Figures 6 and 7. Troubleshooting led me to change the required ski-learn version in the requirements.txt file from 1.3.2 to 1.5.2. I also had to change the file name from API\_Python\_1.0.0.py to API\_Python\_1\_0\_0.py as the system could not locate the file with multiple periods. This allowed me to test whether my written methods for requirements B1 and B2 were functional. I used the following command in the terminal:

uvicorn API\_Python\_1.0.0.0:app –reload

The Bash command allowed me to use a web browser, namely variations of the URI <http://127.0.0.1:8000/>, to directly assess the methods. I was able to verify that “/” returned a JSON message that the API was functional, as seen in Figure 8. I was also able to verify that “/predict/delays” accepted three parameters and returned a prediction of average departure delay in minutes as a JSON response using the browser screen in Figure 9 below.

A screenshot of a computer screen

Description automatically generated

Figure 8: Screenshot Displaying API Functional

A screenshot of a computer

Description automatically generated

Figure 9: Screenshot of API in Browser

For the unit test requirement, I created a file called test\_api.py. I created three unit tests which tested to make sure that the predictive model ran correctly, and that it would only accept time and airport codes in the correct format. As seen in Figure 10, I needed to troubleshoot as only 3 of the 4 tests passed. I was able to resolve this by modifying the API\_Python\_1\_0\_0.py file with troubleshooting and checking the server.

A screenshot of a computer program

Description automatically generated

Figure 10: Screenshot of 3 of 4 Tests Passing

Lastly, I created a Dockerfile for requirement D. This file references the requirements.txt file, the API Python code, and runs a web server to allow FTTP requests to my API. I used two Bash commands in the terminal to build the Docker image and run the Docker container:

docker build -t my-fastapi-app

docker run -p 8000:8000 my-fastapi-app

**Requirement F: Video Capturing API**

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=1b53a806-4cf9-4b16-8e25-b22f011d2958>