COE 892 - Distributed Cloud Computing

Winter 2025

Dr. Khalid A. Hafeez

Lab 4 - FastAPI

Submission Date: March 30, 2025

Eric Muzzo

501019745

Introduction

This lab modifies lab 2 by creating and deploying an API server using the python fastAPI library. Similar to lab 2, this lab follows a request-response format between a client and server. For the purposes of this lab, we will refer to the rover and server as one entity. The server entity is responsible for defining HTTP endpoints that can be reached by a client to interact with the same type of data as the previous labs. The client is built using the python NiceGUI library, and allows an Operator to control the entire application on a web browser. I then deploy the API to an Azure web app that holds the docker image of my application and makes it publicly available.

Program Design & Overview

For instructions on running the applications, please visit the source code repository.

FastAPI Server

As mentioned, FastAPI was used in python to build the RESTful endpoints needed for this application. Due to the size of this application, all data is stored in memory rather than a database. Any internal failure of the application results in complete data loss. Conveniently, fastapi uses uvicorn under the hood to run the application, which handles any unforeseen failures and keeps the application running. My application is distributed amongst several subpackages and modules for organization and readability. Most of them, like routers, are self-explanatory. I will make mention of the structures and models packages. Structures holds the class definitions for the map and rover, along with some other minor objects. Operations on any one of these resources is done through class methods. This way, using these objects throughout the application is standardized and clear. The models package holds pydantic model definitions for the various resources in the application; rovers, map, mines. This allows the data validation on request and response data to happen automatically thanks to pydantic, which works extremely well with fastapi. Thus, I don't have to write a bunch of annoying if statements to ensure inbound or outbound data is correct.

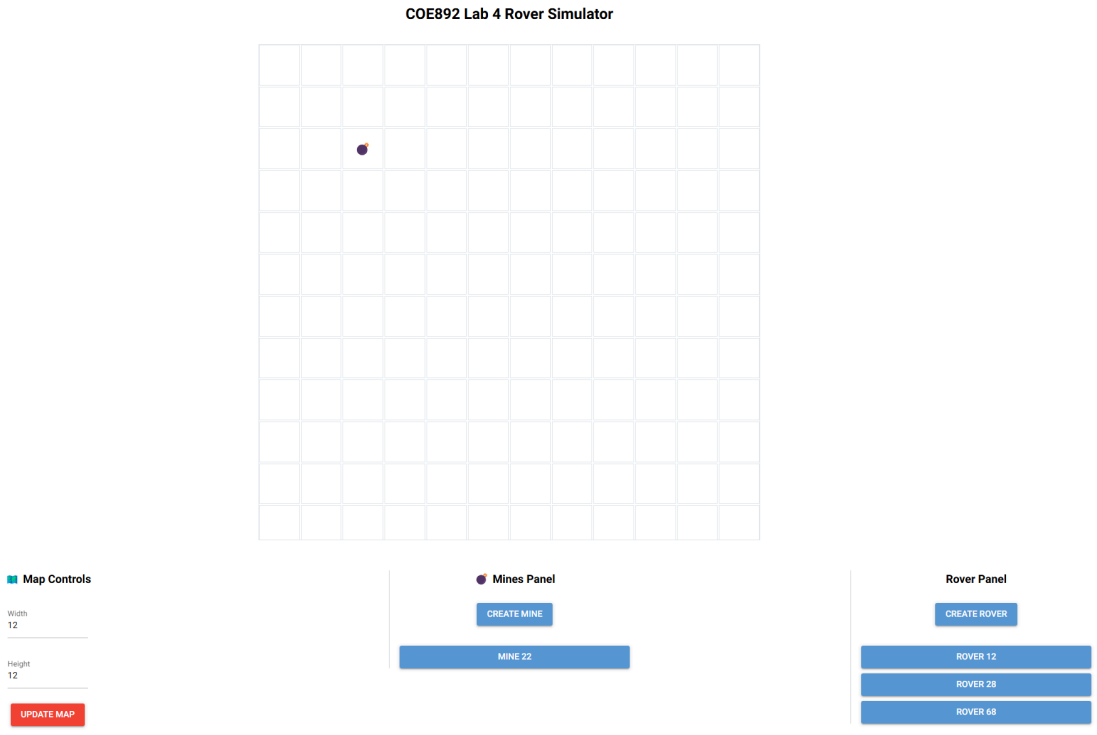
I don't feel the need to restate the required endpoints needed for this application, since they are in the lab manual and are also in the lovely fastapi interactive docs, which you can view here (provided it is still deployed at the time you are reading this. Otherwise, you can run the application locally and visit the /docs endpoint on your own).

This entire application was deployed on Azure as a web app. To do this, a dockerfile is used to build the docker image. This is then pushed to the remote container registry on Azure, which is then used as an image on my Web App For Containers resource.

Operator

The second component of this lab was to build a client frontend that actually interacts with the server side in a visually appealing way. To do this, I decided on using NiceGUI which is a python library for easily building frontend web apps. I will say this now, frontend is not my specialty nor my interest. I have never really built a frontend before, and I did in fact rush this part. The code in my submission is not easy to understand. I do hope that one day I can come back and finish this because I would like to get this working.

Nevertheless I implemented (tried to) a main map grid that dynamically updates with each action. I created 3 separate control panels for the map, mines and rovers. I was unable to get the websocket communication aspect working on the front end. Although it works perfectly on the backend, which I tested with postman, I unfortunately ran out of time on this and could not get it to work. You can interact with the different UI elements to see what each one does. There are various pop up menus that have forms to fill in data and execute commands. To start a rover, click on its list record in the control panel and place it on the map. You can then select it on the map and dispatch it, watching its path in real time.



Conclusion

Overall, I enjoyed this lab very much for several reasons. First, my background is in backend development, and FastAPI is something I have worked with and know well. I take pride in building proper larger scale backend services, and this lab allowed me to practice doing just that. I believe that this is probably the most realistic task I have been given in my engineering undergrad, and so I wanted to put my best foot forward and create the most readable and well structured backend that I could. I think I can say that I did that for my API, but not the frontend. Second, I got to use real cloud computing services to deploy my application. I appreciate tasks like this because they gave me exposure to tools that will actually be used in my future, and I can confidently say that I learned how to use Azure quite well. The UI component of my application was, in my opinion, a failure and did not work out the way I intended. Primarily due to time constraints, but also due to lack of experience.