

Testing

Our efforts in testing our software revolved mostly around using sample manifest.txt files to better understand how we wanted the functionality to behave vs how it actually behaved. Below are sample images of the results of various tests which showcase output that will be used for grid display on the website to be used by Mr. Keogh's employees.

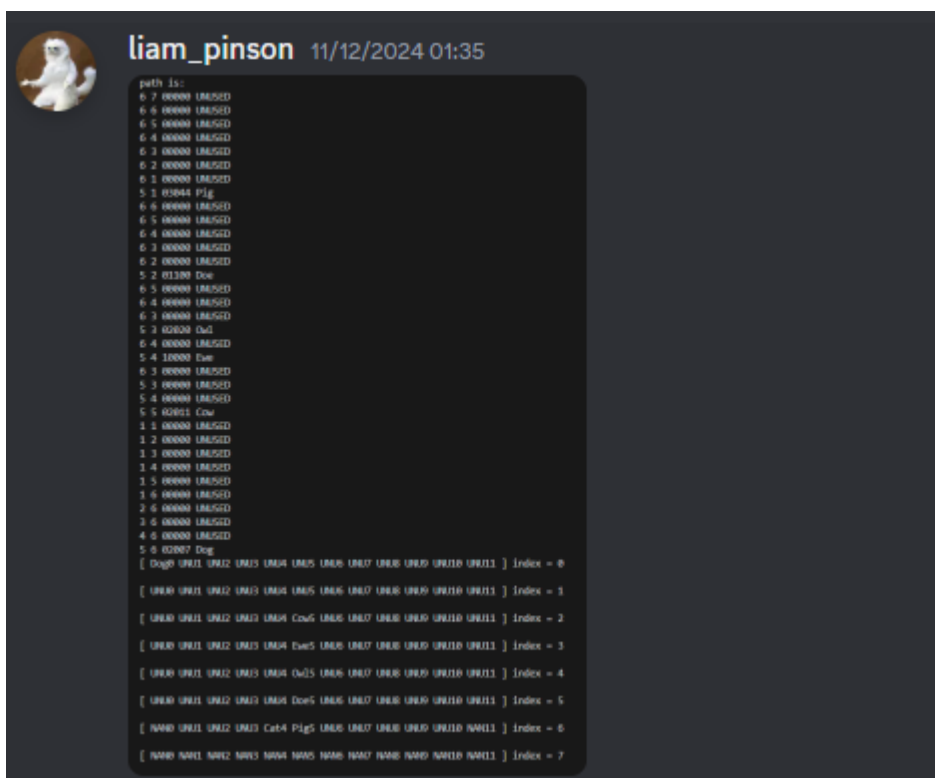
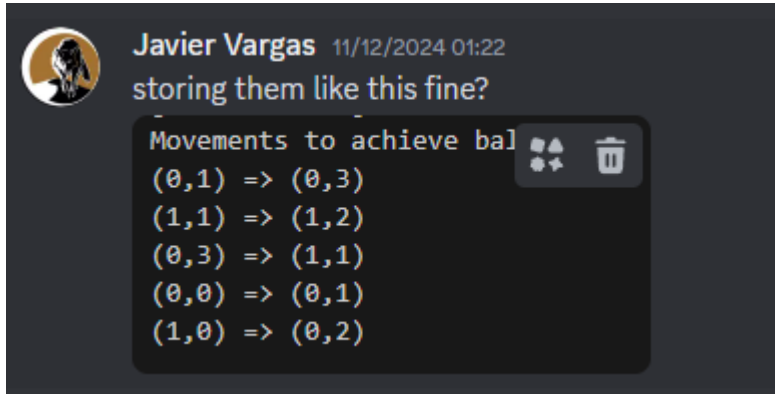
Example 1 shows the result of reading in a sample manifest file to our program during a balance operation. "Loc" here represents the coordinates of each cell on the ship. If one exists in a cell, each container's name and weight is also displayed alongside its coordinates.

Example 1

```
loc 1,1  
['NAN', '00000']  
loc 1,2  
['Cat', '00096']  
loc 1,3  
['Dog', '00008']  
loc 1,4  
['Pig', '00004']  
loc 1,5  
['Hen', '00004']  
loc 1,6  
['Rat', '00001']  
loc 1,7  
['UNUSED', '00000']  
loc 1,8  
['UNUSED', '00000']  
loc 1,9  
['UNUSED', '00000']  
loc 1,10  
['UNUSED', '00000']  
loc 1,11  
['UNUSED', '00000']  
loc 1,12  
['NAN', '00000']  
loc 2,1  
['UNUSED', '00000']  
loc 2,2  
['UNUSED', '00000']  
loc 2,3  
['UNUSED', '00000']  
loc 2,4  
['UNUSED', '00000']  
loc 2,5  
['UNUSED', '00000']  
loc 2,6  
['UNUSED', '00000']  
loc 2,7  
['UNUSED', '00000']  
loc 2,8  
['UNUSED', '00000']  
loc 2,9  
['UNUSED', '00000']  
loc 2,10  
['UNUSED', '00000']  
loc 2,11  
['UNUSED', '00000']  
loc 2,12  
> TERMINAL
```

Example 2

In this image we display the results of successfully balancing a ship. The start and end coordinates are shown for each container that needs to be moved. This image is not representative of the final implementation of this system as we used these results as a foundation to later list every movement required, not just start to end.



Testing Process:

During the development of the container process implementation, we utilized print statements, alerts, and a handful of other techniques to debug and test the program and algorithm.

In this program, we utilized a few different languages that required different methods of error checking. For our javascript file scripts.js, we utilized alerts such as the one below to check methods and functionality.

```
function changeShipArray() {
  //alert("in changeShipArray");
  fetch('/Transfer-process-changes', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(selectedIDs)
  })
  .then(response => response.json())
  .then(data => {
    if (data.status == "success") {
      alert("array processed successfully: " + data.array);
    } else {
      alert("Null Array processed successfully: " + data.message);
    }
  })
  .catch(error => console.error('Error:', error));
}
```

For the majority of the program, since it ran on Python code with Flask to handle backend processes, we utilized print statements to debug.

```
def printPathContNested(self):
    for array in self.pathContNested:
        print("[", end = " ")
        for element in array:
            print(element.name[0:3] + str(array.index(element)) + element)
        print("] index = " + str(self.pathContNested.index(array)) + "\n")

#fix to create a different array of ship() containers for each step
def returnPathArray(self, path):
    # print(path)
    steps = []
```

This type of print statement also works to handle input received from .html files.

```
def test_serialization(uploaded_file=None):
    ship = Ship()

    if uploaded_file:
        # Load grid from the uploaded file
        print(f>Loading grid from uploaded file: {uploaded_file}", file=sys.stderr)
        ship.loadGrid(uploaded_file)

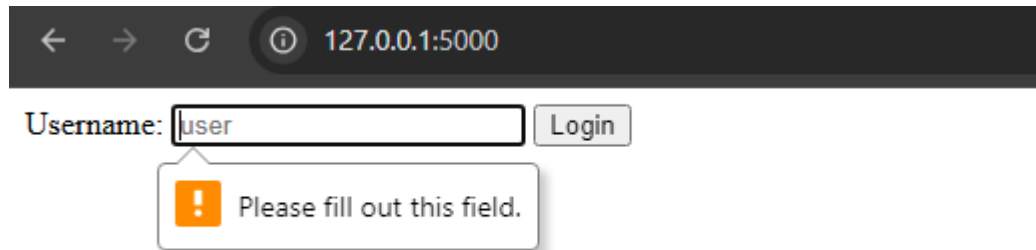
        # Save the state to the file
        with open(STATE_FILE, "wb") as f:
            pickle.dump(ship, f)
        print("Ship state saved after uploading the file.", file=sys.stderr)
    elif os.path.exists(STATE_FILE):
        # Restore the ship's state from the saved file
```

These print debug statement were viewed through alert statements that appeared on the browser and the terminal window.

Example Test Cases:

Some edge cases that may have caused the program to have an error included the following:

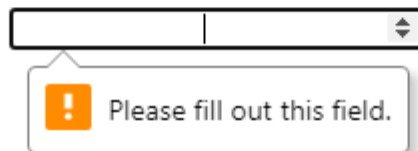
No inputs for login.



A screenshot of a web browser window with the address bar showing '127.0.0.1:5000'. Below the address bar, there is a login form. The label 'Username:' is followed by a text input field containing the text 'user'. To the right of the input field is a 'Login' button. A validation error message is displayed below the input field, consisting of an orange exclamation mark icon and the text 'Please fill out this field.'

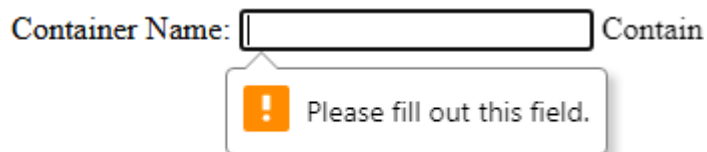
No inputs for containers coming on.

How many containers do you want to load?



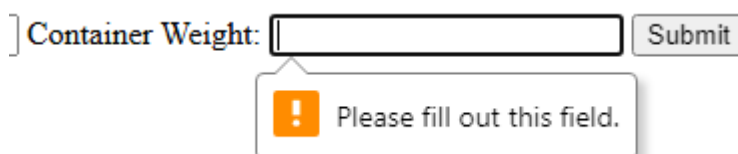
A screenshot of a form field. The field is a text input with a vertical line in the middle and a small downward arrow on the right side. Below the input field is a validation error message, consisting of an orange exclamation mark icon and the text 'Please fill out this field.'

No inputs for container name.



A screenshot of a form field. The label 'Container Name:' is followed by a text input field. To the right of the input field is a 'Contain' button. A validation error message is displayed below the input field, consisting of an orange exclamation mark icon and the text 'Please fill out this field.'

No inputs for container weight.



A screenshot of a form field. The label 'Container Weight:' is followed by a text input field. To the right of the input field is a 'Submit' button. A validation error message is displayed below the input field, consisting of an orange exclamation mark icon and the text 'Please fill out this field.'

Invalid inputs for container names (i.e.: NAN and UNUSED)

Container Name: C

Container name cannot be 'NAN' or 'UNUSED'.

Container Name: Container

Invalid inputs for weight such as negative numbers and numbers over 99999, were automatically floored and capped to 0 and 99999 respectively. Any decimal numbers are also rounded.

For the functionality of transfer, we tested different combinations of containers coming on and off. Some of the tests included processes that had no containers to be moved.