Dear Professor Keogh,
We are Team AI-Migos

Anand Mahadevan, amaha018@ucr.edu
Brandon Chea, tchea008@ucr.edu
David Tellez, dtell008@ucr.edu
Ricardo Villacana, rvill095@ucr.edu

We are happy to hand in our finished project. Please find below:

[A-B] example.pdf ;; symbolizes which pages the pdf spans on this pdf.

Ex. example.pdf spans from page A to page B.

1) [3 - 4] InitialElicitationQuestions.pdf, which contains evidence of our efforts to plan our elicitation of requirements from Mr. Keogh.

2) [5 - 51] Project Pitch for Mr.Keogh's Ship Container VERSION 1.pdf, which contains our original pitch to Mr. Keogh.

3) [52 - 98] Project Pitch for Mr.Keogh's Ship Container VERSION 2.pdf, which contains our modified pitch to Mr. Keogh. This document is 90% identical to TeamXXX_pitch_1.pdf, but contains corrections, amendments and deletions that came to our attention during the pitch.

4) [99 - 110] Requirement Specification.pdf, which shows how we handled requirement engineering for the project.

5) [111-115] Program Design.pdf, which shows how we designed the coding aspect of our program.

6) [116-125] System Design.pdf, which shows how we designed how our system operates.

7) [126-136] Unit and Integration Testing.pdf, which demonstrates the testing phase of our backend and front end as well as its integration.

8) [137-140] Acceptance Testing.pdf, which shows how our final product was tested.

9) [141] Maintenance Plan.pdf, which shows evidence of our efforts to set a plan for the maintenance of our software under the event of specific scenarios that may arise in the future.

10) [142-146] Training and Documentation.pdf, which shows evidence of our efforts to give instruction to operators of our software guidance in using it properly.

11) This URL, https://github.com/Bchea99/CS179M_AIMIGOS, points to the GitHub repository of our code.

12) This youtube video, https://www.youtube.com/watch?v=jZE7IWzeHM0 that shows our software solving one balance, one transfer test case, adding a comment to the log file, and showing the completed log file and outbound manifest file.

Jan 21, 2023

Anand Mahadevan = AM
Ricardo Villacana = RV
Brandon Chea = BC
David Tellez = DT

- When the crane moves over the ship or the buffer zone, is there a difference in time calculation when the crane enters from column 1 versus column 12? - AM
- I noticed that the log file example given to us involved only unloading and loading the ship. Are there specific requirements for the log file when balancing the ship? - AM
- What is the procedure when a container is to be loaded onto the ship but the ship is full to the brim with containers already? - AM
- Regarding the manifest, is there a possibility the manifest will have an invalid ship configuration, for example, an upside down ship? - AM
- When the operator enters the name of the container, either for onloading or offloading, are they required to enter the full name of it, or allowed to choose a container by typing in the first part of the name. For example, for a container such as "Walmart Store Bouncy Balls with extra bounce", can the operator concisely type in "Walmart" to choose that container? - AM
- If we need to move a container somewhere from the left to the right side of the ship, but the only way to do so is by moving the container outside of the 8x12 manifest grid to go up and over the other containers that are in the way, do those movements outside of the grid also count as 1 minute? - RV
- If a truck is unable to deliver a container to be loaded on the ship, does it have to be documented in the log file? If so, would we structure that documentation by indicating this certain container was not loaded by this truck because of this particular reason. Would we know the reason? - RV
- For offloading, it is indicated that if there's an issue with a receiving truck a special truck will be called to handle the container. Does "special" entail the truck will never have an issue and we can see it as a 100% guarantee the container can be handled correctly. - RV
- In the log file you want any issue with a container, such as a noticeable dent, to be documented by stating the container name and the issue. However, for containers with duplicate names should we indicate the location of the container in the manifest or leave it vague? - RV
- When we automatically update the manifest, are we required to create a backup of previous histories of the manifest? -BC
- Approximately how long would we like to continue support for this software in order to account for bug fixes or expanding functionality?-BC
- In the case of our software crashing, can we expect the operator to be able to continue working until the software comes back on?-BC
- If an action is undone by the back button, would you like to have the same log repeated for the action it is doing, or should redundant repetition be avoided in the logs? -DT
- Would you like a schematic of the ship printed to a .txt file after a ship has been balanced? -DT

Jan 21, 2023

- Is it necessary to have the program pause at any given time such that the current employee can write their own log message, or can there be an assigned time to write a log (e.g. at the end of an animation) -DT

Project Pitch for Mr.Keogh's Ship Container Problem

- Ai-Migos Software
- February 7th 2022

Meet the Team

Brandon Chea
Computer Science

Anand Mahadevan
Computer Science

David Tellez
Computer Science

Ricardo Villacana
Computer Science

Presentation Outline

1. Problem Overview

Outlining the problem at hand

Basic Understanding

Here is our understanding of the problem:

- You own a single port in Long Beach, California, U.S., and "get paid per number of containers moved, so the faster [you] work, the better". [25]
- Your employees have "at least high school educations", "speak/read basic English", and work on "barebones" PCs with "fast ethernet". [27]
- All ships that come through your port have one bay, and the maximum amount of containers in that bay are within 8 rows and 12 columns. [26]
- You desire two different tasks to be completed for you through software:
  - **1) Loading and Unloading process of containers to trucks** [9]
  - **2) Balancing the ship through a specific legal definition** [23]

Basic Understanding (2)

- The **manifest** is defined to contain the 1) the name of each container, 2) the location of said container, and 3) the weight of said container within the bay of the ship. [15]
  - This manifest of the ship to be inputted by the operator into the software before either task of loading/unloading or balancing of the ship can occur. [27]
- The **transfer list** describes the set of container moves to be performed by the operator of the crane; it is the duty of the operator to relay those instructions as input to our software. It is not the duty of the software to read the transfer list. [27]

Basic Understanding (3)

- **Loading/Unloading**
  - Given an operator providing a container and location to place that container, produce software to provide an optimal ordering of container movements to complete that task in regards to time. [9]
    - There is to be a "simple animation" in regards to the movement of each container and an estimate of how much time all these movements will take in total. [9]
    - After the operator confirms that container has been moved to the inputted location, the software will edit the given manifest to reflect said results. [9]

Basic Understanding (4)

- **Balancing**
  - Given an operator stating that a balancing of the ship is to be done, produce software that will provide an optimal list of container movements to complete that task in regards to time. [23]
    - The balancing of a ship is defined as "if the total mass of the port side, and the total mass of the starboard side are within ten percent of each other". [24]
    - There is to be a "simple animation" in regards to the movement of each container and an estimate of how much time all these movements will take in total. [9]

Basic Understanding (5)

- **Log File**
  - You require a log file for legal purposes to contain events such as the signing in and out of employees, the movement of containers, and the opening and closing of the manifest file all with time stamps attached to each event. [28]
    - The log file is to be in .txt format, or in other words, "plain text format", readable on notepad. [29]
    - The time representation used of the log file is to be in military time, A.K.A. 24 hour clock. For clarity's sake, 23:00 is to represent 11 P.M. [30]
    - Your operators will be able to write their own typed-out comments to the log file. [28]

Assumptions

- All operators are capable of reading/writing in English
- All operators do not have any severe vision impairments that will negatively affect their ability to operate the software
- Operators will never go against the software or operate on their own accord
- The crane has access to electricity and possesses outlets for a computer to access
- The computer that the software will run on will have the appropriate peripherals for work such as:
    - Mouse and Keyboard
    - Monitor
- The company computer will run on the latest windows operating system

* All bullet points cite to [27]

Stakeholders

- Port of Long Beach
- Mr.Keogh
    - Mr.Keogh’s Crane Operators
- Ship Captain
    - Ship Crewmates
- Truck Drivers

* All bullet points cite to [1]

Stakeholders (2)

- Companies that retain ownership of the containers
- Customers of products stored within the containers
- Department of Homeland Security
- Insurance Companies

* All bullet points cite to [1]

2.
Solution

How our solution works for you

Input

- **Manifest File** - The file sent from the ship that represents the layout of it's bay and the position of all containers onboard(.txt file format)[15]
  - Sent by the incoming ship before it arrives and the user never manually edits the file[16]
  - Total of 96 lines
    - Represents an 8 x 12 grid as the largest possible dimension for an incoming ship[15]
  - Layout: Coordinates, Container Weight, Slot Description[15]
    - Ex: [x,y], {Weight}, Slot Description
      - Slot Descr: NAN, UNUSED, or CONTAINER LABEL

Input

- **Employee Information** - Information required by an employee before using the software. [17]
    - Full Name Layout: FIRSTNAME+[SPACE]+LASTNAME[17]
    - String with no numeric numbers[18]
    - No case sensitivity and no character limit
    - Typed in a textbox and pushed to log file[19]

- **Employee Comments** - Employee can write a comment for certain situations regarding a container.[12]
    - No character limit
    - Available only for log file[12]
    - Typed in a textbox and pushed to log file[19]

Input

- **Operation Keys** - Specific keys assigned to initiate specific tasks
  - Only a single press of the assigned key will be required
  - Example Key and Task Pairs:
    - Press S to switch users
    - Press A to proceed to next animation
    - Press C to push a comment to log file
- **System Clock** - An internal pulse, representative of a second, used by the computer to keep the correct time.[20]
  - Required for specific utilizations of the date and time in the software[11]
  - As a preventative measure, the software will change your computer settings to ensure only an administrator can manipulate the system clock.

Input

- **Entries For Offloading** - The process required by an employee to indicate the crates that need to be offloaded
  - Employee selects offloading operation
  - Grid displays with the containers and company names[21]
  - All desired containers are selected, as designated by the transfer file[16]
  - Employee clicks the DONE option to indicate they are finished selecting

Input

- **Entries for Loading** - The process required by an employee to indicate the crates that need to be loaded
    - Employee selects loading operation
    - The container label is entered into a textbox, then press ENTER
        - A string of up to 256 characters[22]
        - Case sensitive
    - The container weight is entered into a textbox, then press ENTER
    - Employee can continue to add more entries
    - Employee clicks the DONE option to indicate they are finished

Output

- **Program start** – There will be various prompts at the very beginning when the program is booted up.
  - A prompt to input a name to sign is as a user
  - A prompt that asks if the user would like to start a new log file.
    - If YES, it will create a default log file named "KeoghLongBeach.txt". It will then prompt the user for what year to append to the default log file.
    - If NO, there will be no new log file created. [14]
  - If the program has been interrupted in the middle of execution, there will be a prompt that asks if the application should resume where it left off.

Output

- **Program start [cont]** – There will be various prompts at the very beginning when the program is booted up.
  - A prompt asking the user to upload the Manifests document.
  - A prompt asking the user whether they would like to balance the ship, or start a transfer.
    - If the user selected to start a transfer, it will prompt the user to select between loading and unloading.

Output

- **Order of operations list** - For the transfer service, this is a text list of operations that the employee will perform to result in the fastest transfer. For the balance service, it will be a list showing how to balance the ship in the fastest amount of time [9]

Output

- **Time Estimation** – The estimated time until completion will be display after an employee has inputted the Manifest file. [9]
- **Visual Representation of the Manifest**– This will be displayed as a grid displaying the names of each of the companies that the containers belong to. From this grid, operators may choose which slot representing a container to unload. [10]
- **User Comment Prompt**– Upon pressing a designated key, a prompt will appear to allow the user to input a message. [12]
  - The employee will be able to add comments at any time the program is running. These comments will be outputted through the log file. [14]

Output

- **Animation** - This will be an animation showing the best order to move the containers. [9]
  - There will be one animation for each atomic transfer.
  - This will be an endless loop with one change to it per second for each atomic transfer. [7]
  - To proceed to the next animation, an employee can press a designated key.
  - When every animation has been displayed, the animation will output a success message [8]
  - The animation will be displayed for both balancing and transfers.

Output

- **Changing Users** - This will be an output prompting an employee to write their full name. [5]
  - At any point of the program, the current employee can press a designated key to trigger this output.
  - The output will allow an employee to write out their full name to switch users.
  - Only log in is recorded in the output logs, signing out is implicit. [6]

Output

- **Altered Manifest File** - This will be outputted after finishing a cycle. It will show the current state of the manifest file after certain containers have been moved. [9]
  - The name of this modified version of the manifest file will be MANIFEST NAME + OUTBOUND, for example TitanicJonesII.txt will have a modified version named "TitanicJonesIIOUTBOUND.txt [13]
- **Reminder Popup** - At the same time that the Outbound Manifest File is written, a reminder pop-up to the operator to send the file will be displayed. [9]

Output

- **Log File**- This will be a file that will log certain actions in the program. This is needed for legal purposes [11] With the following properties:
  - The "certain actions" being referred to are
    - 1. Time and personnel of a shift change
    - 2. Opening or closing a manifest
    - 3. Every atomic move of a container
    - 4. Any ad-hoc notes that the user wants to make [7]
  - It will be written in plain text to be opened in notepad [11]
  - The log file will follow the naming convention KeoghLongBeach[Year].txt where [Year] will be the current year [13]

Scenario 1 – Loading/Offloading

- This scenario serves as a basic use case of the software for loading/offloading including an example where an operator uses a comment.
- It is 4PM and Dan is clocking in for work.
- Dan signs into the system by typing his name
- Dan then receives a transfer list from the head office
- Dan uploads the Manifest file previously sent by the ship that is arriving at 4:20 pm.

Scenario 1 – Loading/Offloading

- Dan analyzes the transfer list, detecting which crates need to be loaded and offloaded.
- Dan then selects the offloading option displayed by the system
- An 8 x 12 grid displays with the containers and company names
- Dan clicks on the containers that need to be offloaded
- Then Dan selects the loading option displayed by the system
- Dan inserts the necessary data for loading the crates

Scenario 1 – Loading/Offloading

- An ordered operations list of the time optimal way to load and offload the containers will be displayed.
- Then the system will display the grid animation of the first optimal step Dan needs to execute, along with the estimated time to complete the task.
- Dan then presses the A key to indicate he has executed the first step and the animation for the next step is displayed.
- The estimated time to complete the task will be updated for every step.

Scenario 1 – Loading/Offloading

- Dan observes a significant dent when moving a container.
- Dan then presses the C key to enter a comment and states the issue that he has detected, along with the label of the container.
- The comment is then written to the log file by the system.
- Dan continues to follow the instruction of the system, loading and offloading the containers shown to him by the animation.
- Dan finishes his shift when Tommy Smith arrives to sign in for his shift at 12 am.

Scenario 2 – Unexpected Power Outage

- Adam Vinatieri is an employee of yours.
- He is operating the crane utilizing our software.
- There is a power outage that occurs during the moving process.
- There are two different cases in relation to the recovery of the software depending on the time that the power outage occurs.

Scenario 2 — Unexpected Power Outage – Case 1 (2)

- If the power goes out while the software is calculating the optimal move for a container entered by him, he will need to re-enter the operation to be performed on it again after rebooting the system.

Scenario 2 — Unexpected Power Outage – Case 2 (3)

- If the power goes out **after** the software is done calculating the optimal move for a container entered by him, he will be presented with a screen to either state the move has been completed **OR** to enter cancel and enter a different move.

Scenario 3 – Balancing

- This scenario serves as a basic use case for balancing a ship where nothing goes wrong.
- It is 8 AM and Sean clocks in for work by signing his name
- Sean determines that the ship needs to be balanced and communicates this to the software.
- The software displays an animation for the first step alongside an estimated time to complete the task.
- Sean moves through each step while pressing the A key to indicate that the step has been completed.

Scenario 3 – Balancing

- After all steps have been completed, Sean will be prompted with the option of downloading the outbound manifest.
- Sean downloads the manifest and gets a notification that the manifest has been successfully downloaded alongside another notification indicating that it must still be sent to the ship’s captain.
- Sean emails the manifest to the outbound ship’s captain and receives a notification indicating that the manifest has been successfully sent.
- Sean is then able to receive another manifest/task.

3. Post Deployment

What comes next

Maintenance Plan

While our team is unable to predict future events, we acknowledge the following scenarios where our software will need to be updated.

- In the event that the size of a container increases or decreases by any amount, our software representation of the ship bay will need to be updated in order to maintain the functions of onloading, offloading, and balancing.

Maintenance Plan (2)

- In the event that the size of the buffer zone increases or decreases, our software representation of the buffer zone will need to be updated in order to maintain the functions of onloading, offloading, and balancing.
- In the event that the size of the largest ship bay possible increases from 8 rows and/or 12 columns, our software representation of the ship bay will need to be updated in order to maintain the functions of onloading, offloading, and balancing.

Training and Documentation

- The software will have full documentation including a 1-5 page instruction manual (both physical and digital)
- All employees will undergo a training session in order to familiarize them with the software
    - This training seminar will be done virtually in a 1 hour session

Compliance with Regulation

EPA* Regulation[2]

Complies with port and vessel legislature

Port Crane Security and Inspect Act of 2022[3]

Obeys basic security concerns

Ocean Shipping Reform Act of 2022[4]

Our software does not have an effect on shipping fees induced by container relocation

*EPA is the U.S. Environmental Protection Agency

Acceptance Testing

- We have the final deliverable on or before Thursday, March 16th 2023.
- We propose the following tests:
  - Two weeks before the acceptance test, you will send us up to ten scenarios and we will test any four of them you choose, live.
  - With zero notice, you will provide up to three scenarios, and we will test them live.

Acceptance Testing (continued)

- The following are metrics for success
  - For 100 scenarios, SIFT should only be applied 1 time maximum. [31]
  - The system should be able to be resumed within 30 seconds of running after a power cut
  - The system can be restored at any given point of the program with no memory loss
  - The algorithm provided will be optimal or near optimal. In particular, if another software package, or a human using "paper and pencil" could provide an alternative solution that is more than 5% cheaper, the entire project will be deemed a failure. This applies to both transferring and balancing.

Acceptance Testing (continued)

- The following are metrics for success
  - For 100 different manifest files, if the user requests for balancing, the outbound file should accurately display a balanced ship every time.
  - For 100 different manifest files, if the user requests for transfers, the outbound file should accurately display the resulting ship configuration every time.
  - For 100 different manifest files, if the user requests for transfers, there will not be a container left in the buffer zone.
  - For 100 manifest files containing varying amounts of duplications of container names, if a user request for a transfer, the program will be able to output 1. An optimal result and 2. An accurate Outbound file

Contract

- This serves as a Bilateral Agreement between Mr.Keogh and Ai-Migos Software LLC.
- Our team agrees to create software that provides the optimal operator moves regarding:
  - Loading and Unloading Containers
  - Balancing ships
- The final deliverable will be ready on or before Thursday March 16 at 11:59 PM PST.
  - Failure to deliver on time will result in a deduction of the overall fee charged by our company.
- Our team agrees only to the production of features that have been outlined today and will not accept requests for the addition of other features made at a later date.
- We may desire up to 3 hours of your time (or a qualified proxy) in order to answer any additional clarifications.
- Our fee for this software after acceptance testing will be: **$200,000**

Signed (for Ai-Migos Software) x AM, BC, DT, RV Date 2/7/2023

Signed (for Mr. Keogh) x Mr. John Keogh Date 2/7/2023

References

1. Questionswithanswers2.pptx -- Slide 5
2. https://www.epa.gov/vessels-marinas-and-ports
3. https://www.congress.gov/bill/117th-congress/house-bill/6487/text?r=2&s=1
4. https://www.cantwell.senate.gov/imo/media/doc/Ocean%20Shiping%20Reform%20Act%20of%202022%20One-Pager.pdf
5. Problem Overview by Mr. Keogh -- Slide 34
6. Questionswithanswers2.pptx -- Slide 11
7. Questionswithanswers.pptx -- Slide 11
8. Questionswithanswers.pptx -- Slide 15
9. Problem Overview by Mr. Keogh -- Slide 13
10. Problem Overview by Mr. Keogh -- Slide 16
11. Problem Overview by Mr. Keogh -- Slide 33
12. Problem Overview by Mr. Keogh -- Slide 35
13. Questionswithanswers.pptx -- Slide 32
14. Questionswithanswers2.pptx -- Slide 3
15. Problem Overview by Mr. Keogh -- Slide 23
16. Problem Overview by Mr. Keogh -- Slide 12
17. Questionswithanswers.pptx -- Slide 18
18. https://www.birthcertificatecopy.com/articles/naming-laws-in-each-u-s-state/
19. Questionswithanswers.pptx -- Slide 4
20. https://www.computerhope.com/jargon/c/clock.htm
21. Questionswithanswers.pptx -- Slide 40

References (2)

22. Questionswithanswers.pptx -- Slide 25
23. Problem Overview by Mr. Keogh -- Slide 15
24. Problem Overview by Mr. Keogh -- Slide 25
25. Problem Overview by Mr. Keogh -- Slide 7
26. Problem Overview by Mr. Keogh -- Slide 8
27. Problem Overview by Mr. Keogh -- Slide 14
27. Problem Overview by Mr. Keogh -- Slide 17 and Slide 18
28. Problem Overview by Mr. Keogh -- Slide 35
29. Problem Overview by Mr. Keogh -- Slide 33
30. Elicitation meeting 1/27/2023 – Question asked and answered
31. Problem Overview by Mr. Keogh -- Slide 27

THANKS!

Any questions?

You may contact each team member individually at:

- Brandon Chea - tchea008@ucr.edu
- Anand Mahadevan - amaha018@ucr.edu
- David Tellez - dtell008@ucr.edu
- Ricardo Villacana - rvill095@ucr.edu

Project Pitch for Mr.Keogh's Ship Container Problem

- Ai-Migos Software
- February 7th 2022

Meet the Team

Brandon Chea
Computer Science

Anand Mahadevan
Computer Science

David Tellez
Computer Science

Ricardo Villacana
Computer Science

Presentation Outline

1.
Problem Overview

Outlining the problem at hand

Basic Understanding

Here is our understanding of the problem:

- You own a single port in Long Beach, California, U.S., and "get paid per number of containers moved, so the faster [you] work, the better". [25]
- Your employees have "at least high school educations", "speak/read basic English", and work on "barebones" PCs with "fast ethernet". [27]
- All ships that come through your port have one bay, and the maximum amount of containers in that bay are within 8 rows and 12 columns. [26]
- You desire two different tasks to be completed for you through software:
  - **1) Loading and Unloading process of containers to trucks** [9]
  - **2) Balancing the ship through a specific legal definition** [23]

Basic Understanding (2)

- The **manifest** is defined to contain the 1) the name of each container, 2) the location of said container, and 3) the weight of said container within the bay of the ship. [15]
  - This manifest of the ship to be inputted by the operator into the software before either task of loading/unloading or balancing of the ship can occur. [27]
- The **transfer list** describes the set of container moves to be performed by the operator of the crane; it is the duty of the operator to relay those instructions as input to our software. It is not the duty of the software to read the transfer list. [27]

Basic Understanding (3)

- **Loading/Unloading**
  - Given an operator providing a container and location to place that container, produce software to provide an optimal ordering of container movements to complete that task in regards to time. [9]
    - There is to be a "simple animation" in regards to the movement of each container and an estimate of how much time all these movements will take in total. [9]
    - After the operator confirms that container has been moved to the inputted location, the software will edit the given manifest to reflect said results. [9]

Basic Understanding (4)

- **Balancing**
  - Given an operator stating that a balancing of the ship is to be done, produce software that will provide an optimal list of container movements to complete that task in regards to time. [23]
    - The balancing of a ship is defined as "if the total mass of the port side, and the total mass of the starboard side are within ten percent" weight of the heavier side. For example, if the heavier side of the ship is 100 kg, the lighter side must be 90 kg or heavier in order for the ship to be legally considered balanced. [24]
    - There is to be a "simple animation" in regards to the movement of each container and an estimate of how much time all these movements will take in total. [9]

Basic Understanding (5)

- **Log File**
  - You require a log file for legal purposes to contain events such as the signing in and out of employees, the movement of containers, and the opening and closing of the manifest file all with time stamps attached to each event. [28]
    - The log file is to be in .txt format, or in other words, "plain text format", readable on notepad. [29]
    - The time representation used of the log file is to be in military time, A.K.A. 24 hour clock. For clarity's sake, 23:00 is to represent 11 P.M. [30]
      - The log file will be in Pacific Standard Time as the port is in Long Beach, CA
    - Your operators will be able to write their own typed-out comments to the log file. [28]

Assumptions

- All operators are capable of reading/writing in English
- All operators do not have any severe vision impairments that will negatively affect their ability to operate the software
- Operators will never go against the software or operate on their own accord
- The crane has access to electricity and possesses outlets for a computer to access
- The computer that the software will run on will have the appropriate peripherals for work such as:
  - Mouse and Keyboard
  - Monitor
- The company computer will run on the latest windows operating system

* All bullet points cite to [27]

Stakeholders

- Port of Long Beach
- Mr.Keogh
  - Mr.Keogh’s Crane Operators
- Ship Captain
  - Ship Crewmates
- Truck Drivers

* All bullet points cite to [1]

Stakeholders (2)

- Companies that retain ownership of the containers
- Customers of products stored within the containers
- Department of Homeland Security
- Insurance Companies

* All bullet points cite to [1]

2.
Solution

How our solution works for you

Input

- **Manifest File** - The file sent from the ship that represents the layout of it's bay and the position of all containers onboard(.txt file format)[15]
  - Sent by the incoming ship before it arrives and the user never manually edits the file[16]
  - Total of 96 lines
    - Represents an 8 x 12 grid as the largest possible dimension for an incoming ship[15]
  - Layout: Coordinates, Container Weight, Slot Description[15]
    - Ex: [x,y], {Weight}, Slot Description
      - Slot Descr: NAN, UNUSED, or CONTAINER LABEL

Input

- **Employee Information** - Information required by an employee before using the software. [17]
  - Full Name Layout: FIRSTNAME+[SPACE]+LASTNAME[17]
  - String with no numeric numbers[18]
  - No case sensitivity and no character limit
  - Typed in a textbox and pushed to log file[19]

- **Employee Comments** - Employee can write a comment for certain situations regarding a container.[12]
  - No character limit
  - Available only for log file[12]
  - Typed in a textbox and pushed to log file[19]

Input

- **Operation Keys** - Specific keys assigned to initiate specific tasks
  - Only a single press of the assigned key will be required
  - Example Key and Task Pairs:
    - Press S to switch users
    - Press A to proceed to next animation
    - Press C to push a comment to log file
    -
- **System Clock** - An internal pulse, representative of a second, used by the computer to keep the correct time.[20]
  - Required for specific utilizations of the date and time in the software[11]
  - As a preventative measure, the software will change your computer settings to ensure only an administrator can manipulate the system clock.

Input

- **Entries For Offloading** - The process required by an employee to indicate the crates that need to be offloaded
  - Employee selects offloading operation
  - Grid displays with the containers and company names[21]
  - All desired containers are selected, as designated by the transfer file[16]
  - Employee clicks the DONE option to indicate they are finished selecting

Input

- **Entries for Loading** - The process required by an employee to indicate the crates that need to be loaded
  - Employee selects loading operation
  - The container label is entered into a textbox, then press ENTER
    - A string of up to 256 characters[22]
    - Case sensitive
  - The container weight is entered into a textbox, then press ENTER
  - Employee can continue to add more entries
  - Employee clicks the DONE option to indicate they are finished

Output

- **Program start** - There will be various prompts at the very beginning when the program is booted up.
  - A prompt to input a name to sign is as a user
  - A prompt that asks if the user would like to start a new log file.
    - If YES, it will create a default log file named "KeoghLongBeach.txt". It will then prompt the user for what year to append to the default log file.
    - If NO, there will be no new log file created. [14]
  - If the program has been interrupted in the middle of execution, there will be a prompt that asks if the application will resume where it left off.

Output

- **Program start [cont]** – There will be various prompts at the very beginning when the program is booted up.
  - A prompt asking the user to upload the Manifests document.
  - A prompt asking the user whether they would like to balance the ship, or start a transfer.
    - If the user selected to start a transfer, it will prompt the user to select between loading and unloading.

Output

- **Order of operations list** - For the transfer service, this is a text list of operations that the employee will perform to result in the fastest transfer. For the balance service, it will be a list showing how to balance the ship in the fastest amount of time [9]
- **Name of the Manifest/Ship**- The name of the manifest and ship will be displayed at all times. This is to ensure that the employee knows they are working on the correct ship/manifest file.

Output

- **Time Estimation** – The estimated time until completion will be display after an employee has inputted the Manifest file. [9]
- **Visual Representation of the Manifest**– This will be displayed as a grid displaying the names of each of the companies that the containers belong to. From this grid, operators may choose which slot representing a container to unload. [10]
- **User Comment Prompt**– Upon pressing a designated key, a prompt will appear to allow the user to input a message. [12]
  - The employee will be able to add comments at any time the program is running. These comments will be outputted through the log file. [14]

Output

- **Animation** – This will be an animation showing the best order to move the containers. [9]
    - There will be one animation for each atomic transfer.
    - This will be an endless loop with one change to it per second for each atomic transfer. [7]
    - To proceed to the next animation, an employee can press a designated key.
    - When every animation has been displayed, the animation will output a success message [8]
    - The animation will be displayed for both balancing and transfers.

Output

- **Changing Users** - This will be an output prompting an employee to write their full name. [5]
  - At any point of the program, the current employee can press a designated key to trigger this output.
  - The output will allow an employee to write out their full name to switch users.
  - Only log in is recorded in the output logs, signing out is implicit. [6]

Output

- **Altered Manifest File** - This will be outputted after finishing a cycle. It will show the current state of the manifest file after certain containers have been moved. [9]
  - The name of this modified version of the manifest file will be MANIFEST NAME + OUTBOUND, for example TitanicJonesII.txt will have a modified version named "TitanicJonesIIOUTBOUND.txt [13]
- **Reminder Popup** - At the same time that the Outbound Manifest File is written, a reminder pop-up to the operator to send the file will be displayed. The pop-up will only be closed when a user activates the close operation. [9]

Output

- **Log File**- This will be a file that will log certain actions in the program. This is needed for legal purposes [11] With the following properties:
  - The “certain actions” being referred to are
    - 1. Time and personnel of a shift change
    - 2. Opening or closing a manifest
    - 3. Every atomic move of a container
    - 4. Any ad-hoc notes that the user wants to make [7]
  - It will be written in plain text to be opened in notepad [11]
  - The log file will follow the naming convention KeoghLongBeach[Year].txt where [Year] will be the current year [13]

Scenario 1 – Loading/Offloading

- This scenario serves as a basic use case of the software for loading/offloading including an example where an operator uses a comment.
- It is 4PM and Dan is clocking in for work.
- Dan signs into the system by typing his name
- Dan then receives a transfer list from the head office
- Dan uploads the Manifest file previously sent by the ship that is arriving at 4:20 pm.

Scenario 1 – Loading/Offloading

- Dan analyzes the transfer list, detecting which crates need to be loaded and offloaded.
- Dan then selects the offloading option displayed by the system
- An 8 x 12 grid displays with the containers and company names
- Dan clicks on the containers that need to be offloaded
- Then Dan selects the loading option displayed by the system
- Dan inserts the necessary data for loading the container.

Scenario 1 – Loading/Offloading

- An ordered operations list of the time optimal way to load and offload the containers will be displayed.
- Then the system will display the grid animation of the first optimal step Dan needs to execute, along with the estimated time to complete the task.
- Dan then presses the A key to indicate he has executed the first step and the animation for the next step is displayed.
- The estimated time to complete the task will be updated for every step.

Scenario 1 – Loading/Offloading

- Dan observes a significant dent when moving a container.
- Dan then presses the C key to enter a comment and states the issue that he has detected, along with the label of the container.
- The comment is then written to the log file by the system.
- Dan continues to follow the instruction of the system, loading and offloading the containers shown to him by the animation.
- Dan finishes his shift when Tommy Smith arrives to sign in for his shift at 12 am.

Scenario 2 — Unexpected Power Outage

- Adam Vinatieri is an employee of yours.
- He is operating the crane utilizing our software.
- There is a power outage that occurs during the moving process.
- There are two different cases in relation to the recovery of the software depending on the time that the power outage occurs.

Scenario 2 — Unexpected Power Outage – Case 1 (2)

- If the power goes out while the software is calculating the optimal move for a container entered by him, he will need to re-enter the operation to be performed on it again after rebooting the system.

Scenario 2 — Unexpected Power Outage – Case 2 (3)

- If the power goes out **after** the software is done calculating the optimal move for a container entered by him, he will be presented with a screen to either state the move has been completed **OR** to enter cancel and enter a different move.

Scenario 3 – Balancing

- This scenario serves as a basic use case for balancing a ship where nothing goes wrong.
- It is 8 AM and Sean clocks in for work by signing his name
- Sean determines that the ship needs to be balanced and communicates this to the software.
- The software displays an animation for the first step alongside an estimated time to complete the task.
- Sean moves through each step while pressing the A key to indicate that the step has been completed.

Scenario 3 – Balancing

- After all steps have been completed, Sean will be prompted with the option of downloading the outbound manifest.
- Sean downloads the manifest and gets a notification that the manifest has been successfully downloaded alongside another notification indicating that it must still be sent to the ship's captain.
- Sean emails the manifest to the outbound ship's captain and receives a notification indicating that the manifest has been successfully sent.
- Sean is then able to receive another manifest/task.

3. Post Deployment

What comes next

Maintenance Plan

While our team is unable to predict future events, we acknowledge the following scenarios where our software will need to be updated.

- In the event that the size of a container increases or decreases by any amount, our software representation of the ship bay will need to be updated in order to maintain the functions of onloading, offloading, and balancing.

Maintenance Plan (2)

- In the event that the size of the buffer zone increases or decreases, our software representation of the buffer zone will need to be updated in order to maintain the functions of onloading, offloading, and balancing.
- In the event that the size of the largest ship bay possible increases from 8 rows and/or 12 columns, our software representation of the ship bay will need to be updated in order to maintain the functions of onloading, offloading, and balancing.

Training and Documentation

- The software will have full documentation including a 1-5 page instruction manual (both physical and digital)
- All employees will undergo a training session in order to familiarize them with the software
  - This training seminar will be done virtually in a 1 hour session

Compliance with Regulation

EPA* Regulation[2]

Complies with port and vessel legislature

Port Crane Security and Inspect Act of 2022[3]

Obeys basic security concerns

Ocean Shipping Reform Act of 2022[4]

Our software does not have an effect on shipping fees induced by container relocation

*EPA is the U.S. Environmental Protection Agency

Acceptance Testing

- We have the final deliverable on or before Thursday, March 16th 2023.
- We propose the following tests:
  - Two weeks before the acceptance test, you will send us up to ten scenarios and we will test any four of them you choose, live.
  - With zero notice, you will provide up to three scenarios, and we will test them live.

Acceptance Testing (continued)

- The following are metrics for success
  - For 100 scenarios, SIFT will only be applied 1 time maximum. [31]
  - The system will be able to be resumed within 30 seconds of running after a power cut
  - The system can be restored at any given point of the program with no memory loss
  - The algorithm provided will be optimal or near optimal. In particular, if another software package, or a human using "paper and pencil" could provide an alternative solution that is more than 5% cheaper, the entire project will be deemed a failure. This applies to both transferring and balancing.

Acceptance Testing (continued)

- The following are metrics for success
  - For 100 different manifest files, if the user requests for balancing, the outbound file will accurately display a balanced ship every time.
  - For 100 different manifest files, if the user requests for transfers, the outbound file will accurately display the resulting ship configuration every time.
  - For 100 different manifest files, if the user requests for transfers, there will not be a container left in the buffer zone.
  - For 100 manifest files containing varying amounts of duplications of container names, if a user request for a transfer, the program will be able to output 1. An optimal result and 2. An accurate Outbound file

Contract

- This serves as a Bilateral Agreement between Mr.Keogh and Ai-Migos Software LLC.
- Our team agrees to create software that provides the optimal operator moves regarding:
  - Loading and Unloading Containers
  - Balancing ships
- The final deliverable will be ready on or before Thursday March 16 at 11:59 PM PST.
  - Failure to deliver on time will result in a deduction of the overall fee charged by our company.
- Our team agrees only to the production of features that have been outlined today and will not accept requests for the addition of other features made at a later date.
- We may desire up to 3 hours of your time (or a qualified proxy) in order to answer any additional clarifications.
- Our fee for this software after acceptance testing will be: **$200,000**

Signed (for Ai-Migos Software) x AM, BC, DT, RV Date 2/7/2023

Signed (for Mr. Keogh) x Mr. John Keogh Date 2/7/2023

References

1. Questionswithanswers2.pptx -- Slide 5
2. https://www.epa.gov/vessels-marinas-and-ports
3. https://www.congress.gov/bill/117th-congress/house-bill/6487/text?r=2&s=1
4. https://www.cantwell.senate.gov/imo/media/doc/Ocean%20Shiping%20Reform%20Act%20of%202022%20One-Pager.pdf
5. Problem Overview by Mr. Keogh -- Slide 34
6. Questionswithanswers2.pptx -- Slide 11
7. Questionswithanswers.pptx -- Slide 11
8. Questionswithanswers.pptx -- Slide 15
9. Problem Overview by Mr. Keogh -- Slide 13
10. Problem Overview by Mr. Keogh -- Slide 16
11. Problem Overview by Mr. Keogh -- Slide 33
12. Problem Overview by Mr. Keogh -- Slide 35
13. Questionswithanswers.pptx -- Slide 32
14. Questionswithanswers2.pptx -- Slide 3
15. Problem Overview by Mr. Keogh -- Slide 23
16. Problem Overview by Mr. Keogh -- Slide 12
17. Questionswithanswers.pptx -- Slide 18
18. https://www.birthcertificatecopy.com/articles/naming-laws-in-each-u-s-state/
19. Questionswithanswers.pptx -- Slide 4
20. https://www.computerhope.com/jargon/c/clock.htm
21. Questionswithanswers.pptx -- Slide 40

References (2)

22. Questionswithanswers.pptx -- Slide 25
23. Problem Overview by Mr. Keogh -- Slide 15
24. Problem Overview by Mr. Keogh -- Slide 25
25. Problem Overview by Mr. Keogh -- Slide 7
26. Problem Overview by Mr. Keogh -- Slide 8
27. Problem Overview by Mr. Keogh -- Slide 14
27. Problem Overview by Mr. Keogh -- Slide 17 and Slide 18
28. Problem Overview by Mr. Keogh -- Slide 35
29. Problem Overview by Mr. Keogh -- Slide 33
30. Elicitation meeting 1/27/2023 – Question asked and answered
31. Problem Overview by Mr. Keogh -- Slide 27

THANKS!

Any questions?

You may contact each team member individually at:

- Brandon Chea - tchea008@ucr.edu
- Anand Mahadevan - amaha018@ucr.edu
- David Tellez - dtell008@ucr.edu
- Ricardo Villacana - rvill095@ucr.edu

Software Requirements Specification

for

Shipping Container Software Solution

Ai-Migos Software

March 26, 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Brandon Chea	2/26/23	Setting up template	0.1
Brandon Chea	3/25/23	Finalizing Spec Sheet details	1

1. Introduction

1.1 Purpose

The purpose of this document is to catalog the requirements of the shipping container software solution. The software system is intended to handle the task of loading and unloading shipping containers on cargo ships as well as balancing ships.

1.2 Document Conventions

- Terms of interest will be bolded and italicized. I.e. ***shipping container***
- The development team and designer of this document have not received express permission to use the personal information of any parties. For this reason, anyone referenced in this document is done so by their occupation or relationship with AI-Migos Software LLC. I.e. *client, crane operator, ship captain, etc.*

1.3 Intended Audience and Reading Suggestions

This document is designed to be read and understood by the development team behind this software as well as the client.

1.4 Product Scope

This software solution serves to expedite the task of loading and unloading shipping containers on cargo ships. Additionally, this software is capable of balancing ships in accordance with maritime balancing laws.

1.5 References

- This document references multiple elicitation sessions conducted within the month of February, 2023 between the client and AI-Migos Software LLC.

- This document also references several problem overview presentations received from the client within February, 2023
- This document is built with the IEEE official Software Requirements Specification Sheet template.

2. Overall Description

2.1 Product Perspective

This software automates the process of selecting and moving containers to and from a cargo ship. Additionally, it balances the containers on ships in accordance with maritime law. This software is designed to be used by crane operators employed by the client in order to streamline the process of container transfers and balancing.

2.2 Product Functions

The software product accomplishes two primary tasks, container selection for transfer and balancing ships. These functions are elaborated on further in Section 4.

- ***Container Transfer*** - the task of automating container transfer is done with two primary functions
  - ***Loading*** - The loading functionality allows for a shipping container to be moved from the dock onto the cargo ship
  - ***Unloading*** - The unloading functionality allows for a shipping container to be taken off of the ship by relocating the smallest amount of containers as possible
- ***Ship Balancing*** - In the case of ships requiring balancing at the dock, the software solution is capable of automatically selecting the optimal containers to move in accordance with maritime law.

2.3 User Classes and Characteristics

- Crane Operators - Employees of the client that are tasked with container transfers and ship balancing in conjunction with the software. Crane operators possess the following characteristics based on our elicitation session in February 2023:
    - Capable of reading/writing in English
    - Operators work in 8 hour shifts
        - 12 to 8, 8 to 4, and 4 to 12
    - Upon the completion of the software, Crane Operators are provided with a short training session and manual
- Ship Captain and Crewmates - Ship personnel will have minimal to no interaction with the software. Upon the completion of all tasks associated with the software, the crane operator will send an outbound manifest of the ship's contents.
- Client - The client employs the crane operators and owns the shipping yard.

2.4 Operating Environment

The operating environment for the use of the software solution is as follows:

- The crane cabin has room for one operator at a time that has access to the following hardware:
    - A barebones work computer that operates the software
        - The computer additionally features steady access to internet, a web browser, a text editor, and a built-in calculator
    - Basic peripherals i.e., mouse and keyboard
- The crane tower is bright and has a 360 degree viewing angle. It is also a noisy environment.

2.5 Design and Implementation Constraints

The following design and implementation constraints are as follows:

- Scalability - Our software is only capable of handling ***X2 Class Ships*** possessing one 8x12 container bay. At the time of this document being finalized, there is no plan to expand upon these dimensions. However, the software is designed to allow for future expansions upon request from the client.
- Hardware constraints - Our software is designed for systems with low computational capabilities.

2.6 User Documentation

Crane Operators are provided with the following:

- full documentation including a 5 page training manual
- 1 hour training virtual training session

2.7 Assumptions and Dependencies

Crane operators are assumed to have the following characteristics:

- Fluency in English including reading/writing capabilities
- No severe vision impairments that negatively affect the operators ability to interact with the software
- Operators do go against the directions indicated by the software

3. External Interface Requirements

3.1 User Interfaces

The user interface is to be designed with a GUI capable of fitting any standard monitor size.

3.2 Hardware Interfaces

The computational requirements of the software must not be demanding enough that most modern devices are incapable of running it. Additionally, the software must be designed to work with standard hardware interfaces from the computer such as a mouse and keyboard.

3.3 Software Interfaces

The software is intended to be operated locally on any device running Windows OS. The software also features a GUI which allows for the user to select specific operations.

3.4 Communications Interfaces

While the device to use the software is expected to have internet access, the software does not require any communication interfaces as it will be run locally on the system.

4. System Features

Below are the system features required for the deployment of the software solution.

4.1 Container Transfer - Loading

4.1.1 Description

The loading portion of container transfer deals with the movement of shipping containers onto the ship using the optimal path determined by an algorithm.

4.1.2 Stimulus/Response

When opting to load a container onto a ship, the user is prompted for the weight and name of the container to be loaded. An algorithm then decides on the most time efficient location for the container. An animation on the screen displays where the operator should place the container. The user is able to press a button labeled “Finished” to indicate that the loading operation is complete and the action is recorded within the log file.

4.1.3 Functional Requirements

- Algorithm - An algorithm must be utilized to locate the optimal location of where to place a container.
- Animation - An animation must be used to display where to place the container.
- User Input - The user must be able to input the weight and name of the container.
- Log file capabilities - The software must be capable of writing to a log file when a container is successfully loaded.

4.2 Container Transfer - Unloading

4.2.1 Description

The unloading portion of container transfer deals with the removal of shipping containers in order to move them onto a truck. This is done using an algorithm that locates the optimal set of moves to access a specific container.

4.2.2 Stimulus/Response

When unloading a container, the user is presented with a screen that displays the containers with their names and specific locations on the ship. Once the user selects a container, the user is presented with a list of operations required to access that container. The user is able to press a button to proceed to the animation for each single move until the selected container has been unloaded. The user is then able to press a button to indicate that the unloading process is concluded and the action is recorded in the log file.

4.2.3 Functional Requirements

- Algorithm - An algorithm is needed to identify the optimal set of moves to access a container.

- Animation - An animation must be used to display where to place the container.
- User Input - The user must be able to select a specific container on the ship.
- Log file capabilities - The software must be capable of writing to a log file when a container is successfully unloaded.

4.3 Ship Balancing

4.3.1 Description

The ship balancing feature allows for the user to balance a ship in accordance with maritime law. This is done so with an algorithm that identifies the optimal place to place a container in order for a ship to become balanced. In the case of a ship being unbalanceable, the algorithm will determine that a ***SIFT*** operation must be performed.

4.3.2 Stimulus/Response

The user is able to press a button in order to balance a ship given the contents of its manifest. Afterwards a set of operations and animations are displayed to indicate where to place a specific container alongside buttons to iterate through each animation. If a ship is considered to be unbalanceable, then the user is notified on the screen that a ***SIFT*** operation must be performed. In this case, the user must manually pull each container into the buffer zone and place them in the locations shown on screen.

4.3.3 Functional Requirements

- Algorithm - An algorithm is needed to find the optimal place to place containers on a ship in order to achieve balance in accordance with maritime law. Additionally, this algorithm needs to have a way of determining if a ship is unbalanceable.
- SIFT - Ship balancing must be capable of performing ***SIFT*** in the case of a ship not being balanceable.
- Log file capabilities - The software must be capable of writing to a log file that a ship has become balanceable in accordance with maritime law.

4.4 Outbound Manifest Output

4.4.1 Description

The outbound manifest output is a feature in which the software writes and outputs a manifest for a ship leaving the port.

4.4.2 Stimulus/Response

When a user finishes performing their operations on a ship they are able to hit a button labeled "Finish". A text file is then uploaded to a folder on the computer's desktop labeled "output" where the user is able to access the outbound manifest. Upon the outbound manifest being uploaded the user receives a notification indicating that the file must be sent out. The generation of this outbound manifest is indicated in the log file and denoted with the original manifest name appended with "OUTBOUND".

4.4.3 Functional Requirements

- Manifest Input - In order for an outbound manifest to be generated, a ship's original manifest must be uploaded to the software.
- Log file capabilities - The software must be capable of writing to a log file when an outbound manifest is generated.

4.5 Log File Capabilities

4.5.1 Description

The log file catalogs the successful operations performed by the software and user. The user also has the capability of adding comments and signing in.

4.5.2 Stimulus/Response

Upon a user accessing the software, they are prompted to sign in with their first and last name. This name is recorded within the log file and the previous user is automatically signed out. In the case that a user wishes to add a comment to the log file, they select from a drop down menu in the top right which gives the option to add a comment. The log file updates automatically upon a user entering a comment. Every time a successful operation is performed, the log file will automatically update with the exact time stamp. Upon the user pressing the “Finish” button after performing the operations, the log file automatically writes to a folder on the computer desktop under the label “output”.

4.5.3 Functional Requirements

- User input - The software must be able to record what the user types in order to update the log file with comments and user sign-ins.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The performance requirements are as follows:

- For 100 different manifest files, a transfer request will display the accurate ship configuration every time.
- For 100 different manifests, a balance request will display the accurate balancing for a ship configuration every time.
- For 100 different manifests, a user request for a transfer will return an optimal result and an accurate outbound file.

5.2 Safety Requirements

This software solution must comply with the Port Crane Security Act of 2022 regarding basic security concerns.

Appendix A: Glossary

- ***SIFT*** - A legal maritime operation to be conducted when a ship is technically unbalanceable
- ***X2 Class Ships -*** Ships with only one bay

Program Design

Front End:

For the front end we are using Tkinter which is built into python. We made the decision based on the following reasons:

- Tkinter is included with Python. This will keep it simple when combined with the backend such that the User will not have to install anything additional to run this.
- Tkinter is simple and easy to learn. Our team lacks experience with building frontend applications. Tkinter being easy to learn will make us able to complete this project within the given timeline without dedicating time to learning something more complicated.

Back End:

- **Ship Array**
    - Initialize a numpy array of dimensions 8 rows by 12 columns of type object.
    - For access to this array, make two helper functions for row and column to convert from User notation to matrix notation.
        - Row: Formula: 8 - row ;; Ex. Row 7 == 8 - 7 = 1
        - Column: Formula: column - 1 ;; Ex. Row 5 == 5 - 1 = 4
- **Read Manifesto**
    - From each line in the manifest which is a .txt file, match all numbers using the regex expression "\b\d+\b" , and append the container name to them from "line.split(",")[-1].strip()".
    - Set array row, column to [Container Name, Container weight] list, all values present in above bullet.
- **Write Updated Manifesto**
    - Obtain the new file name to write to using
      {the old file name} + "OUTBOUND" + ".txt"
      Ex. case1.txt → case1OUTBOUND.txt
    - Clear the file to write by open the file in write mode and closing it
    - Open the file to write in append mode

    - Iterate through all elements in the ship array, and format the output manifest file exactly as specifications have detailed.
        - Row index, Column Index, Container weight with filled zeroes until weight string is 5 characters long, and Container name on each line. 8 times 12 = 96 total lines exactly always
- **Log File**
    - Date and Time
        - Every line of the log file starts with Date and current time. Use python datetime library.
          Example: February 26, 2023: 21:25.
    - Before system start:
        - Prompt user whether or not log file is to be cleared out
        - Prompt user to enter name to sign in
    - During system running
        - Change User
            - Check if there was a previous signed in user, indicate that they are signed out now
            - Sign in new user
        - Enter Comment
            - Allow user to enter a free-form comment into log file
        - Load/Unload/Balancing
            - After each load, unload, and balancing finishing, write to the log file that these functions finish
        - Manifest Open/Close
            - After a manifest is opened or closed, write to the log file that these functions finish
- **Time calculation**
    - In load, unload, and balancing, keep track of the real time minutes movement of containers take. 1 minute, every cell movement in the ship, 2 minutes from the ship OR buffer to a truck, and 4 minutes from the ship to the buffer.
        - For load, add (8 - i) + (j - 1) for the optimal location the container to be placed, where i,j is the row column value for the container to place

        - For everything else, in the move_c method, when a container to move has no other containers above it, add 1 minute to a running time_taken count for the cell column or row it travels.
- **Balancing**
    - Loop through all Ship array, if any container's weight is larger than 0:
        - If the container is on the left side [column <= 6], add to the left cells list, and add to the left weight.
        - If the container is on the right side [column > 6], add to the right cells list, and add to the right weight.
    - Stay in while loop as long as ship is unbalanced
        - Formula for unbalance: max weight is larger of left and right weight. min weight is smaller than left and right weight.
          (max weight - min weight) / max weight is > 10%
        - If unbalanced, enter loop while unbalanced
            - If the left weight is larger, move the first container in the left side list to be the last container in the right side list.
            - If the right weight is larger, move the first container in the right side list to be the last container in the left side list.
            - It is possible for balancing to be impossible, no matter the container orientations. Set a timer of 5 seconds to exit the unbalanced loop and perform SIFT.
                - SIFT is to take all containers to the buffer zone, alternate putting the heaviest container left from the buffer zone on either side of the ship until the lowest possible row is filled. Once it is, repeat this process on higher and higher rows.

    - With the balanced lists of left and right, determine which containers have to actually be moved from the left to the right and be moved from the right to the left.
    - Call the recursive method, move_c, on all containers that need to be moved to their other side.
        - Determine the container column that is to be moved, loop down the column from the top, if there are containers present above the container to move, move them out of the way beforehand by recursively calling move_c. If that container would be moved out of bounds [column 0 or column 13], make it move the other direction.
        - Once there are no containers above the container to move, set its corresponding array value to UNUSED, check if any columns on the way are full, if they are, move the top container out of the way. Continuously try to change the column closer to the destination column, if it is blocked by a container, move up a row, and try again. Once the container has traveled to the destination column, move down rows until the slot right below is not a UNUSED one. Finally, place the container in the ship array to finish the movement of that container.
    - After these steps, and the recursive calls are all resolved, the ship will be balanced, or be legally allowed to sail through SIFT. This is represented through the ship's array.
- **Load/Unload**
    - Load
        - Receive user input for the exact, caps-sensitive name of the container and its weight to load into the ship.
        - Cycle through every column, and keep track of the smallest time to travel in order to place the container.
            - In each column, check if the cell below is UNUSED, if it is, continue down until the container below it's UNUSED.
            - Keep track of the lowest time container placement in each column.

            - Formula: (8 - row number to be placed) + (column number to be placed - 1) time
            - Choose the container placement location for the container with lowest time in the bullet above
    - Unload
        - Receive user input for the exact, caps-sensitive name of the container to unload off the ship.
        - Utilize the move_c function as specified in "Balancing" with location parameter -1 to indicate to move the container out of the ship.
            - Pass in the container name for the function to remove containers on top of the container, and remove it out of the ship.

System Design

- **Understanding The Problem**
    - You own a single port in Long Beach, California, U.S., and "get paid per number of containers moved, so the faster [you] work, the better". [25]
    - Your employees have "at least high school educations", "speak/read basic English", and work on "barebones" PCs with "fast ethernet". [27]
    - All ships that come through your port have one bay, and the maximum amount of containers in that bay are within 8 rows and 12 columns. [26]
    - Two different tasks to be completed through software:
        - **1) Loading and Unloading process of containers to trucks**[9]
        - **2) Balancing the ship through a specific legal definition**[23]
    - **Manifest:** contains 1) the name of each container, 2) the location of said container, and 3) the weight of said container within the ship's bay. [15]
        - This manifest of the ship is to be inputted by the operator into the software before either task of loading/unloading or balancing the ship can occur. [27]
    - **Transfer list:** describes the set of container moves to be performed by the operator of the crane; it is the duty of the operator to relay those instructions as input to our software. It is not the duty of the software to read the transfer list. [27]
    - **Loading/Unloading**
        - Given an operator providing a container and location to place that container, produce software to provide an optimal ordering of container movements to complete that task in regards to time. [9]
            - There is to be a "simple animation" in regards to the movement of each container and an estimate of how much time all these movements will take in total. [9]
            - After the operator confirms that the container has been moved to the inputted location, the software will edit the given manifest to reflect said results. [9]
    - **Balancing**
        - Given an operator stating that a balancing of the ship is to be done, produce software that will provide an optimal list of container movements to complete that task in regards to time. [23]
            - The balancing of a ship is defined as "if the total mass of the port

side, and the total mass of the starboard side are within ten percent" of the weight of the heavier side. For example, if the heavier side of the ship is 100 kg, the lighter side must be 90 kg or heavier in order for the ship to be legally considered balanced. [24]
            - There is to be a "simple animation" in regards to the movement of each container and an estimate of how much time all these movements will take in total. [9]
- **Log File**
    - You require a log file for legal purposes to contain events such as the signing in and out of employees, the movement of containers, and the opening and closing of the manifest file all with time stamps attached to each event. [28]
        - The log file is to be in .txt format, or in other words, "plain text format", readable on notepad. [29]
        - The time representation used in the log file is to be in military time, A.K.A. 24 hour clock. For clarity's sake, 23:00 is to represent 11 P.M. [30]
            - The log file will be in Pacific Standard Time as the port is in Long Beach, CA
        - Your operators will be able to write their own typed-out comments to the log file. [28]

- **Assumptions (required to design our system)**
    - All operators are capable of reading/writing in English
    - All operators do not have any severe vision impairments that will negatively affect their ability to operate the software
    - Operators will never go against the software or operate on their own accord
    - The crane has access to electricity and possesses outlets for a computer to access
    - The computer that the software will run on will have the appropriate peripherals for work such as:
        - Mouse and Keyboard
        - Monitor
    - The company computer will run on the latest windows operating system

- **Stakeholders (those directly or indirectly affected by the software)**
    - Port of Long Beach
    - Mr.Keogh
        - Mr.Keogh's Crane Operators
    - Ship Captain
        - Ship Crewmates
    - Truck Drivers
    - Companies that retain ownership of the containers
    - Customers of products stored within the containers
    - Department of Homeland Security
    - Insurance Companies

- **System Inputs**
    - **Manifest File** - The file sent from the ship that represents the layout of it's bay and the position of all containers onboard(.txt file format)[15]
        - Sent by the incoming ship before it arrives and the user never manually edits the file[16]
        - Total of 96 lines, as it represents an 8 x 12 grid as the largest possible dimension for an incoming ship[15]
        - Layout: Coordinates, Container Weight, Slot Description[15]
            - Ex: [x,y], {Weight}, Slot Description
                - Slot Descr: NAN, UNUSED, or CONTAINER LABEL
    - **Employee Information** - Information required by an employee before using the software. [17]
        - Full Name Layout: FIRSTNAME+[SPACE]+LASTNAME[17]
        - String with no numeric numbers[18]
        - No case sensitivity and no character limit
        - Typed in a textbox and pushed to log file[19]
    - **Employee Comments** - Employee can write a comment for certain situations regarding a container.[12]
        - No character limit

        - Available only for log file[12]
        - Typed in a textbox and pushed to log file[19]

- **Operation Keys** - Specific keys assigned to initiate specific tasks
    - Only a single press of the assigned key will be required
    - Example Key and Task Pairs:
        - Press S to switch users
        - Press A to proceed to next animation
        - Press C to push a comment to log file
- **System Clock** - An internal pulse, representative of a second, used by the computer to keep the correct time.[20]
    - Required for specific utilizations of the date and time in the software[11]
    - As a preventative measure, the software will change your computer settings to ensure only an administrator can manipulate the system clock.
- **Entries For Offloading** - The process required by an employee to indicate the crates that need to be offloaded
    - Employee selects offloading operation
    - Grid displays with the containers and company names[21]
    - All desired containers are selected, as designated by the transfer file[16]
    - Employee clicks the DONE option to indicate they are finished selecting
- **Entries for Loading** - The process required by an employee to indicate the crates that need to be loaded
    - Employee selects loading operation
    - The container label is entered into a textbox, then press ENTER
        - A string of up to 256 characters and case sensitive[22]
    - The container weight is entered into a textbox, then press ENTER
    - Employee can continue to add more entries

    - Employee clicks the DONE option to indicate they are finished

- **System Outputs**
    - **Program start** - There will be various prompts at the very beginning when the program is booted up.
        - A prompt to input a name to sign in as a user
        - A prompt that asks if the user would like to start a new log file.
            - If YES, it will create a default log file named "KeoghLongBeach.txt". It will then prompt the user for what year to append to the default log file.
            - If NO, there will be no new log file created. [14]
        - If the program has been interrupted in the middle of execution, there will be a prompt that asks if the application will resume where it left off.
        - A prompt asking the user to upload the Manifests document.
        - A prompt asking the user whether they would like to balance the ship, or start a transfer.
            - If the user selected to start a transfer, it will prompt the user to select between loading and unloading.
    - **Order of operations list** - For the transfer service, this is a text list of operations that the employee will perform to result in the fastest transfer. For the balance service, it will be a list showing how to balance the ship in the fastest amount of time [9]
    - **Name of the Manifest/Ship**- The name of the manifest and ship will be displayed at all times. This is to ensure that the employee knows they are working on the correct ship/manifest file.
    - **Time Estimation** - The estimated time until completion will be displayed after an employee has inputted the Manifest file. [9]
    - **Visual Representation of the Manifest** - This will be displayed as a grid displaying the names of each of the companies that the containers belong to. From this grid, operators may choose which slot represents a container to unload. [10]

- **User Comment Prompt** - Upon pressing a designated key, a prompt will appear to allow the user to input a message. [12]
    - The employee will be able to add comments at any time the program is running. These comments will be outputted through the log file. [14]
- **Animation** - This will be an animation showing the best order to move the containers. [9]
    - There will be one animation for each atomic transfer.
    - This will be an endless loop with one change to it per second for each atomic transfer. [7]
    - To proceed to the next animation, an employee can press a designated key.
    - When every animation has been displayed, the animation will output a success message [8]
    - The animation will be displayed for both balancing and transfers.
- **Changing Users** - This will be an output prompting an employee to write their full name. [5]
    - At any point of the program, the current employee can press a designated key to trigger this output.
    - The output will allow an employee to write out their full name to switch users.
    - Only log in is recorded in the output logs, signing out is implicit. [6]
- **Altered Manifest File** - This will be outputted after finishing a cycle. It will show the current state of the manifest file after certain containers have been moved. [9]
    - The name of this modified version of the manifest file will be MANIFEST NAME + OUTBOUND, for example, TitanicJonesII.txt will have a modified version named "TitanicJonesIIOUTBOUND.txt [13]
- **Reminder Popup** - At the same time that the Outbound Manifest File is written, a reminder pop-up to the operator to send the file will be displayed. The pop-up will only be closed when a user activates the close operation. [9]
- **Log File** - This will be a file that will log certain actions in the program. This is needed for legal purposes [11] With the following properties:
    - The "certain actions" being referred to are

            - 1. Time and personnel of a shift change
            - 2. Opening or closing a manifest
            - 3. Every atomic move of a container
            - 4. Any ad-hoc notes that the user wants to make [7]
        - It will be written in plain text to be opened in notepad [11]
        - The log file will follow the naming convention KeoghLongBeach[Year].txt where [Year] will be the current year [13]

- **System Maintenance**
    - In the event that the size of a container increases or decreases by any amount, our software representation of the ship bay will need to be updated in order to maintain the functions of onloading, offloading, and balancing.
    - In the event that the size of the buffer zone increases or decreases, our software representation of the buffer zone will need to be updated in order to maintain the functions of onloading, offloading, and balancing.
    - In the event that the size of the largest ship bay possibly increases from 8 rows and/or 12 columns, our software representation of the ship bay will need to be updated in order to maintain the functions of onloading, offloading, and balancing.

- **Training And Documentation**
    - The software will have full documentation including a 1-5 page instruction manual (both physical and digital)
    - All employees will undergo a training session in order to familiarize them with the software
        - This training seminar will be done virtually in a 1 hour session

- **Conformance With Legal Mandates**
    - EPA* Regulation[2]
        - Complies with port and vessel legislature

- Port Crane Security and Inspect Act of 2022[3]
  - Obeys basic security concerns
- Ocean Shipping Reform Act of 2022[4]
  - Our software does not have an effect on shipping fees induced by container relocation

- **Acceptance Testing**
  - **Metrics For Success**
    - For 100 scenarios, SIFT will only be applied 1 time maximum. [31]
    - The system will be able to be resumed within 30 seconds of running after a power cut
    - The system can be restored at any given point of the program with no memory loss
    - The algorithm provided will be optimal or near optimal. In particular, if another software package, or a human using "paper and pencil" could provide an alternative solution that is more than 5% cheaper, the entire project will be deemed a failure. This applies to both transferring and balancing.
    - For 100 different manifest files, if the user requests for balancing, the outbound file will accurately display a balanced ship every time.
    - For 100 different manifest files, if the user requests for transfers, the outbound file will accurately display the resulting ship configuration every time.
    - For 100 different manifest files, if the user requests for transfers, there will not be a container left in the buffer zone.
    - For 100 manifest files containing varying amounts of duplications of container names, if a user request for a transfer, the program will be able to output:

      1. An optimal result
      2. An accurate Outbound file

References

1. Questionswithanswers2.pptx -- Slide 5
2. https://www.epa.gov/vessels-marinas-and-ports
3. https://www.congress.gov/bill/117th-congress/house-bill/6487/text?r=2&s=1
4. https://www.cantwell.senate.gov/imo/media/doc/Ocean%20Shiping%20Reform%20Act%20of%202022%20One-Pager.pdf
5. Problem Overview by Mr. Keogh -- Slide 34
6. Questionswithanswers2.pptx -- Slide 11
7. Questionswithanswers.pptx -- Slide 11
8. Questionswithanswers.pptx -- Slide 15
9. Problem Overview by Mr. Keogh -- Slide 13
10. Problem Overview by Mr. Keogh -- Slide 16
11. Problem Overview by Mr. Keogh -- Slide 33
12. Problem Overview by Mr. Keogh -- Slide 35
13. Questionswithanswers.pptx -- Slide 32
14. Questionswithanswers2.pptx -- Slide 3
15. Problem Overview by Mr. Keogh -- Slide 23
16. Problem Overview by Mr. Keogh -- Slide 12
17. Questionswithanswers.pptx -- Slide 18
18. https://www.birthcertificatecopy.com/articles/naming-laws-in-each-u-s-state/
19. Questionswithanswers.pptx -- Slide 4
20. https://www.computerhope.com/jargon/c/clock.htm
21. Questionswithanswers.pptx -- Slide 40
22. Questionswithanswers.pptx -- Slide 25
23. Problem Overview by Mr. Keogh -- Slide 15
24. Problem Overview by Mr. Keogh -- Slide 25
25. Problem Overview by Mr. Keogh -- Slide 7
26. Problem Overview by Mr. Keogh -- Slide 8
27. Problem Overview by Mr. Keogh -- Slide 14
28. Problem Overview by Mr. Keogh -- Slide 17 and Slide 18
29. Problem Overview by Mr. Keogh -- Slide 35
30. Problem Overview by Mr. Keogh -- Slide 33

31. Elicitation meeting 1/27/2023 – Question asked and answered
32. Problem Overview by Mr. Keogh -- Slide 27

Unit Testing

Back End:

Planning before coding on 2/16/2023:

To test the correctness of the log file:

1) Check if the log file resets correctly when the operator selects yes to reset the log file.
2) Check if the log file correctly reflects the change of user when the operator selects to change user.
3) Check if the log file correctly reflects the comment entered by the operator.

To test the correctness of the outbound manifest:

4) Check if the outbound manifest has the correct name, and if the ship array is initialized to the correct values.
5) Check if the finished outbound manifest has been updated correctly with ship array modifications.

To test the correctness of the Load operation:

6) Check the correct load of a singular container ("Test",123 kg) based on the matching ship arrays and the estimated time taken to complete the operation.
7) Check the correct load of a singular container ("Test",123 kg) with a full column 1 based on the matching ship arrays and the estimated time taken to complete the operation.
8) Check the correct load of a singular container ("Test",123 kg) with an empty column 1 and almost full column 2 based on the matching ship arrays and the estimated time taken to complete the operation.

To test the correctness of the Unload operation:

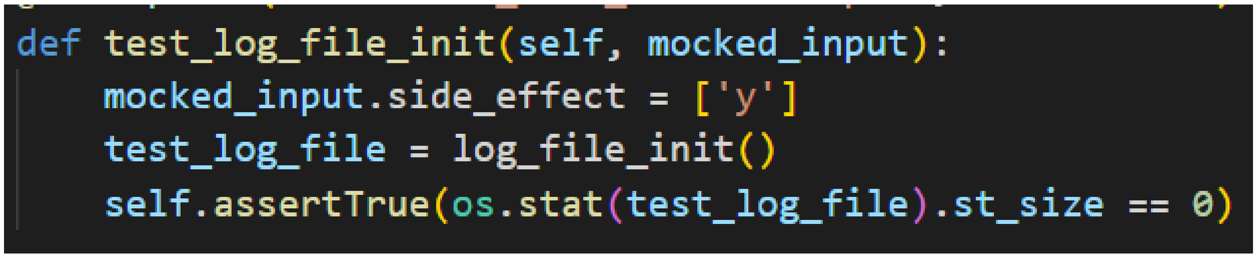
9) Check the correct unload of a singular container ("Test",123 kg) with no other containers on the ship based on the matching ship arrays and the estimated time taken to complete the operation.
10) Check the correct unload of a singular container ("Test",123 kg) with a single container above it based on the matching ship arrays and the estimated time taken to complete the operation.
11) Check the correct load of a singular container ("Test",123 kg) with a full column of containers above with expected container movement based on the matching ship arrays and the estimated time taken to complete the operation.

To test the correctness of the Balance operation:

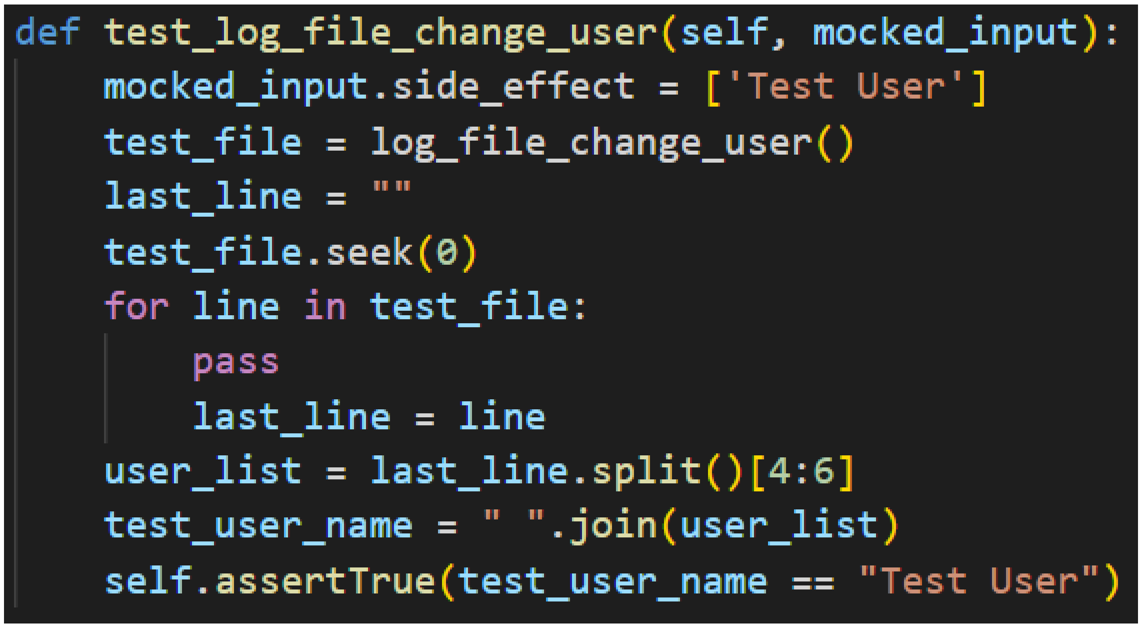
12) Check if the formula implemented for the balance formula is correct through utilizing multiple weights that are edge cases.
13) Check the correct balancing of a ship that is able to be balanced through its ratio of left and right side weights based on the matching ship arrays and the estimated time taken to complete the operation.

14) Check the correct balancing of a ship that is able to be balanced through its ratio of left and right side weights with larger weights based on the matching ship arrays and the estimated time taken to complete the operation.
15) Check the correct balancing of a ship that is not able to be balanced through its ratio of left and right side weights with larger weights through performing SIFT based on the matching ship arrays and the estimated time taken to complete the operation.
16) Check the correct balancing of a ship that is not able to be balanced through its ratio of left and right side weights with larger weights through performing SIFT with larger weights based on the matching ship arrays and the estimated time taken to complete the operation.

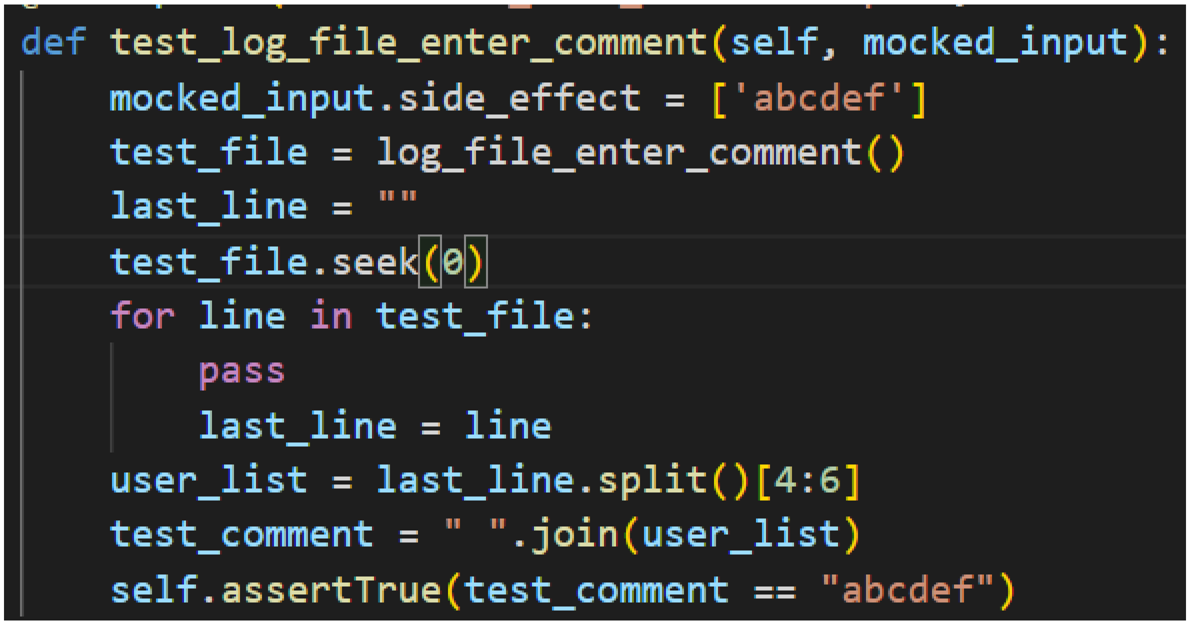
Github: Test cases within "backend_test/backend_test.py" are associated with functions in "backend_test/container_load_balance.py"


```python
def test_log_file_init(self, mocked_input):
    mocked_input.side_effect = ['y']
    test_log_file = log_file_init()
    self.assertTrue(os.stat(test_log_file).st_size == 0)
```

- Test if Log file correctly resets file if user determines that they want it reset

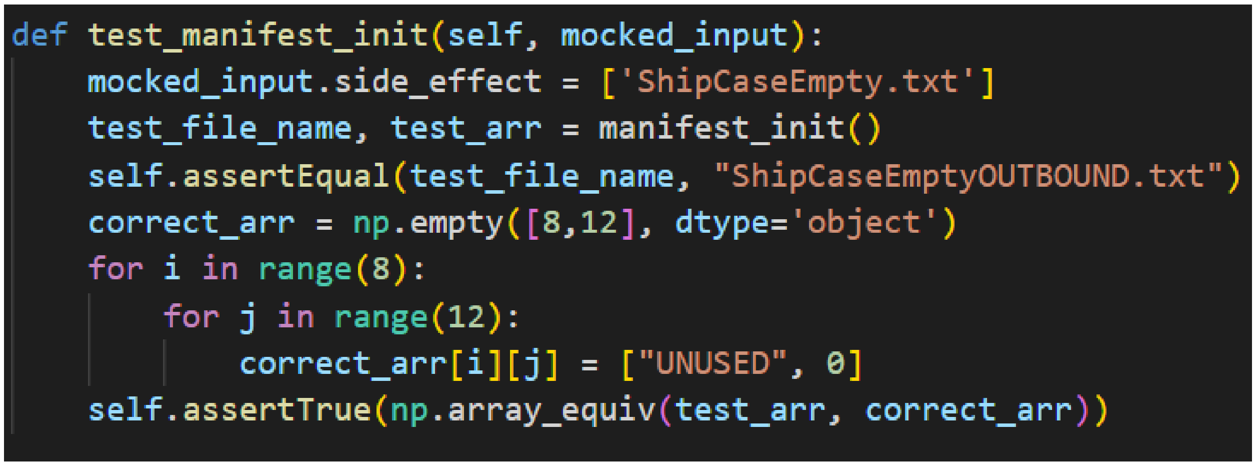

```python
def test_log_file_change_user(self, mocked_input):
    mocked_input.side_effect = ['Test User']
    test_file = log_file_change_user()
    last_line = ""
    test_file.seek(0)
    for line in test_file:
        pass
        last_line = line
    user_list = last_line.split()[4:6]
    test_user_name = " ".join(user_list)
    self.assertTrue(test_user_name == "Test User")
```

- Test if Log file correctly signs in a new user when the new user enters their name



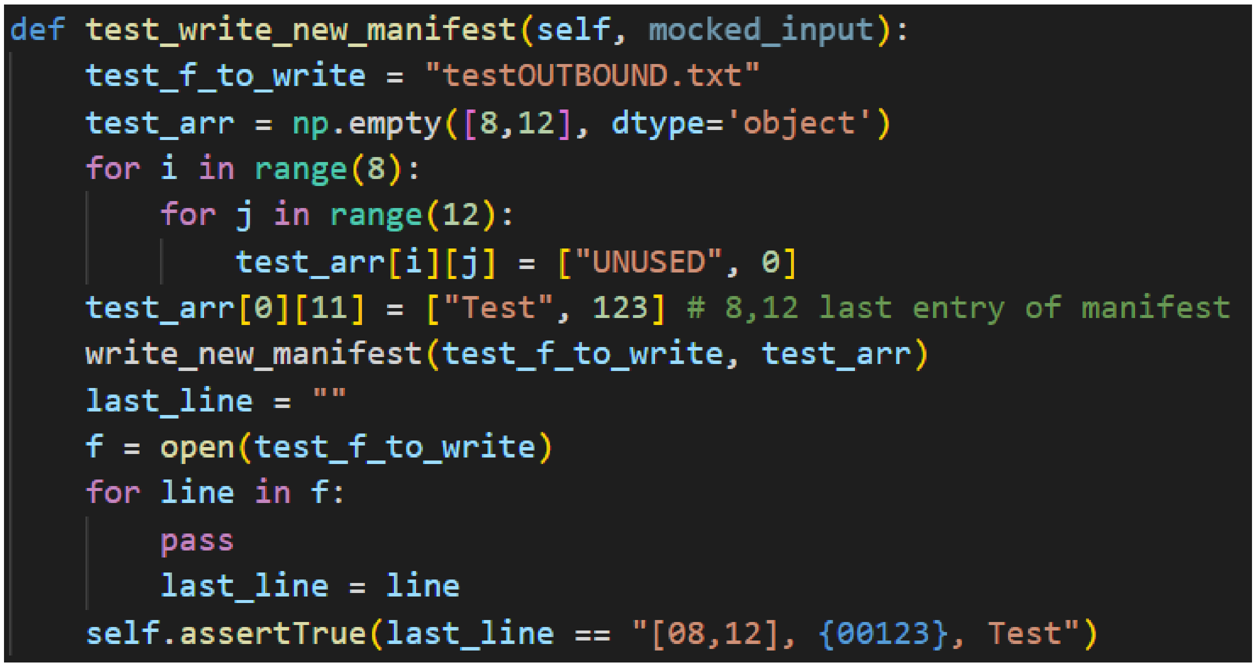
```python
def test_log_file_enter_comment(self, mocked_input):
    mocked_input.side_effect = ['abcdef']
    test_file = log_file_enter_comment()
    last_line = ""
    test_file.seek(0)
    for line in test_file:
        pass
        last_line = line
    user_list = last_line.split()[4:6]
    test_comment = " ".join(user_list)
    self.assertTrue(test_comment == "abcdef")
```

- Test if Log file enter comment correctly appends a log file entry with the given comment



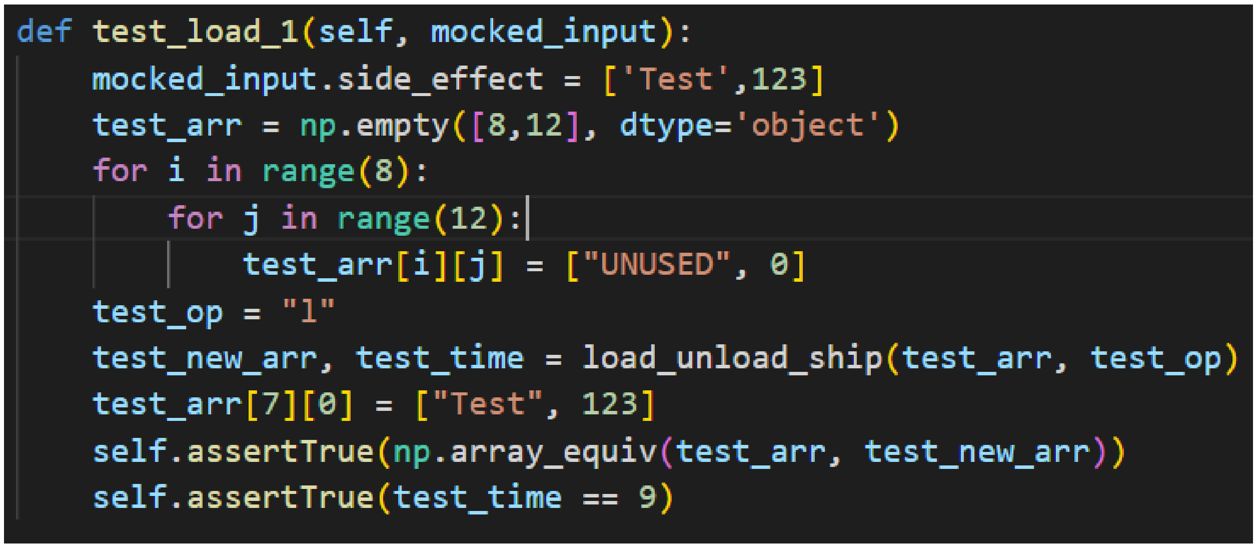
```python
def test_manifest_init(self, mocked_input):
    mocked_input.side_effect = ['ShipCaseEmpty.txt']
    test_file_name, test_arr = manifest_init()
    self.assertEqual(test_file_name, "ShipCaseEmptyOUTBOUND.txt")
    correct_arr = np.empty([8,12], dtype='object')
    for i in range(8):
        for j in range(12):
            correct_arr[i][j] = ["UNUSED", 0]
    self.assertTrue(np.array_equiv(test_arr, correct_arr))
```

- Test if Manifest initialization correctly defaults the ship array to the correctly values given an input file



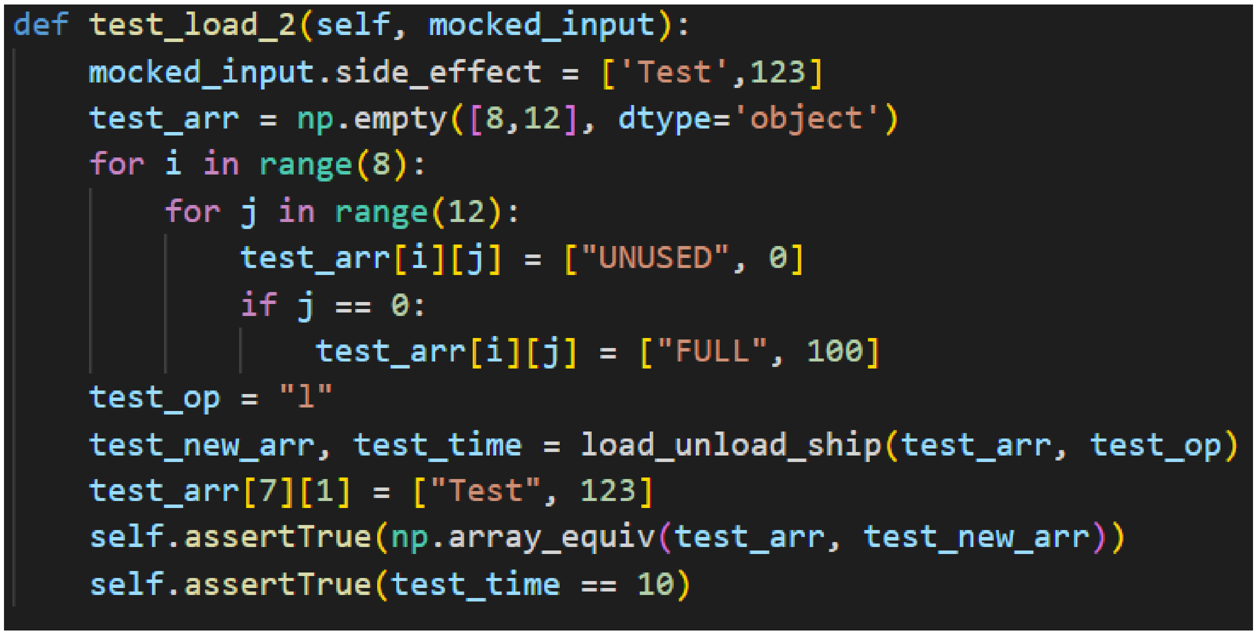
```python
def test_write_new_manifest(self, mocked_input):
    test_f_to_write = "testOUTBOUND.txt"
    test_arr = np.empty([8,12], dtype='object')
    for i in range(8):
        for j in range(12):
            test_arr[i][j] = ["UNUSED", 0]
    test_arr[0][11] = ["Test", 123] # 8,12 last entry of manifest
    write_new_manifest(test_f_to_write, test_arr)
    last_line = ""
    f = open(test_f_to_write)
    for line in f:
        pass
        last_line = line
    self.assertTrue(last_line == "[08,12], {00123}, Test")
```

- Test if Write new manifest correctly updates the old manifest to reflect the container movements from the old ship array



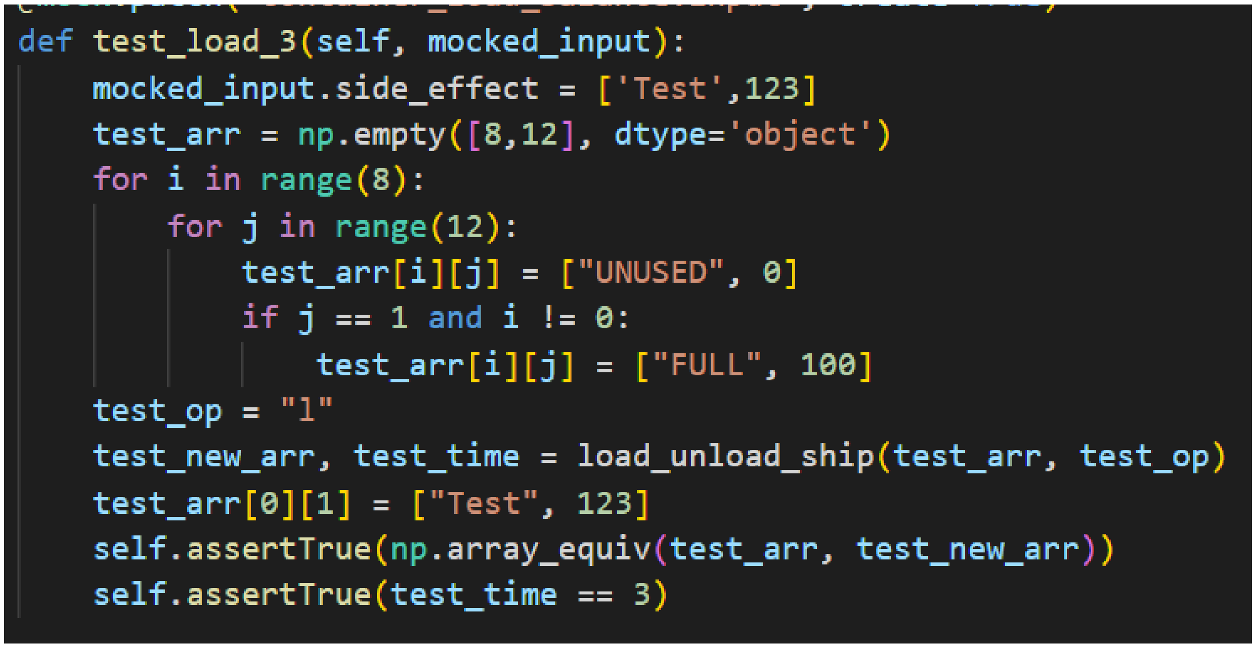
```python
def test_load_1(self, mocked_input):
    mocked_input.side_effect = ['Test',123]
    test_arr = np.empty([8,12], dtype='object')
    for i in range(8):
        for j in range(12):
            test_arr[i][j] = ["UNUSED", 0]
    test_op = "l"
    test_new_arr, test_time = load_unload_ship(test_arr, test_op)
    test_arr[7][0] = ["Test", 123]
    self.assertTrue(np.array_equiv(test_arr, test_new_arr))
    self.assertTrue(test_time == 9)
```

- Test if Load container correctly places the container in optimal crane movement time given user input of the container name and weight



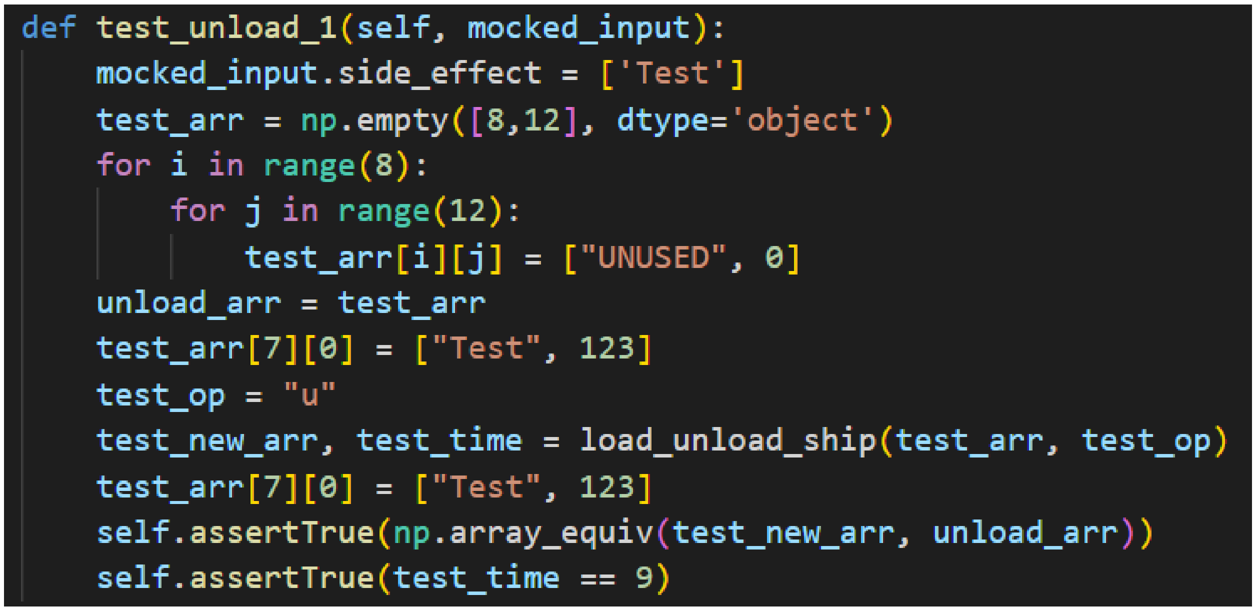
```python
def test_load_2(self, mocked_input):
    mocked_input.side_effect = ['Test',123]
    test_arr = np.empty([8,12], dtype='object')
    for i in range(8):
        for j in range(12):
            test_arr[i][j] = ["UNUSED", 0]
            if j == 0:
                test_arr[i][j] = ["FULL", 100]
    test_op = "l"
    test_new_arr, test_time = load_unload_ship(test_arr, test_op)
    test_arr[7][1] = ["Test", 123]
    self.assertTrue(np.array_equiv(test_arr, test_new_arr))
    self.assertTrue(test_time == 10)
```

- Test if Load container correctly places the container with completely full column 1 in optimal crane movement time given user input of the container name and weight

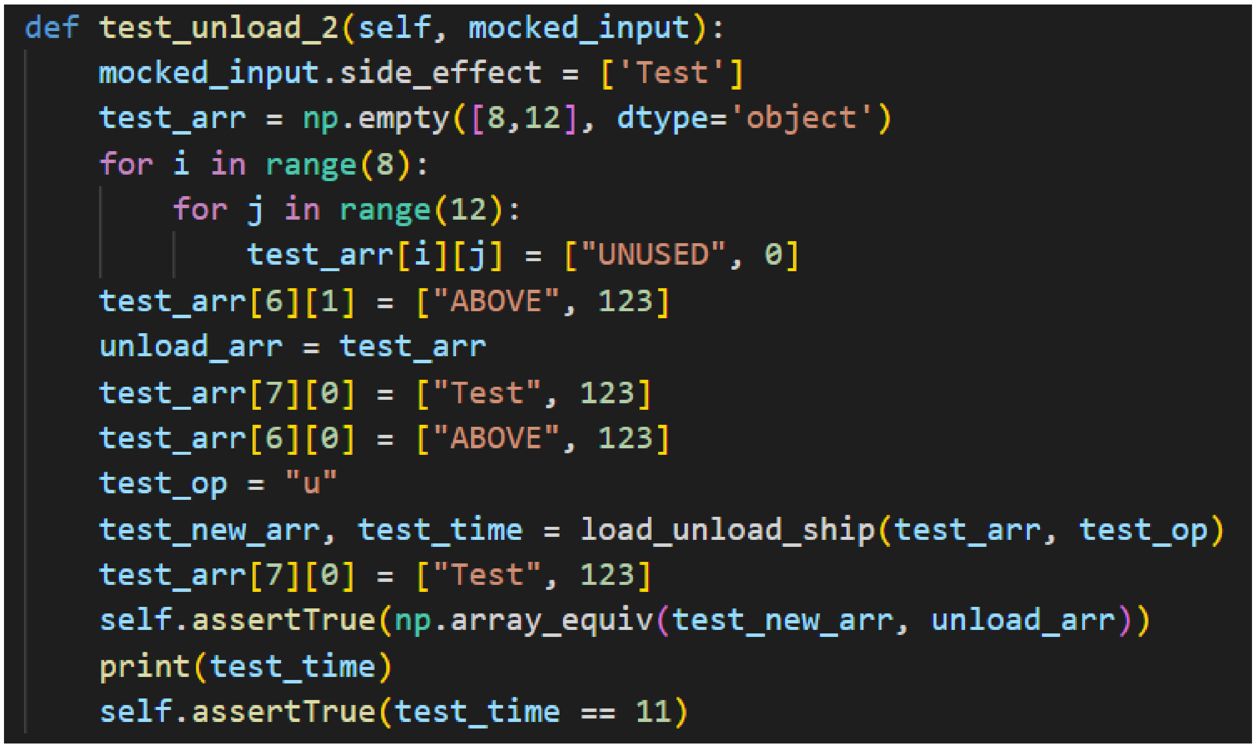


```python
def test_load_3(self, mocked_input):
    mocked_input.side_effect = ['Test',123]
    test_arr = np.empty([8,12], dtype='object')
    for i in range(8):
        for j in range(12):
            test_arr[i][j] = ["UNUSED", 0]
            if j == 1 and i != 0:
                test_arr[i][j] = ["FULL", 100]
    test_op = "l"
    test_new_arr, test_time = load_unload_ship(test_arr, test_op)
    test_arr[0][1] = ["Test", 123]
    self.assertTrue(np.array_equiv(test_arr, test_new_arr))
    self.assertTrue(test_time == 3)
```

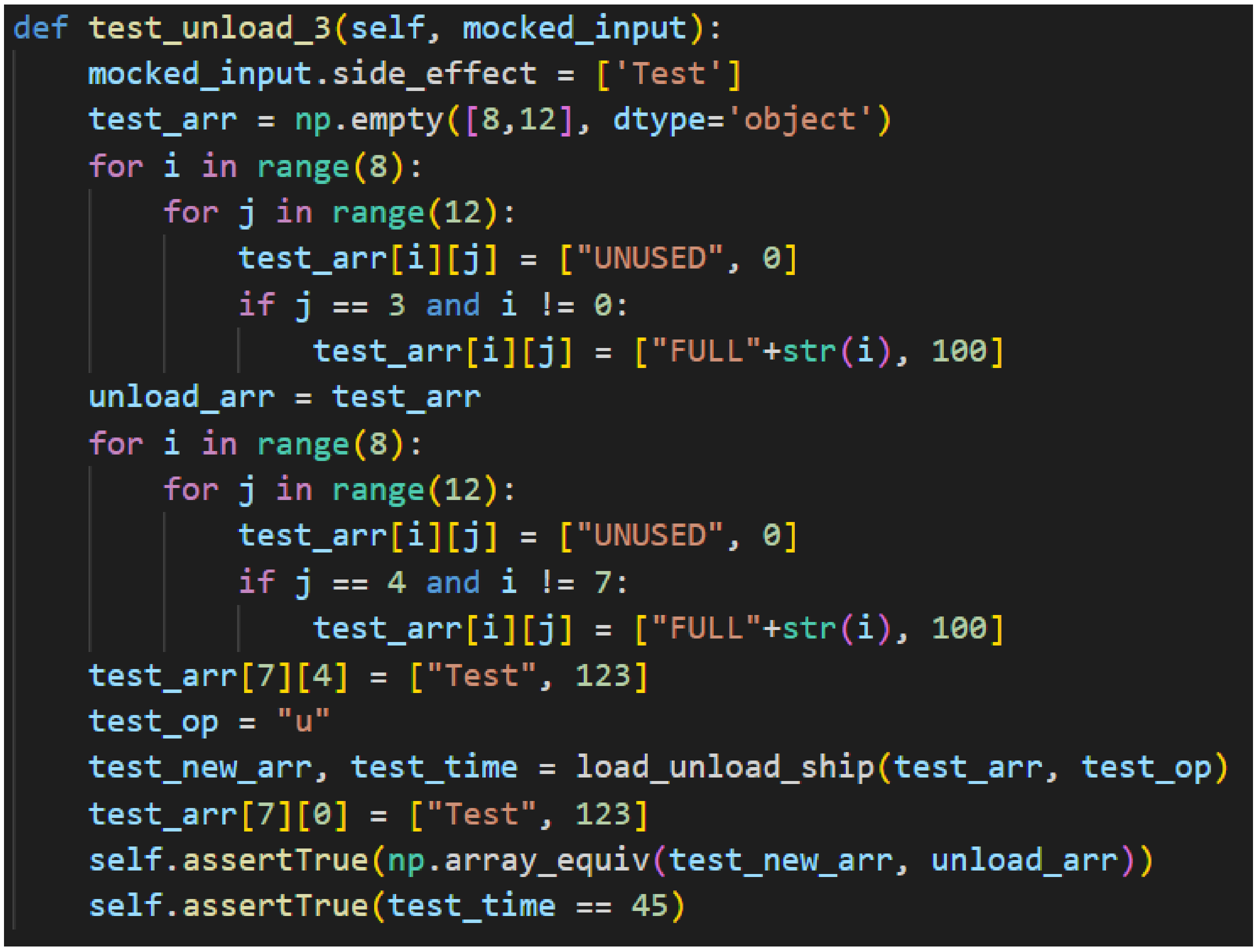
- Test if Load container correctly places the container with almost full column 2 and empty 1 to determine that the software is choosing the quickest container movement to place the container in optimal crane movement time given user input of the container name and weight


```
def test_unload_1(self, mocked_input):
    mocked_input.side_effect = ['Test']
    test_arr = np.empty([8,12], dtype='object')
    for i in range(8):
        for j in range(12):
            test_arr[i][j] = ["UNUSED", 0]
    unload_arr = test_arr
    test_arr[7][0] = ["Test", 123]
    test_op = "u"
    test_new_arr, test_time = load_unload_ship(test_arr, test_op)
    test_arr[7][0] = ["Test", 123]
    self.assertTrue(np.array_equiv(test_new_arr, unload_arr))
    self.assertTrue(test_time == 9)
```

- Test Unload a container if when a container in the ship array is to be unloaded from the ship given user input, it is indeed removed from the ship in optimal crane movement time.

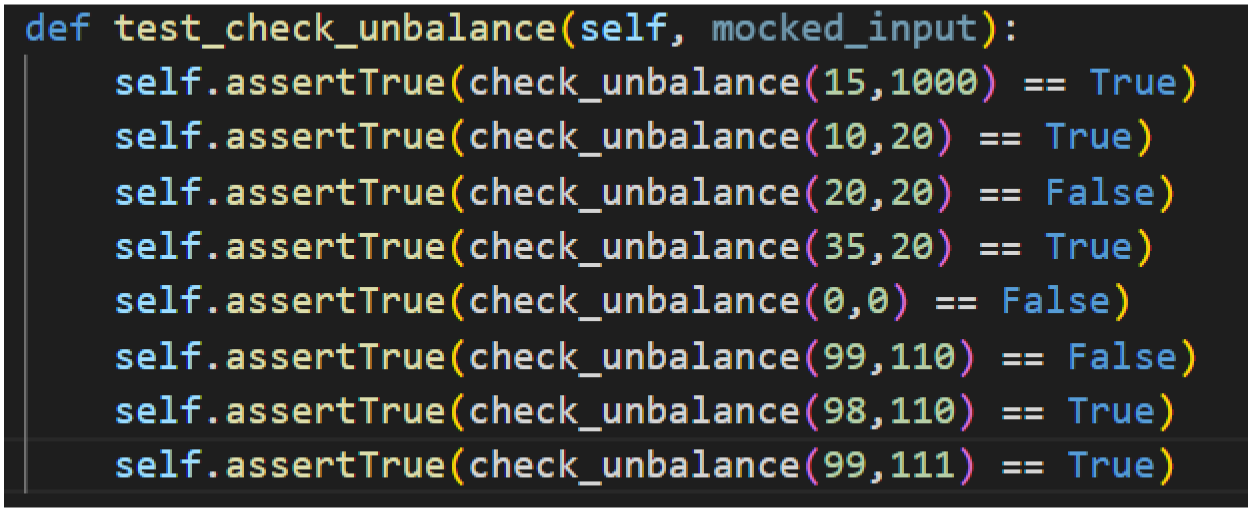

```
def test_unload_2(self, mocked_input):
    mocked_input.side_effect = ['Test']
    test_arr = np.empty([8,12], dtype='object')
    for i in range(8):
        for j in range(12):
            test_arr[i][j] = ["UNUSED", 0]
    test_arr[6][1] = ["ABOVE", 123]
    unload_arr = test_arr
    test_arr[7][0] = ["Test", 123]
    test_arr[6][0] = ["ABOVE", 123]
    test_op = "u"
    test_new_arr, test_time = load_unload_ship(test_arr, test_op)
    test_arr[7][0] = ["Test", 123]
    self.assertTrue(np.array_equiv(test_new_arr, unload_arr))
    print(test_time)
    self.assertTrue(test_time == 11)
```

- Test Unload a container if when a container in the ship array is to be unloaded with a container above it from the ship given user input, it is indeed removed from the ship in optimal crane movement time.



```python
def test_unload_3(self, mocked_input):
    mocked_input.side_effect = ['Test']
    test_arr = np.empty([8,12], dtype='object')
    for i in range(8):
        for j in range(12):
            test_arr[i][j] = ["UNUSED", 0]
            if j == 3 and i != 0:
                test_arr[i][j] = ["FULL"+str(i), 100]
    unload_arr = test_arr
    for i in range(8):
        for j in range(12):
            test_arr[i][j] = ["UNUSED", 0]
            if j == 4 and i != 7:
                test_arr[i][j] = ["FULL"+str(i), 100]
    test_arr[7][4] = ["Test", 123]
    test_op = "u"
    test_new_arr, test_time = load_unload_ship(test_arr, test_op)
    test_arr[7][0] = ["Test", 123]
    self.assertTrue(np.array_equiv(test_new_arr, unload_arr))
    self.assertTrue(test_time == 45)
```

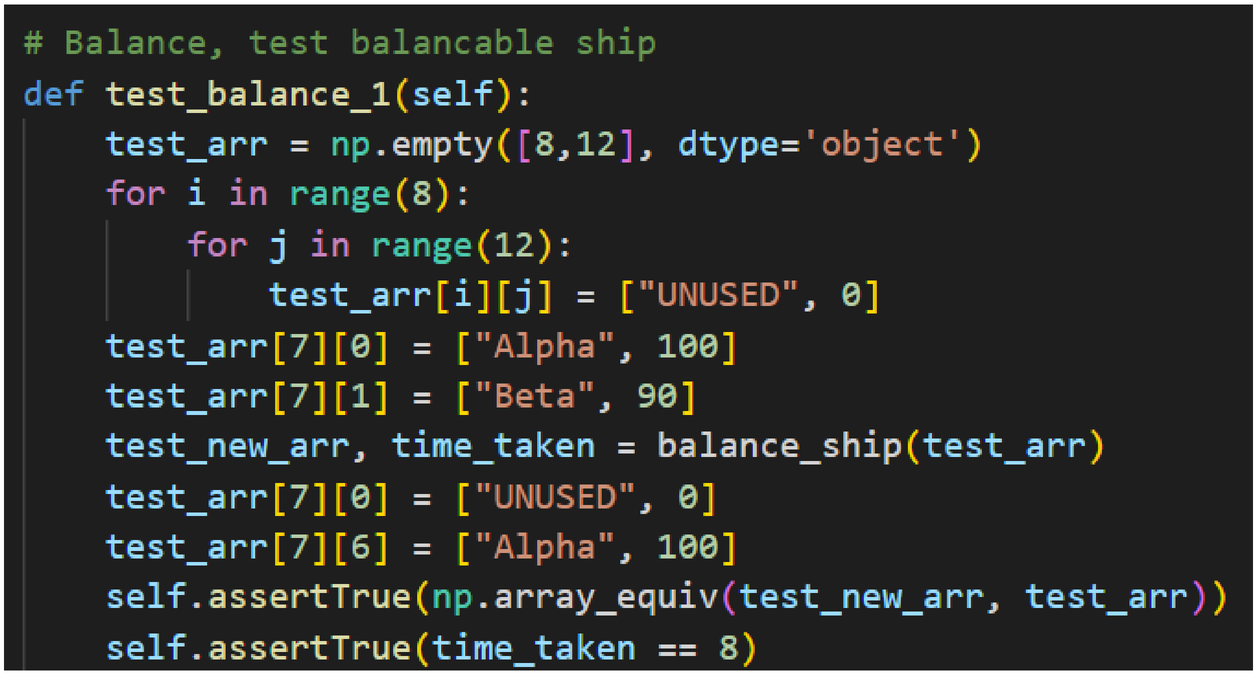
- Test Unload a container if when a container in the ship array is to be unloaded with a full column of containers above it from the ship given user input, it is indeed removed from the ship in optimal crane movement time.



```python
def test_check_unbalance(self, mocked_input):
    self.assertTrue(check_unbalance(15,1000) == True)
    self.assertTrue(check_unbalance(10,20) == True)
    self.assertTrue(check_unbalance(20,20) == False)
    self.assertTrue(check_unbalance(35,20) == True)
    self.assertTrue(check_unbalance(0,0) == False)
    self.assertTrue(check_unbalance(99,110) == False)
    self.assertTrue(check_unbalance(98,110) == True)
    self.assertTrue(check_unbalance(99,111) == True)
```

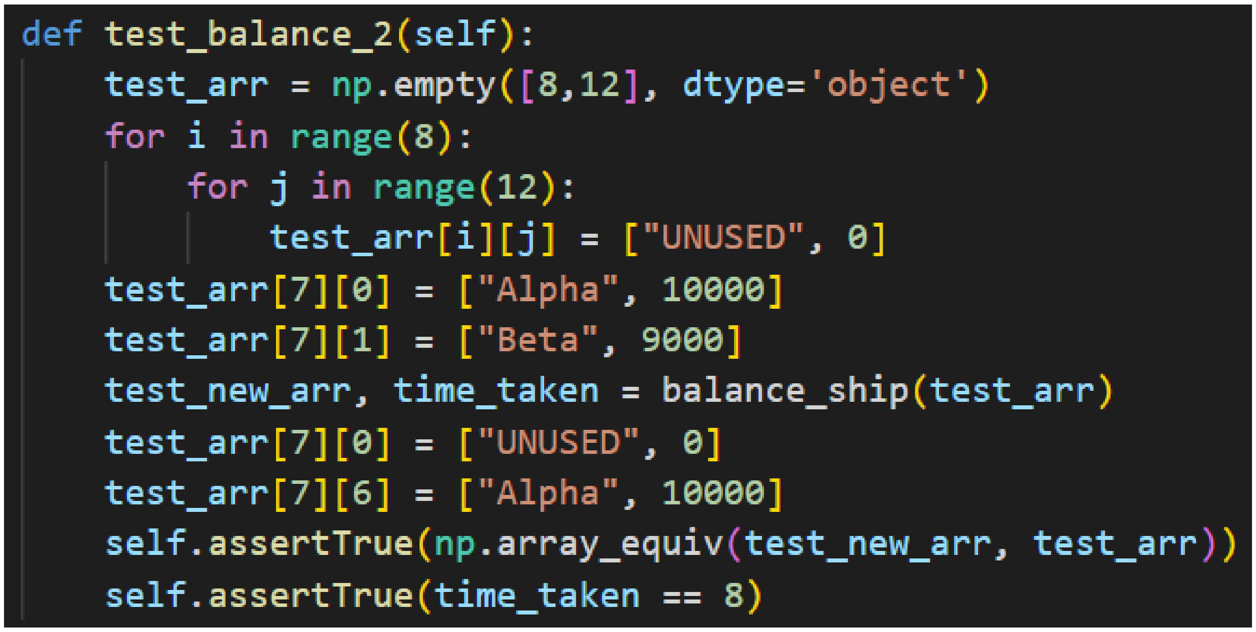
- Test if Check unbalance function correctly determines if two weights are unbalanced given by the formula:

  (max_weight - min_weight) / max_weight > 10%



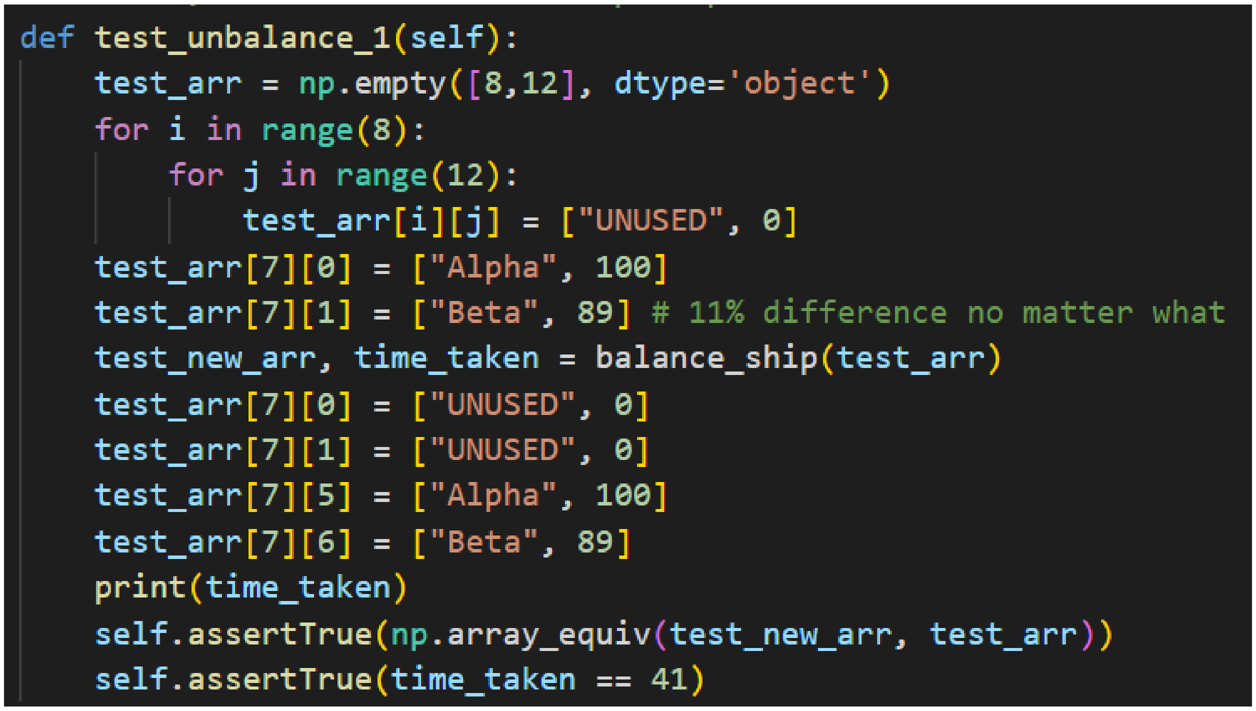
```python
# Balance, test balancable ship
def test_balance_1(self):
    test_arr = np.empty([8,12], dtype='object')
    for i in range(8):
        for j in range(12):
            test_arr[i][j] = ["UNUSED", 0]
    test_arr[7][0] = ["Alpha", 100]
    test_arr[7][1] = ["Beta", 90]
    test_new_arr, time_taken = balance_ship(test_arr)
    test_arr[7][0] = ["UNUSED", 0]
    test_arr[7][6] = ["Alpha", 100]
    self.assertTrue(np.array_equiv(test_new_arr, test_arr))
    self.assertTrue(time_taken == 8)
```

- Test if Balance correctly balances a ship with containers that are possible to be arranged in a legally balanced way as outlined in check_unbalance



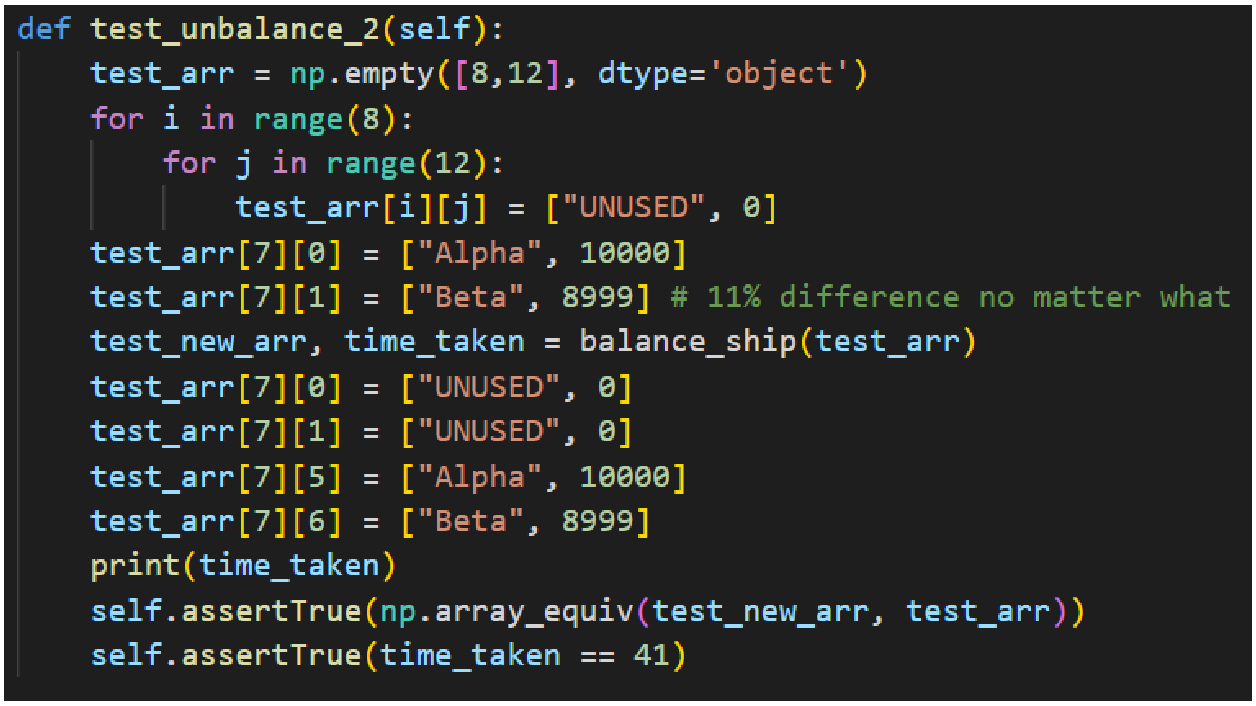
```python
def test_balance_2(self):
    test_arr = np.empty([8,12], dtype='object')
    for i in range(8):
        for j in range(12):
            test_arr[i][j] = ["UNUSED", 0]
    test_arr[7][0] = ["Alpha", 10000]
    test_arr[7][1] = ["Beta", 9000]
    test_new_arr, time_taken = balance_ship(test_arr)
    test_arr[7][0] = ["UNUSED", 0]
    test_arr[7][6] = ["Alpha", 10000]
    self.assertTrue(np.array_equiv(test_new_arr, test_arr))
    self.assertTrue(time_taken == 8)
```

- Test if Balance correctly balances a ship with containers with larger weights that are possible to be arranged in a legally balanced way as outlined in check_unbalance



```python
def test_unbalance_1(self):
    test_arr = np.empty([8,12], dtype='object')
    for i in range(8):
        for j in range(12):
            test_arr[i][j] = ["UNUSED", 0]
    test_arr[7][0] = ["Alpha", 100]
    test_arr[7][1] = ["Beta", 89] # 11% difference no matter what
    test_new_arr, time_taken = balance_ship(test_arr)
    test_arr[7][0] = ["UNUSED", 0]
    test_arr[7][1] = ["UNUSED", 0]
    test_arr[7][5] = ["Alpha", 100]
    test_arr[7][6] = ["Beta", 89]
    print(time_taken)
    self.assertTrue(np.array_equiv(test_new_arr, test_arr))
    self.assertTrue(time_taken == 41)
```

- Test if Unbalance correctly is unable to balance via the legal balancing method. See that SIFT is performed where the containers are alternated between port and starboard sides of the ship from the middle.



```python
def test_unbalance_2(self):
    test_arr = np.empty([8,12], dtype='object')
    for i in range(8):
        for j in range(12):
            test_arr[i][j] = ["UNUSED", 0]
    test_arr[7][0] = ["Alpha", 10000]
    test_arr[7][1] = ["Beta", 8999] # 11% difference no matter what
    test_new_arr, time_taken = balance_ship(test_arr)
    test_arr[7][0] = ["UNUSED", 0]
    test_arr[7][1] = ["UNUSED", 0]
    test_arr[7][5] = ["Alpha", 10000]
    test_arr[7][6] = ["Beta", 8999]
    print(time_taken)
    self.assertTrue(np.array_equiv(test_new_arr, test_arr))
    self.assertTrue(time_taken == 41)
```

- Test if Unbalance correctly is unable to balance via the legal balancing method with larger containers. See that SIFT is performed where the containers are alternated between port and starboard sides of the ship from the middle.

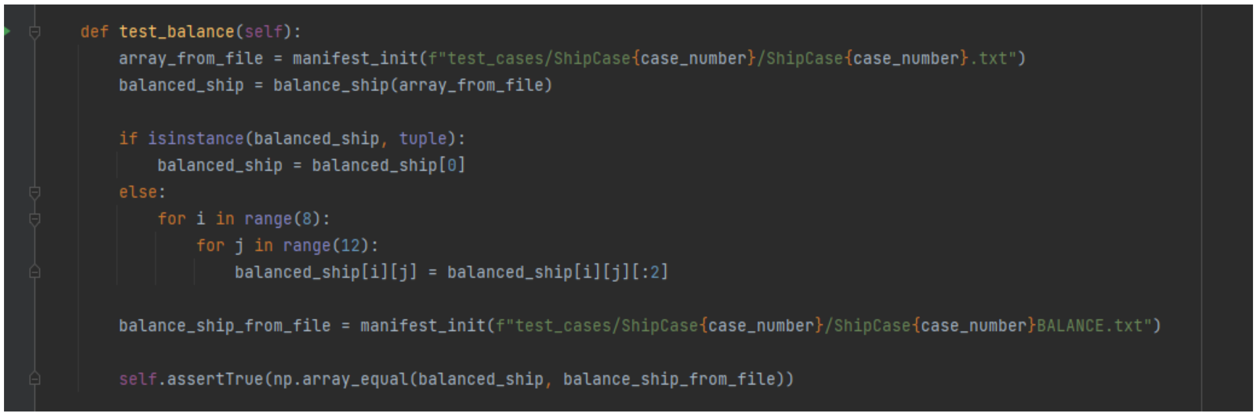
Integration Testing

We decided that once we integrated all of our unittest components, it was important for us that all of the test case files were working in unloading, loading, and balancing in each change we did. For our integration testing whenever we made any significant change to the codebase, we ran through each of the test cases to verify that they were working properly.

For example: At first, we knew that test cases like testcase1 and testcase2 worked because they simply took an unobstructed container from ship to truck. However, in testcase3 there was one obstacle in the way. Testcase3's output was not working so we had to modify how the animation worked and how we got those values for moving a container with obstacles in the way. After we verified that testcase3 worked, we went back and verified with testcase 1 and testcase2.

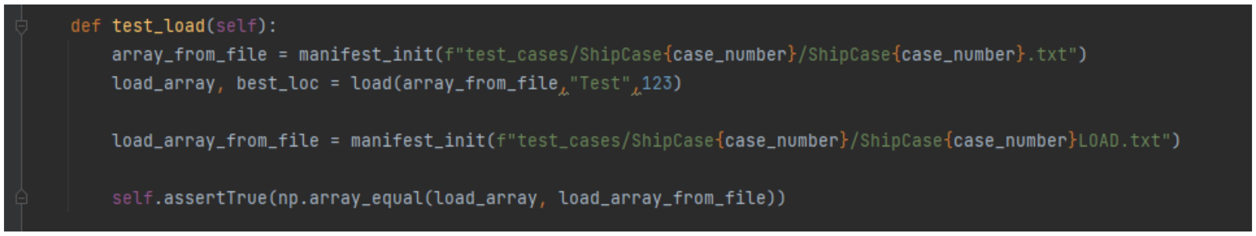
We did this with the help of some unittest.cases to speed along the process, but we also manually tested it when those were not available.

Our integration testing is composed of the following functions:



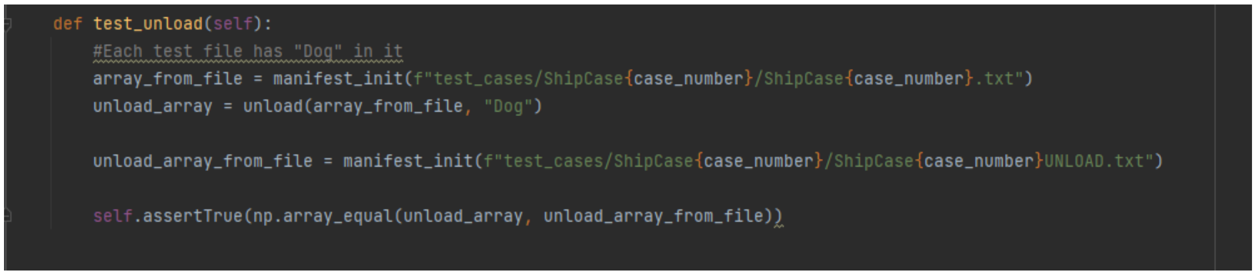
```python
def test_balance(self):
    array_from_file = manifest_init(f"test_cases/ShipCase{case_number}/ShipCase{case_number}.txt")
    balanced_ship = balance_ship(array_from_file)

    if isinstance(balanced_ship, tuple):
        balanced_ship = balanced_ship[0]
    else:
        for i in range(8):
            for j in range(12):
                balanced_ship[i][j] = balanced_ship[i][j][:2]

    balance_ship_from_file = manifest_init(f"test_cases/ShipCase{case_number}/ShipCase{case_number}BALANCE.txt")

    self.assertTrue(np.array_equal(balanced_ship, balance_ship_from_file))
```

Given a sample Manifest file, we need to see if it gets an expected output after balancing. This accounts for both regular balancing and SIFT.



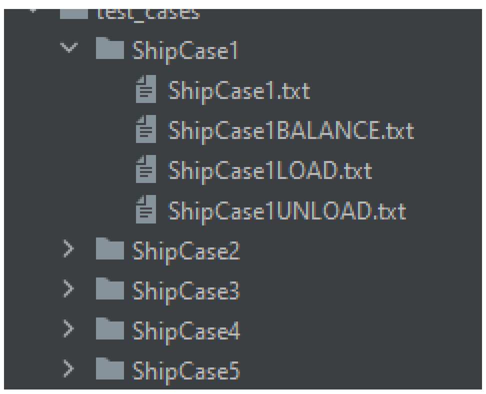
```python
def test_load(self):
    array_from_file = manifest_init(f"test_cases/ShipCase{case_number}/ShipCase{case_number}.txt")
    load_array, best_loc = load(array_from_file,"Test",123)

    load_array_from_file = manifest_init(f"test_cases/ShipCase{case_number}/ShipCase{case_number}LOAD.txt")

    self.assertTrue(np.array_equal(load_array, load_array_from_file))
```

Given a sample Manifest file, we need to see if it gets an expected output after loading a container. This container is constant to help with testing. This accounts for both regular balancing and SIFT.


```
def test_unload(self):
    #Each test file has "Dog" in it
    array_from_file = manifest_init(f"test_cases/ShipCase{case_number}/ShipCase{case_number}.txt")
    unload_array = unload(array_from_file, "Dog")

    unload_array_from_file = manifest_init(f"test_cases/ShipCase{case_number}/ShipCase{case_number}UNLOAD.txt")

    self.assertTrue(np.array_equal(unload_array, unload_array_from_file))
```

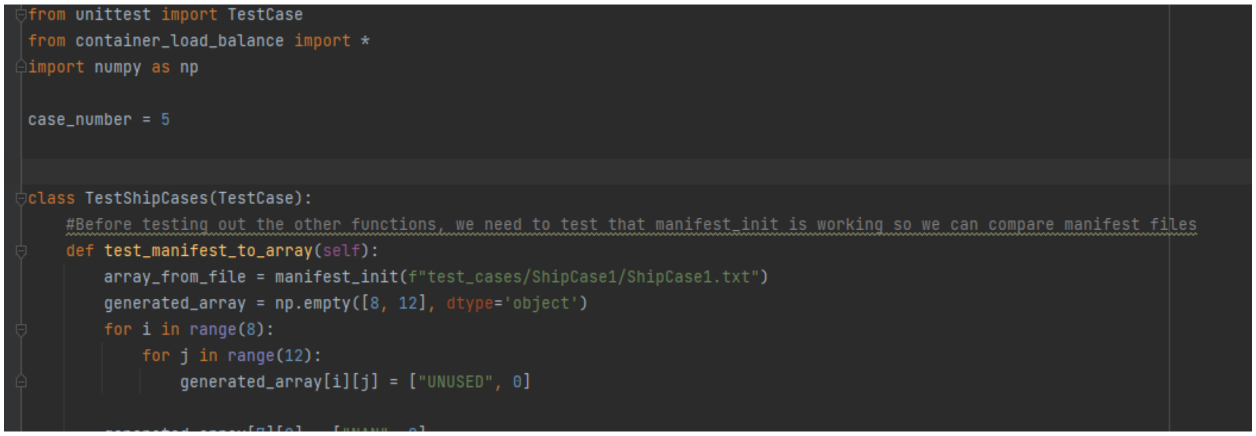
Given a sample Manifest file, we need to see if it gets an expected output after unloading a container. This container to unload is constant within all test files.

The information that these functions pull from are in the same directory and look like this


```
test_cases
  ShipCase1
    ShipCase1.txt
    ShipCase1BALANCE.txt
    ShipCase1LOAD.txt
    ShipCase1UNLOAD.txt
  ShipCase2
  ShipCase3
  ShipCase4
  ShipCase5
```

There is one test file and 3 expected output files for verification.

In order to quickly go through each test we made a constant case_number variable that we can change to quickly go check every single test case works before proceeding with a test.

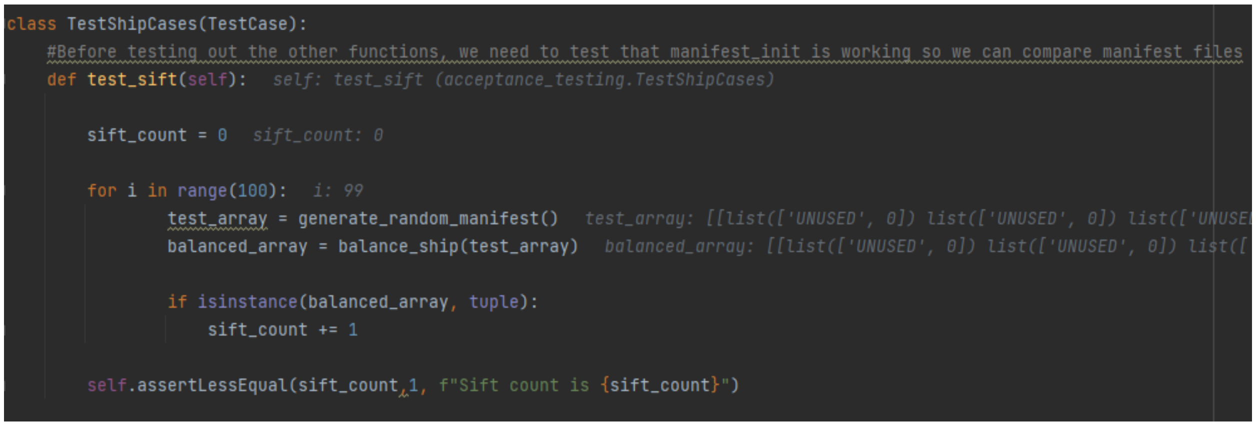

```
from unittest import TestCase
from container_load_balance import *
import numpy as np

case_number = 5


class TestShipCases(TestCase):
    #Before testing out the other functions, we need to test that manifest_init is working so we can compare manifest files
    def test_manifest_to_array(self):
        array_from_file = manifest_init(f"test_cases/ShipCase1/ShipCase1.txt")
        generated_array = np.empty([8, 12], dtype='object')
        for i in range(8):
            for j in range(12):
                generated_array[i][j] = ["UNUSED", 0]
```

Acceptance Testing

Unfortunately with Acceptance Testing, we were unable to produce our expected results. There were a few bugs that we were unable to resolve in time and so we did not make the expected results. We would like to acknowledge that our tests were extensive in scope, and may have not been appropriate for a project within this particular time frame. We could only confirm that all of the test cases were provided by Mr. Keogh (See integration testing for more information). Despite this, we will describe the methods we use to try and get these results.

All of these criteria are taken from ProjectPitch_Version2, slides 42 and 43.

1. For 100 scenarios, SIFT will only be applied 1 time maximum



```
class TestShipCases(TestCase):
    #Before testing out the other functions, we need to test that manifest_init is working so we can compare manifest files
    def test_sift(self):   self: test_sift (acceptance_testing.TestShipCases)

        sift_count = 0   sift_count: 0

        for i in range(100):   i: 99
            test_array = generate_random_manifest()   test_array: [[list(['UNUSED', 0]) list(['UNUSED', 0]) list(['UNUSE
            balanced_array = balance_ship(test_array)   balanced_array: [[list(['UNUSED', 0]) list(['UNUSED', 0]) list([

            if isinstance(balanced_array, tuple):
                sift_count += 1

        self.assertLessEqual(sift_count,1, f"Sift count is {sift_count}")
```

We wanted to mitigate SIFT as much as possible so we created this function to
a. Generate a random test array
b. Balance the array and
c. Test if SIFT was used by checking if the output was a tuple.

2. The system will be able to be resumed within 30 seconds of running after a power cut

We were unable to implement this feature.

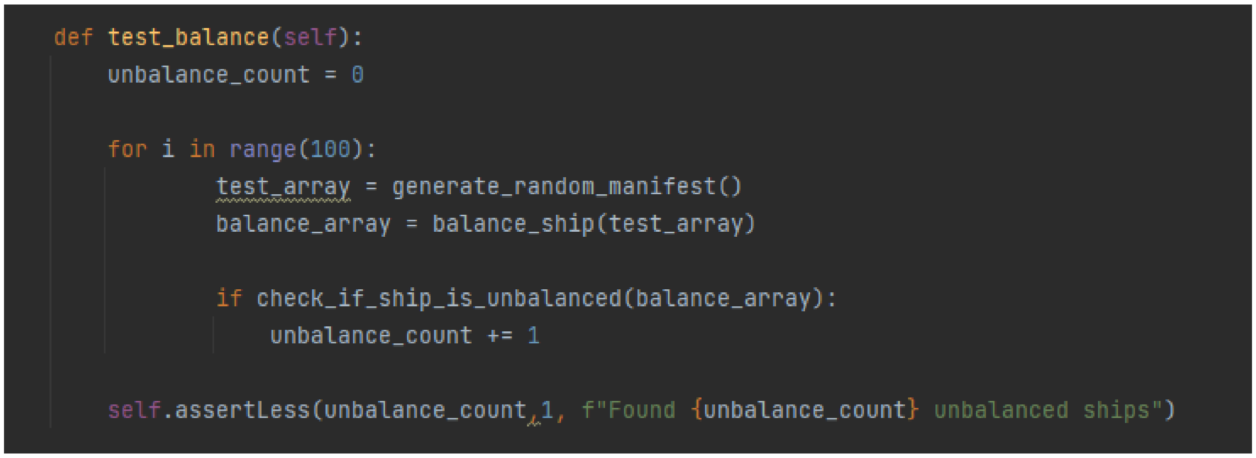
3. The system can be restored at any given point of the program with no memory loss

We were unable to implement this feature.

4. The algorithm provided will be optimal or near-optimal. In particular, if another software package, or a human using "paper and pencil" could provide an alternative solution that is more than 5% cheaper, the entire project will be deemed a failure. This applies to both transferring and balancing.

After several of our team members tried to find a more optimal solution, none were available.

5. For 100 different manifest files, if the user requests for balancing, the outbound file will accurately display a balanced ship every time.



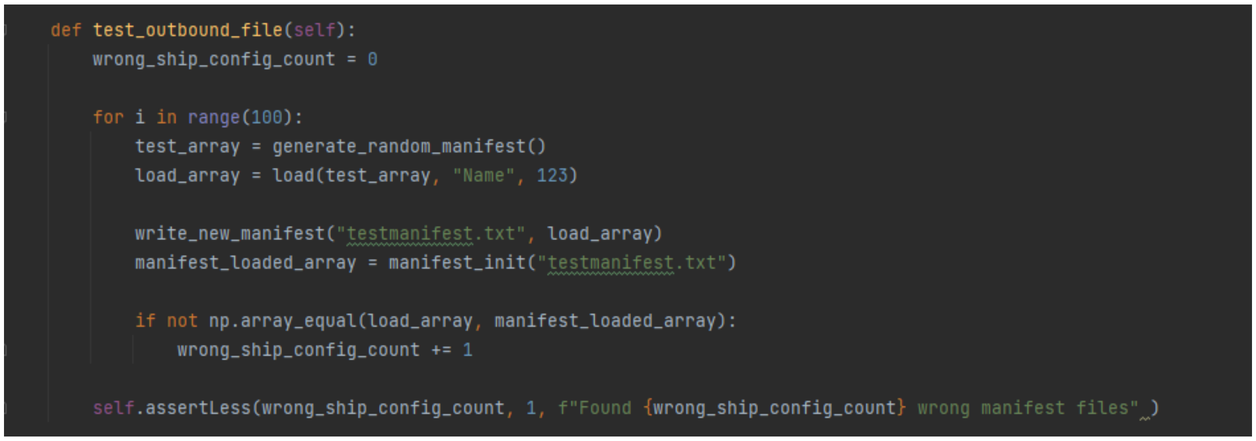
```
def test_balance(self):
    unbalance_count = 0

    for i in range(100):
            test_array = generate_random_manifest()
            balance_array = balance_ship(test_array)

            if check_if_ship_is_unbalanced(balance_array):
                unbalance_count += 1

    self.assertLess(unbalance_count,1, f"Found {unbalance_count} unbalanced ships")
```

This is a test to check that the outbound file will be accurate after balancing by

a. Creating a random ship configuration
b. Balancing that random ship
c. Checking to see if it is unbalanced

If it is unbalanced, we can see that the function did not properly balance the ship.

6. For 100 different manifest files, if the user requests for transfers, the outbound file will accurately display the resulting ship configuration every time.

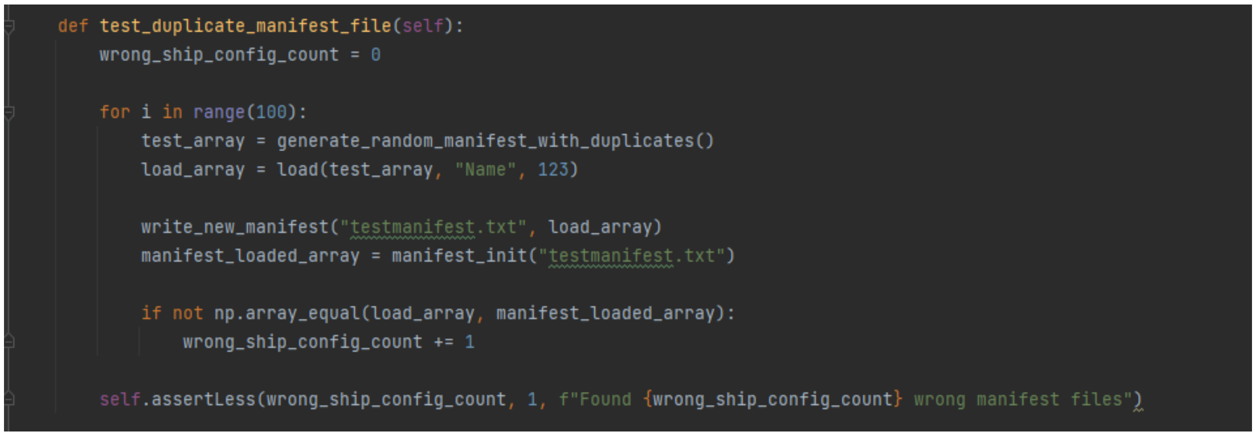

```python
def test_outbound_file(self):
    wrong_ship_config_count = 0

    for i in range(100):
        test_array = generate_random_manifest()
        load_array = load(test_array, "Name", 123)

        write_new_manifest("testmanifest.txt", load_array)
        manifest_loaded_array = manifest_init("testmanifest.txt")

        if not np.array_equal(load_array, manifest_loaded_array):
            wrong_ship_config_count += 1

    self.assertLess(wrong_ship_config_count, 1, f"Found {wrong_ship_config_count} wrong manifest files" )
```

This test is to check if an outbound file will be accurate by
a. Creating a test array
b. Loading a container named “Name” with weight 123
c. Wrote that output to a new manifest file.
d. Load the manifest file into an array
e. Check if the outbound array is the same as the modified load_array

7. For 100 different manifest files, if the user requests for transfers, there will not be a container left in the buffer zone.

We did not use the buffer zone except for instances of SIFT, so this test was not performable in our program.

8. For 100 manifest files containing varying amounts of duplications of container names, if a user request for a transfer, the program will be able to output 1. An optimal result and 2. An accurate Outbound file


```python
def test_duplicate_manifest_file(self):
    wrong_ship_config_count = 0

    for i in range(100):
        test_array = generate_random_manifest_with_duplicates()
        load_array = load(test_array, "Name", 123)

        write_new_manifest("testmanifest.txt", load_array)
        manifest_loaded_array = manifest_init("testmanifest.txt")

        if not np.array_equal(load_array, manifest_loaded_array):
            wrong_ship_config_count += 1

    self.assertLess(wrong_ship_config_count, 1, f"Found {wrong_ship_config_count} wrong manifest files")
```

This is a test that is similar to #6 but it has an emphasis on duplicates. Instead of creating a ship configuration with strictly unique container names, it seeks to create duplicates.

Maintenance Plan

While our team is unable to predict future events, we acknowledge the following scenarios where our software will need to be updated.

- In the event that the size of a container increases or decreases by any amount, our software representation of the ship bay will need to be updated in order to maintain the functions of onloading, offloading, and balancing.
- In the event that the size of the buffer zone increases or decreases, our software representation of the buffer zone will need to be updated in order to maintain the functions of onloading, offloading, and balancing.
- In the event that the size of the largest ship bay possible increases from 8 rows and/or 12 columns, our software representation of the ship bay will need to be updated in order to maintain the functions of onloading, offloading, and balancing.

In the event that any of the above events do occur, a fee of **$20,000** and a time period of **1 month** is required for each issue to be resolved.

Training and Documentation

Introduction:
Welcome to the Ship Container Loading/Unloading and Balancing Application! This user manual is designed to guide you through the process of using the application with ease, even if you have minimal knowledge of technology. The application allows you to load/unload containers onto a ship and balance the ship's weight distribution to ensure safe and efficient transport. Please follow the instructions provided in this manual to make the most of the application's features.

Getting Started:

1. Once the application has loaded, you will be presented with a simple and user-friendly interface that includes a few prompts and buttons to guide you through the process.
2. After initially launching the program, you will be prompted to sign in by pressing “CTRL + S” or by clicking on the “Actions” drop down menu and selecting the "Sign In (CTRL + S)" option.
3. You will then be prompted to start a new log file(Click the “Yes” button) or load an existing log file(Click the “No” button).

Creating a New Log File:

1. After signing in and choosing "Yes" to start a new log file, you will be asked if you want to append the current year to the log file that is named 'KeoghLongBeach.txt' by default. You will have the option to click a “Yes” or “No” button.
2. A prompt will appear that displays the name of the log file and a “Continue” button to proceed to uploading a new manifest file.
3. The application will automatically save all actions and comments made during the session to the log file.

Loading an Existing Log File:

1. After signing in and choosing "No" to load an existing log file, a prompt will appear that displays the log file that will be loaded, such as 'KeoghLongBeach{current_year}.txt'
2. You will then need to press the “Continue” button to proceed to uploading a new manifest file.

Uploading a Manifest File:

1. Ensure the manifest file in the required format (.TXT). The file should contain the necessary information about the bay of the incoming ship and the proper container information.
2. You will be prompted to "Please upload a Manifest file." Then the application will prompt you to browse for the manifest file.
3. You will see a "No file selected" message on the screen. To select a file, click on the "Select File" button below the message. This will open a file explorer window.
4. Navigate to the folder where your manifest file is stored and select it. The application will display the selected file's path with a message like "Selected file: {file_path}" on the screen.
5. Once you have selected a file, a "Continue" button will appear on the screen. Click on this button to proceed with the application. The program will then parse the file and create an array to represent the grid's state based on the container information from the uploaded manifest file.
6. You can now proceed with loading/unloading containers and balancing the ship using the uploaded manifest file as a reference.

Operation Selection:

1. After selecting a manifest file to upload, the application will prompt you to click a “Balance the ship” or “Start a transfer” button.
2. If the “Balance the ship” button is clicked, you will proceed to balancing the ship
3. If the ‘Start a transfer” button is clicked, you will have to click either the “Unload” or “Load” buttons that appear to proceed with unloading or loading the containers.

Loading Containers:

1. Ensure that you have followed instructions in the "Uploading a Manifest File" section and clicked the “Load” button.
2. When loading containers, you will be prompted to enter the container's name and weight. Fill in the required information and click "Submit".
3. You will now see an interface displaying the current layout of the ship's containers. The layout will be represented by a grid.
4. You will be prompted to "Please load the container to the indicated spot". The slot of where the container needs to be loaded will blink red.
5. Once the container is loaded, the “Submit” button can be pressed to continue loading more containers.
6. Once you have finished loading the containers, you can click the “Finished” button to proceed to loading/unloading more containers, or balancing the ship.

Unloading Containers:

1. Ensure that you have followed instructions in the "Uploading a Manifest File" section and clicked the “Unload” button.
2. You will now see an interface displaying the current layout of the ship's containers. The layout will be represented by a grid.
3. To unload a container, first, click on the container slot in the grid containing the container you wish to remove. Make sure that the chosen slot is a valid candidate for unloading.
4. A message will appear, indicating the name of the container that was selected. Then a “Back” button will appear to select another container or you may select the “Generate Order of Operations List” button to display a list of moves for unloading the selected containers.
5. After the order of operations list appears, you may press the “Proceed to Animation” button to visualize the movements through an animated grid.
6. The animation will show the ships layout with a grid and have alternating colors along the path that a container needs to move.

7. If there are more moves left, a "Next" button is displayed to proceed to the next animation. Otherwise, a "Finished" button is displayed to proceed to loading/unloading more containers, or balancing the ship.

Balancing the Ship:

1. Ensure that you have followed instructions in the "Uploading a Manifest File" section and clicked the “Balance the ship” button.
2. The application will analyze the ship's weight distribution and calculate the necessary movements to balance the ship.
3. After the calculation is complete, you will be presented with a list of operations detailing the movements required to balance the ship. Review this list to understand the required steps.
4. Click the "Proceed to Animation" button to visualize the balancing process. The grid will display the movements with a red highlight on the containers being moved.
5. If there are multiple steps in the balancing process, click the "Next" button to progress through each step. Once all steps have been completed, click the "Finished" button to display the options to load/unload more containers, or balance the ship.

Additional Features:

1. You can sign in to the application while it is running by pressing "CTRL + S" or click "Actions" on the menu bar and select "Sign In (CTRL + S)". Enter your full name(FirstName + [SPACE] + LastName) and click "Submit".
2. You can also add comments to the log file at any point in the application. To do so, press "CTRL + U" or click "Actions" on the menu bar and select "Add comment (CTRL + U)". Type your comment in the text box and click "Submit".

Troubleshooting:

If you encounter any issues while using the application, try the following steps:

1. Make sure you have entered the container information correctly when loading containers.

2. Ensure that you have selected a valid cell on the grid for loading/unloading containers.
3. Double-check the list of operations when balancing the ship to ensure you have followed the steps correctly.
4. Restart the application if any unexpected errors occur.

Conclusion:
This user manual should provide you with all the necessary information to seamlessly use the Ship Container Loading/Unloading and Balancing Application. With this guide, you should be able to load/unload containers and balance the ship effectively.

https://github.com/Bchea99/CS179M_AIMIGOS