Operating Room Logistics Robot

Team I: Critical Design Review Report

Tanmay Agarwal
Siddharth Ghodasara
Roman Kaufman
Robert Kim
Jinkai Qiu
Gaurav Sethia

Sponsors: Prof. Zackory Erickson, Prof. Jeff Ichnowski

May 2, 2024

Carnegie
Mellon
University

Abstract

Hospital operating rooms (ORs) are high-stakes environments where efficiency and accuracy are paramount. Challenges in managing the logistics of surgical inventories can lead to delays and increased stress among medical staff. In response, the team ORB Robotics has developed an autonomous robotic system specifically designed to enhance OR logistics. This mobile robotic system aims to address the critical inefficiencies in inventory management, starting with the retrieval and tracking of surgical items. Tested in a controlled environment at Carnegie Mellon University's AI Makerspace, the robot demonstrated its ability to navigate around an OR setup, accurately identify and handle medical supplies, and provide users with the real-time inventory knowledge. The next phase of development will focus on refining the system's reliability and robustness, developing grasping pipeline to eliminate the need for infrastructural changes, and expanding its capabilities to count the quantity of medical supplies using vision algorithms.

Contents

1 Project Description

In the dynamic and demanding environment of hospital operating rooms (ORs), where every second counts and precision is paramount, the logistical handling of medical supplies remains a critical challenge. Current systems, entrenched in outdated and manual processes, often lead to inefficiencies, with medical staff struggling to locate essential supplies quickly during surgeries. This not only hampers the efficiency of medical procedures but also poses significant risks to patient safety due to potential breaches in sterile conditions when staff must leave the OR to retrieve items [3].

Moreover, the financial implications are profound, as hospitals face substantial losses from expired and misplaced inventory amidst the disorder of conventional supply management[17]. These challenges are compounded by a persistent shortage of nursing staff [2], further straining the ability of healthcare facilities to maintain optimal operational flow during surgical procedures.

The proposed solution, the Operating Room Bot (ORB), is designed to address these inefficiencies by automating the delivery and management of medical supplies within the OR. ORB seeks to enhance the operational efficiency of healthcare services by ensuring that medical supplies are readily available at the point of use and accurately tracked, thus eliminating the need for staff to leave the OR during critical moments. This not only safeguards the sterile environment but also allows healthcare professionals to remain focused on patient care.

ORB aims to create an integration of robotic logistics solutions within the OR, ensuring a streamlined, efficient, and safer OR environment. This initiative promises to modernize OR operations by reducing wastage, lowering costs, and most importantly, enhancing patient care outcomes by maintaining stringent safety standards. Through ORB, we envision a future where technology and healthcare professionals collaborate seamlessly, setting new standards in patient care and operational efficiency in hospitals.

2 Use Cases

In order to effectively address the complex logistics challenges within operating rooms (ORs), the ORB system has been developed to enhance both the inspection and delivery of medical supplies. This initiative is illustrated through two primary use cases: inspection and delivery, each narrated from the unique perspectives of frontline medical staff at UPMC.

In the inspection use case, the focus is on the daily operations overseen by a UPMC nurse. The OR corridors are a hub of activity, where the ORB system plays a pivotal role. As depicted in Figure 2, the system navigates the aisles, its advanced sensors actively scanning each bin and shelf. It accurately logs the quantities and locations of critical supplies like syringes, gauze, and sutures into the hospital's inventory management software. During his shift, Nurse Billy observes the ORB identifying a low stock of IV catheters, essential for the day's procedures. The system efficiently flags this inventory shortage and automatically initiates a restock request, thus ensuring continuous availability of necessary medical items.

The delivery use case, narrated from the viewpoint of Dr. Michael J. Singh, a chief surgeon at UPMC, highlights a scenario during a complex surgery. When an unexpected need for a specific hemostatic agent arises, Dr. Singh utilizes the ORB system to promptly request the item. The corresponding steps shown in Figure 3 detail how the ORB receives the command, swiftly retrieves the item from the supply room, and navigates through the hospital to deliver it directly to the OR. This process

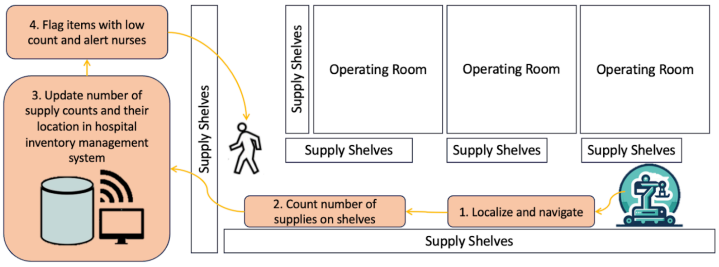


Figure 1: Graphical Representation of inspection function of ORB Robot

not only minimizes disruptions in the surgical workflow but also maintains the sterility and safety of the delivered supplies.

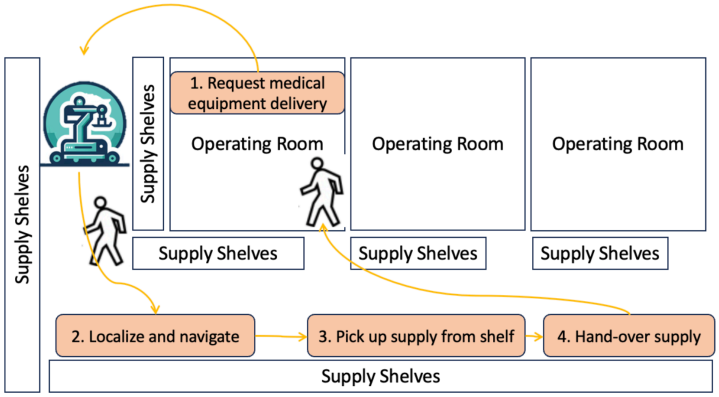


Figure 2: Graphical Representation of delivery function of ORB Robot

Both use cases demonstrate the ORB system's crucial role in streamlining OR operations by automating the tasks of inventory inspection and high-priority medical supply delivery. This integration of advanced robotics within the OR setting significantly enhances the efficiency of medical procedures, reduces the workload on medical staff, and contributes to improved patient care outcomes. With ORB, UPMC has embraced a future where technology and healthcare seamlessly converge to foster a more efficient and safer surgical environment.

3 System-Level Requirements

Upon conducting a thorough needs analysis and engaging with stakeholders and potential customers, we formulated a three-tier objective tree for the ORB, as illustrated in Figure 13 within the Appendix. These requirements were gathered through consultations with our sponsors and medical professionals at the University of Pittsburgh Medical Center (UPMC), taking into consideration available resources and time constraints. It's important to note that although the project scope extends beyond the Master of Robotic System Development (MRSD) program itself, our mandatory functional requirements derive from a subset of our primary objectives. No changes to the requirements have been made since the Preliminary Design Review (PDR).

3.1 Mandatory System-Level Requirements

The mandatory requirements are further split into Functional & Performance Requirements in Table 1 and Non-Functional Requirements in Table 2. These are based on the scope of the project for MRSD. Each requirement is detailed to ensure clear understanding and traceability from the need analysis phase through to implementation and testing.

This structured approach ensures that ORB's development aligns with the critical needs of the operating room environment, optimizing both the design and functionality to enhance operational efficiency and support medical staff effectively.

Table 1: Mandatory Functional and Performance Requirements

Functional	Performance
M.F.1 The system shall receive inputs from the users.	**M.P.1** The system will receive inputs from the user in less than 5 seconds.
M.F.2 The system shall localize in the pre-mapped environment.	**M.P.2** The system will localize in the environment within an error of 5cm.
M.F.3 The system shall collect medical supplies.	**M.P.3** The system will collect objects with success rate of more than 80% accuracy.
M.F.4 The system shall plan and navigate to its destination.	**M.P.4** The system will plan a global path in 5 seconds and navigate to its destination within maximum speed of 0.8m/s.
M.F.5 The system shall deliver medical supplies to operating room.	**M.P.5** The system will deliver supply to operating room within 5 minutes.
M.F.6 The system shall inspect supply inventory to estimate the quantity of each medical supply	**M.P.6** The system will inspect 10 supplies with accuracy of 2 item counts for LOW segment
M.F.7 The system shall update the inventory	**M.P.7** Within a 15 minute sweep of the supply the system will have inventory knowledge in keeping with M.P.6 above.

Table 2: Mandatory Non-Functional Requirements

Requirements	
M.N.1	The system will perform rapid retrieval of desired objects.
M.N.2	The system will adhere to all relevant standards pertaining to medical robotic systems.
M.N.3	The system will have a modular design: for hardware, software, and all things in between.
M.N.4	The system will be aesthetic.
M.N.5	The system will detect malfunctions and errors so as to notify user
M.N.6	The system will remain available in times of need.

Table 3: Desirable Functional Requirements

Functional	Performance
D.F.1 The system shall detect and avoid dynamic obstacles.	**D.P.2** The system will detect and avoid dynamic obstacles with more than 75% accuracy.
D.F.2 The system shall store multiple objects on the robot.	**D.P.2** The system will store at least 2 objects on the robot.
D.F.3 The system shall have manual motion override mode.	**D.P.3** The system will switch to a manual motion override mode within 1 second of force input.
D.F.4 The system shall provide analysis on medical supply logistics.	**D.P.3** The system will provide analysis on medical supply logistics every 12 hours.

3.2 Mandatory Non-Function Requirements

3.3 Desired Functional Requirements

3.4 Desired Non-Functional Requirements

Table 4: Desirable Non Functional Requirements

Requirements	
D.N.1	The system will have a comparable cost to similar systems in the market
D.N.2	The system will produce low noise
D.N.3	The system will be designed such that it can be maintained easily.

4 Functional Architecture

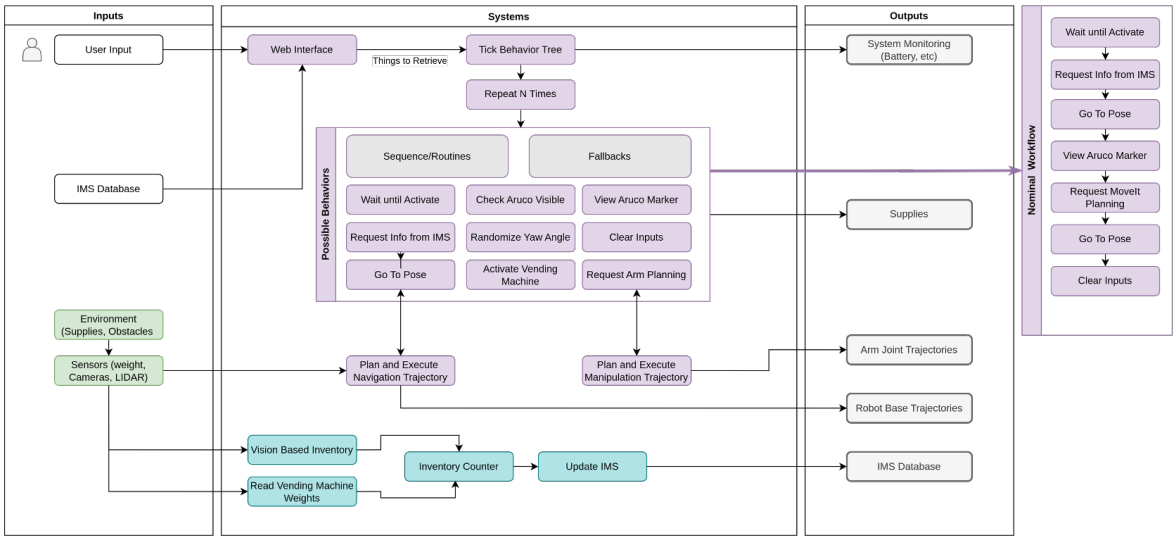


Figure 3: Functional Architecture

The updated functional architecture diagram in Figure 3 provides a comprehensive overview of our system's core functions, mapping the progression of information from initial input through to the final output. These diagrams are structured into three main segments, representing the inputs, the system's internal processes, and the outputs. The placement of these elements along the X-axis indicates the sequence of operations, with time represented spatially from left to right. When two processes are depicted one above the other, it signifies that they occur simultaneously, emphasizing the system's ability to handle multiple functions efficiently.

- **Environment**: Data from the environment is captured through various sensors such as cameras and LiDAR, providing crucial real-time information about the operating room's state.
- **User Inputs**: These are commands provided by users through an interface, directing the system's actions based on specific medical needs.
- **Outputs**: The system outputs actions for robot motion and the real-time OR inventory knowledge.

Our system comprises two main subsystems: the retrieval subsystem and the inspection subsystem.

- **Inspection Subsystem**: This subsystem utilizes data from sensors to inspect and assess the condition and availability of medical supplies. This critical information is then relayed to the inventory management system (IMS), which is enhanced in the updated architecture to include more sophisticated analytics and tracking capabilities.
- **Delivery Subsystem**: Following user commands, this subsystem engages to accurately locate and navigate to the required medical items. It collects the items and securely transports them back to the user, ensuring that the right supplies are delivered promptly and efficiently.

It is critical to note that the delivery subsystem and the inspection system, along with many other functionalities, are grouped together inside **Possible Behaviors**, indicating they are all independent behaviors that can be called by the behavior tree when needed. For further functionalities in the future and fallback behaviors, we will only need to modify the behavior tree definition, keeping these behaviors the same.

The output of ORB is the supply delivered to the user, details and analysis on the operating room inventory, and status of the system as well. Overall, the functional architecture integrates these subsystems to ensure efficient processing of information and execution of tasks, particularly in managing and dispensing medical supplies.

5 Cyberphysical Architecture

The cyberphysical architecture of the system delineates a refined implementation strategy, showcasing a interaction between system modules that orchestrate data flow and task execution. It is illustrated in Figure 4

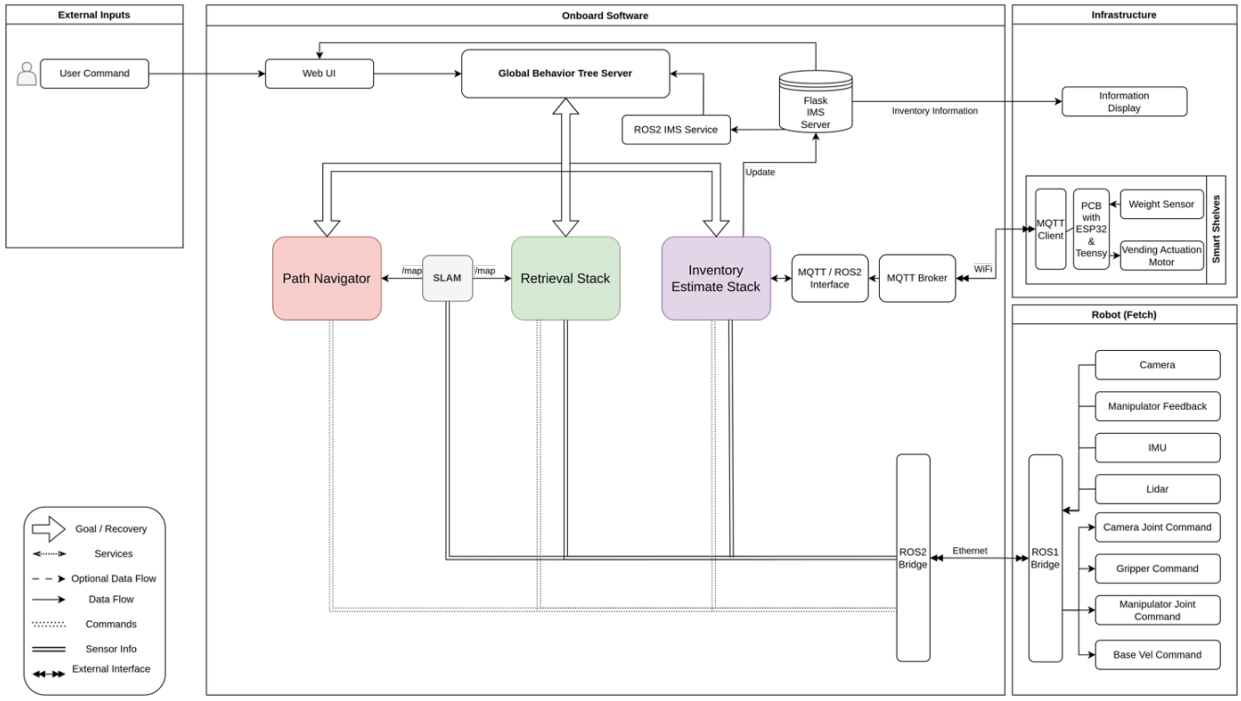


Figure 4: Cyberphysical architecture

The cyberphysical architecture is divided into four parts, representing external inputs, onboard software, infrastructures, and the robot itself. The entire process starts with an user command through the Web UI, which will be passed to the Global Behavior Tree Manager for task level interpretation, which will then be decomposed to individual actions for **Path Navigator**, **Retrieval Stack**, and **Inventory Estimation Stack**.

The **Inventory Estimation Stack** maintains a wireless communication with the infrastructure. It will both actuate vending machine and report latest inventory level estimation.

All three sub-components communicate with the robot (ROS1 [16]), through a ROS1-ROS2 [10, 8, 12] dynamic bridge directly over the network. A Wired connection makes sure minimal latency and reliable behaviors.

5.1 Sensing

We use necessary and built-in sensor topics from Fetch. This includes Camera, Lidar, IMU, Joint Pose, amongst others. We did not write the sensor drivers by ourselves.

5.2 Navigation

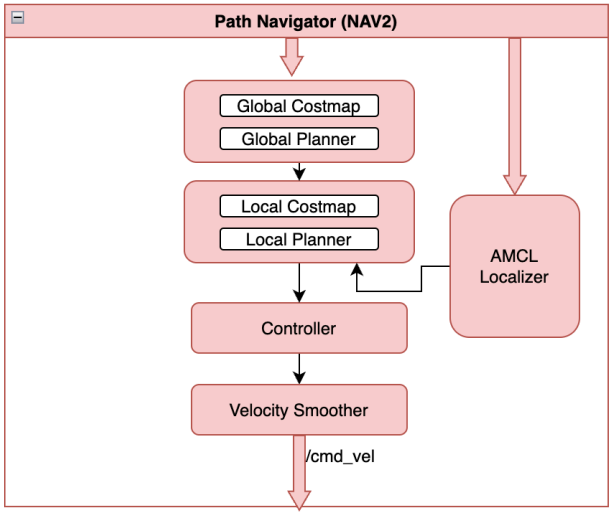


Figure 5: Navigation Architecture

Our navigation system utilizes Nav2 stack to provide map based 2D navigation. We use primarily a 2D lidar data and localize through AMCL within a predefined map. Mapping primarily utilizes slam toolbox which is an implementation of graph based SLAM.

5.3 Retrieval Stack

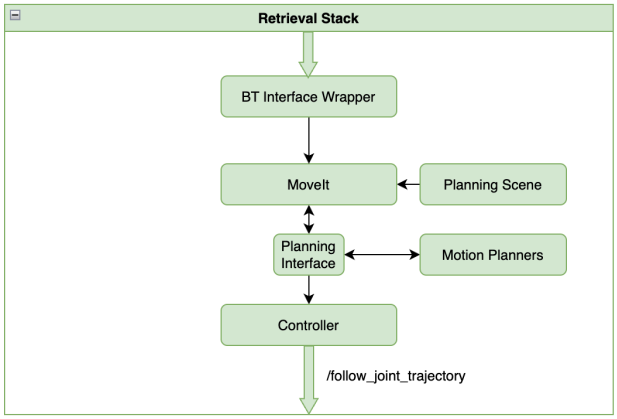


Figure 6: Navigation Architecture

Our retrieval stack utilizes Moveit2 stack to provide smooth trajectories. As of now we do not actively perceive obstacles. Instead, we primarily use the trajectory generation capabilies of Moveit2.

5.4 Inventory Estimation Stack

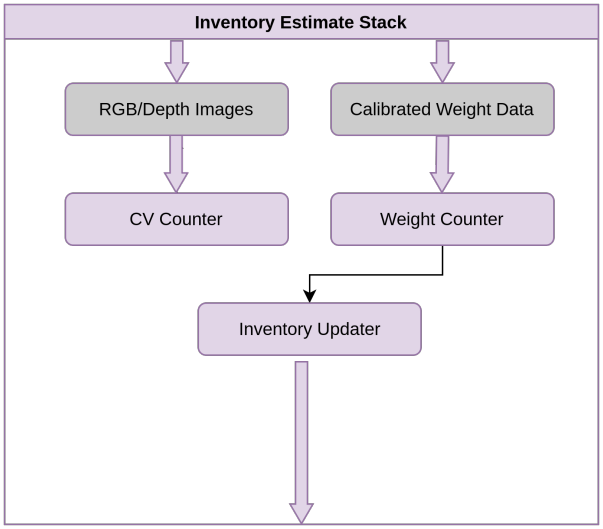


Figure 7: Navigation Architecture

Our Inventory stack utilizes both on-robot camera and weight sensor augmented infrastructure. We intend to fuse the estimates as well as raw data to accrately determine inventory movement. For visual inspection we tested large object detection [13] and segmentation networks [6] and finetuned YOLOv8 [14]

6 Current System Status

6.1 Targeted Requirements

For the Spring Semester, our team focused on achieving the following seven performance requirements.5.

Table 5: Targeted Performance Requirements for Spring Semester

Requirement	Status	Subsystem
M.P.1 The system will receive inputs from the user in less than 5 seconds.	Passed	Interface
M.P.2 The system will localize in the environment within an error of 5cm.	Passed	Navigation
M.P.3 The system will collect objects with success rate of more than 80% accuracy.	Passed	Manipulation, Smart Shelf
M.P.4 The system will plan a global path in 5 seconds and navigate to its destination within maximum speed of 0.8m/s.	Passed	Navigation
M.P.5 The system will deliver supply to operating room within 5 minutes.	Passed	Navigation
M.P.6 The system will inspect 10 supplies with accuracy of 2 item counts for LOW segment	Passed	Inspection
M.P.7 Within a 15 minute sweep of the supply the system will have inventory knowledge in keeping with M.P.6 above.	Passed	Inspection

6.2 Overall System Depiction

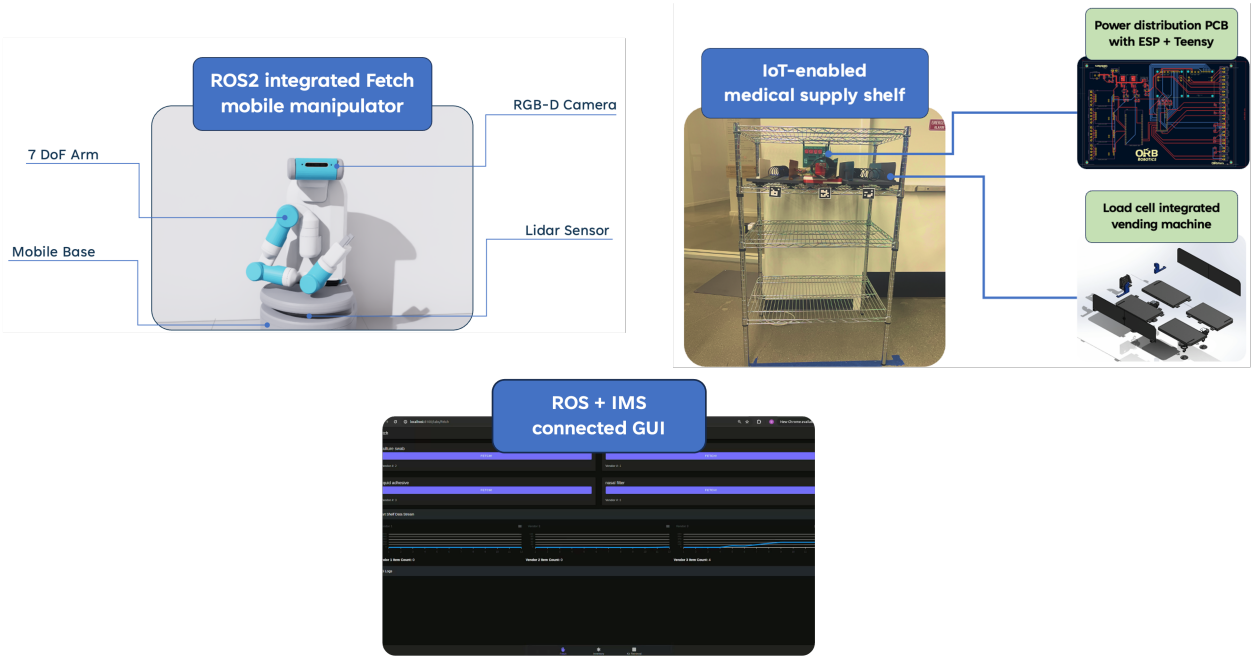


Figure 8: Overall System Depiction

The current system consists of three major components designed to optimize the handling and management of medical supplies within a hospital setting: the Fetch mobile robot, the smart shelf, and the graphical user interface (GUI). Each element is interconnected, creating a seamless operation flow that enhances the efficiency and accuracy of medical inventory management.

Fetch Mobile Manpulator: The Fetch robot, equipped with a 7 Degrees of Freedom (DoF) arm and a mobile base, is central to the system's operation. It incorporates an ROS2 framework running within a Docker container for modularity and scalability. The robot is outfitted with a Lidar sensor for navigation and an RGB-D camera for object recognition and manipulation tasks. This integration allows the Fetch robot to perform complex navigation and retrieval tasks autonomously.

Smart Shelf: The smart shelf is an adaptation of conventional OR shelving, equipped with integrated vending machines in each row to dispense specific medical items upon command. This system utilizes ESP32 microcontroller to communicate via MQTT protocol, allowing for precise control over the dispensing mechanisms. The use of a PID controller with Teensy microcontroller ensures accurate operation of the motors based on the commands received from the Fetch robot, effectively minimizing errors and delays.

User Interface: The GUI serves as the human-machine interface, connecting users directly to the ROS2 backend and the central inventory management system (IMS). It provides intuitive controls for requesting medical items, alongside dynamic displays of inventory levels and item counts within the OR. Additionally, the GUI offers real-time logs from ROS, allowing users to monitor the robot's activities and system status effectively. This interface ensures that all user interactions are straightforward, enhancing user experience and system usability.

With all components integrated, the basic operational flow is as illustrated in the diagram below.

Figure 9: ORB System's Operational Flow

6.3 Subsystem Descriptions

6.3.1 Behavior Tree

The Behavior Tree (BT) subsystem in Figure 10 functions as the central operational control within our project, integrated into the NAV2 package. This subsystem is crucial for coordinating the interaction between various components of the system. The key feature of the BT is its ability to implement fallback mechanisms that enhance the system's adaptability to real-time deviations. Specifically, it includes mechanisms for dynamic obstacle detection and path re-planning, which allow the robot to navigate around unexpected obstacles not present in the static environment map. This capability ensures reliable navigation and operational efficiency in the dynamically changing settings of a hospital [1].

The fallback mechanism within our Behavior Tree (BT) is also designed to handle deviations from the goal position, ensuring that the Fetch robot can reliably and safely plan its arm trajectory when interacting with the smart shelf. For example, if the Fetch robot's RGB camera does not detect an ArUco marker, which is used for locating the items, the BT initiates a fallback procedure. This procedure involves the mobile base performing a yaw rotation to search for ArUco markers. Once the markers are detected, the system guides the robot to a position where its arm can accurately reach and interact with the specified shelf. This dynamic response capability is essential for maintaining operational accuracy and safety, especially in environments where initial conditions may change or be less controlled.

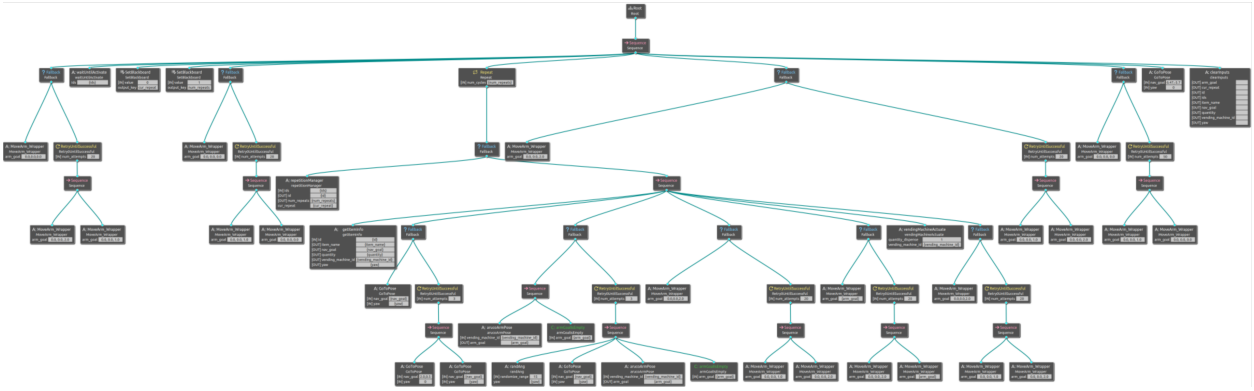


Figure 10: Behavior Tree of ORB System

6.3.2 Graphical User Interface

The Graphical User Interface (GUI) subsystem serves as the primary interface point for users within the medical supply retrieval system. It is integrated with the central Inventory Management System (IMS), enabling immediate access to up-to-date inventory data and item specifics. Through an HTTP ROS Bridge connection, the GUI facilitates direct command and control interactions with the Fetch robot's ROS2 environment, allowing users to initiate retrieval tasks and monitor progress in real-time. Additionally, the GUI displays real-time sensor data, including load cell measurements from the smart shelf, and provides access to ROS logs, which assist in monitoring the robot's current activities. This comprehensive interface ensures users can easily manage and interact with the system.

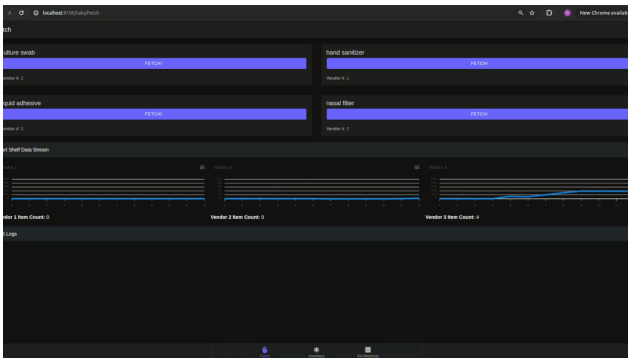


Figure 11: Graphical User Interface for the ORB System

6.3.3 Navigation

The Navigation subsystem leverages the NAV2 package [11, 15, 9], adapted for integration with the Fetch robot, which operates primarily on ROS. To bridge this gap, we implemented ROS - ROS2 Bridge package to implement communication between the navigation stack and the robot's control systems. This setup allows NAV2 to manage navigation commands, receive sensor data like odometry from ROS topics, and send goal poses back to Fetch's navigation stack. This integration ensures that Fetch can utilize the advanced features of NAV2, such as dynamic collision avoidance and sophisticated path planning, within its existing ROS framework.

Initially, the navigation stack was developed and validated using the Isaac Sim simulation environment as depicted in Figure 12. However, during real-world testing, we observed some deviations from the robot's desired pose to the actual pose when attempting to align with the smart shelves. To address this, we implemented a fallback mechanism within NAV2 framework, which allowed Fetch to re-localize and replan its route to the goal pose dynamically.

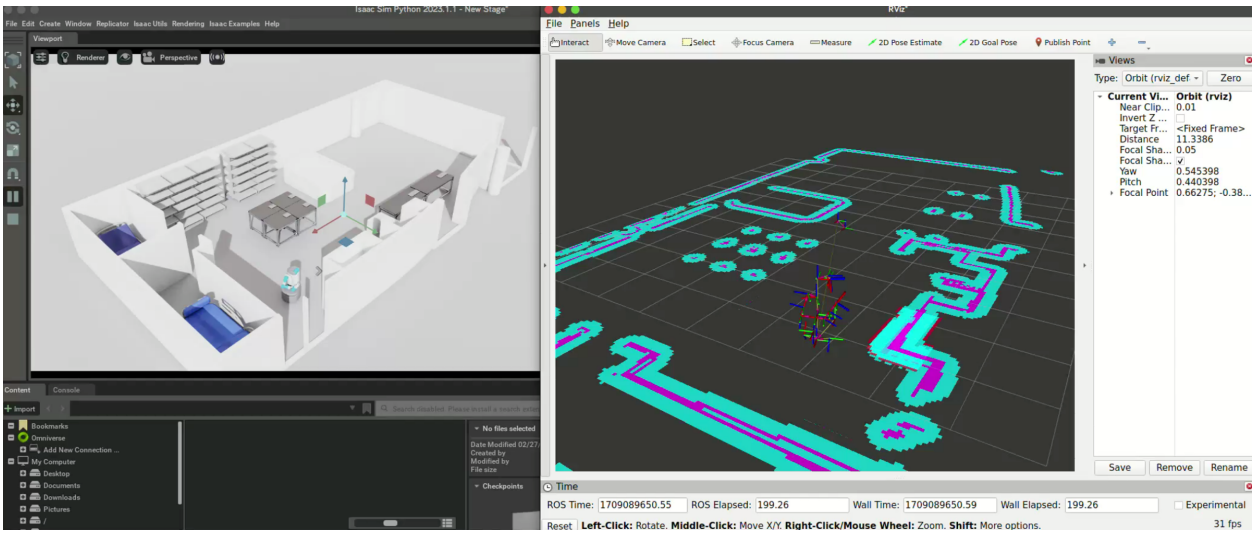


Figure 12: Fetch NAV2 Integration Testing in Isaac Sim (left) and Visualization in Rviz (right)

6.3.4 Manipulation

The Manipulation subsystem is primarily built around MoveIt 2 [5] to handle the precise control needed for Fetch's arm movements. Initially developed and simulated using Isaac Sim, this subsystem allows the robot to execute complex manipulative tasks required in medical settings. However, we encountered significant latency issues during the early stages of real-world deployment due to the overhead introduced by bridging MoveIt 2 (ROS2) with the Fetch robot's native joint position controller. This led to packet drops and inconsistencies in arm motion control. We also tested the entire pipeline on Isaac Sim [4].

To resolve these challenges, we developed a custom MoveIt 2 wrapper, which significantly reduced the data exchange across the ROS to ROS2 bridge, thus minimizing latency and improving the reliability of data transmission. The manipulation strategy employed involves using a Cartesian path for the robot's arm trajectory. This method ensures that the orientation of the bucket attached to the end effector remains constant throughout the motion, preventing the contents from being dislodged or spilled.

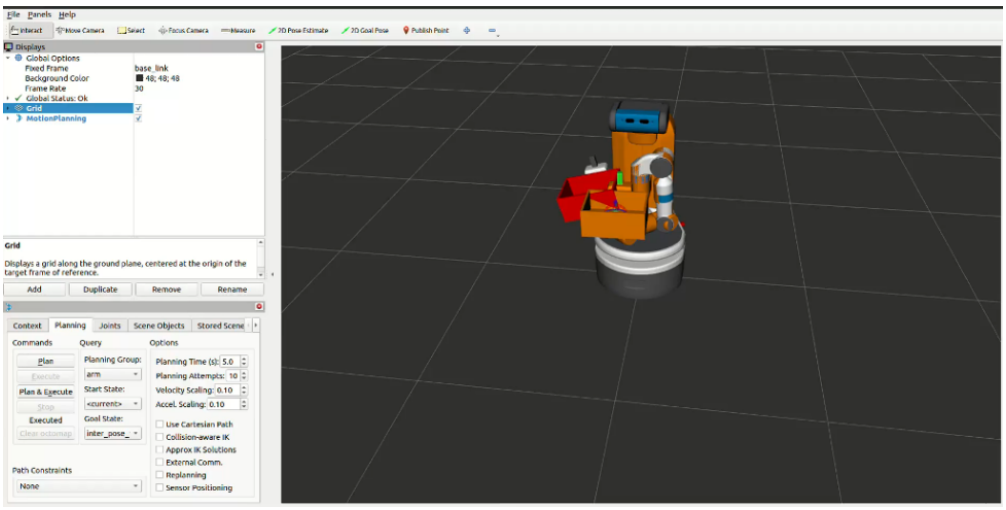


Figure 13: Visualization of manipulator motion in Rviz

6.3.5 Smart Shelf

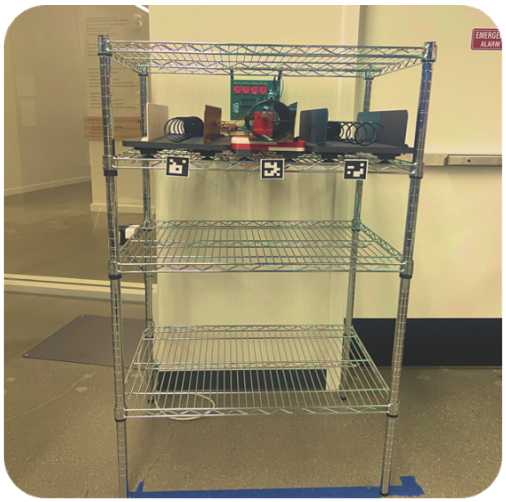


Figure 14: Smart Shelf Subsystem in AI Makerspace Testbed

The Smart Shelf subsystem integrates with the Fetch robot system through a combination of ROS2 communications, MQTT protocol [7], and direct microcontroller management. The shelf is equipped with vending machines, controlled by a power distribution PCB that connects ESP32 and Teensy microcontrollers. The ESP32, operating as an MQTT client, receives commands from the Fetch robot to dispense items, which it then relays to the Teensy microcontroller. The Teensy uses these commands to operate motors via PID control that drive the vending machine's spiral coils as shown in Figure 15, dispensing the requested items.

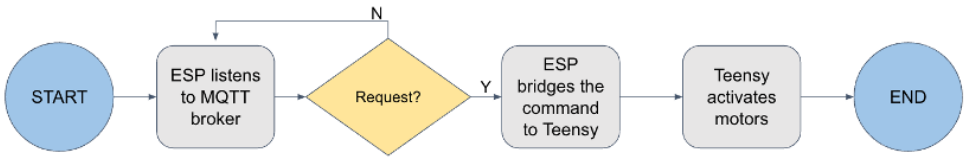


Figure 15: Flowchart of Item Dispensing Operation

Each vending machine's support structure incorporates calibrated half-bridge load cells, allowing for real-time weight measurement of the items on the shelf. This data is transmitted back to the Fetch robot's ROS2 system via MQTT, enabling the GUI linked to the central Inventory Management System to calculate and display the current inventory on each vending machine based on item weights. This setup provides essential inventory management and inspection capabilities within the system.

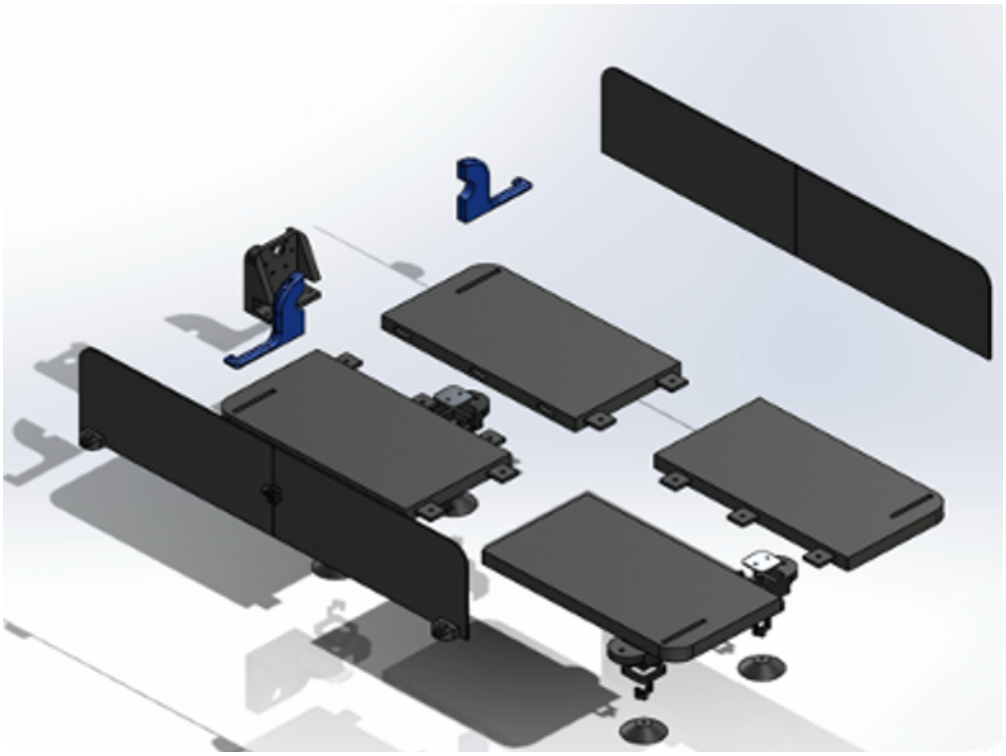


Figure 16: Exploded View of Vending Machine Platform CAD Model

6.4 Modeling, Analysis and Testing

We utilized Isaac Sim to validate the integration and functionality of our ROS2 subsystems, including navigation, manipulation, and the Behavior Tree (BT). Isaac Sim provided a digital twin environment where we could extensively test the Fetch robot's movements and interaction mechanisms. This simulation was pivotal for refining the robot's path planning algorithms and arm manipulation sequences, ensuring they were precise and reliable.

Following the individual component tests, we conducted integration testing in a real-world environment at the AI Makerspace to evaluate the overall system performance. This involved running the robot for a total of 50 operational cycles, during which it successfully completed the task of collecting medical items upon user command 42 times. The eight unsuccessful attempts included three validation demonstrations for System Validation Day (SVD), where specific issues were identified and subsequently addressed to enhance system reliability and performance.

6.5 SVD Performance Evaluation

The set of validation criteria that were tested in the SVD and SVD Encore are shown in Table 6. Our robot performed the set of tasks while meeting the desired system requirements. While the HRI fallback rate was being met, there were issues in the speech transcription accuracy of the HRI subsystem during SVD. This caused the HRI system to misinterpret the user's speech command.

Table 6: Targeted Requirements for Spring Semester

Procedure	Success Criteria
User presses a button within the GUI for the delivery of an item or a surgical kit	The system delivers a medical item within 5 minutes while traveling a distance of 15m The system delivers a different item after completing the previous task
The user monitors the GUI to check the total quantity of each item in an OR inventory	The system provides real-time inventory information The system counts the number of items present on vending shelves via weight sensors with an accuracy of +/- 2 items.

6.6 Strong/ Weak Points

In reviewing the performance of the ORB system, we have identified key strengths and areas where improvements are necessary:

Strengths:

- **Speed**: The system was able to execute tasks within 2.5 minutes.

Weaknesses:

- **Precision**: The system currently struggles with precision in some critical areas, particularly in localization. This imprecision occasionally results in the robot's inability to align correctly with the smart shelf, impacting the Fetch robot's ability to plan and execute arm trajectories effectively.

- **Reliability**: The main area requiring improvement is the system's reliability. Issues primarily arise from network latencies that affect communication between the system's components and the central server. These delays sometimes disrupt the system's real-time operational capabilities, leading to errors in task execution. Enhanced network protocols and optimized data handling strategies are needed to bolster the system's robustness and ensure consistent performance under all operational conditions.

7 Project Management

As previously discussed, our work plan includes two work iterations and a possible third iteration, depending on the schedule. The first iteration has been completed successfully. It involved inspection and retrieval tasks within a testbed environment established in the AI Makerspace. This environment featured testbed shelves equipped with weight sensors and actuators. In the fall semester, the second iteration of our work plan will be executed. This iteration involves carrying out the tasks of inspection and retrieval in an environment that simulates a hospital setting. In this setup, the testbed shelves will no longer be equipped with actuators (vending machines). The work breakdown structure, which details the second iteration further, is shown in Figure 17.

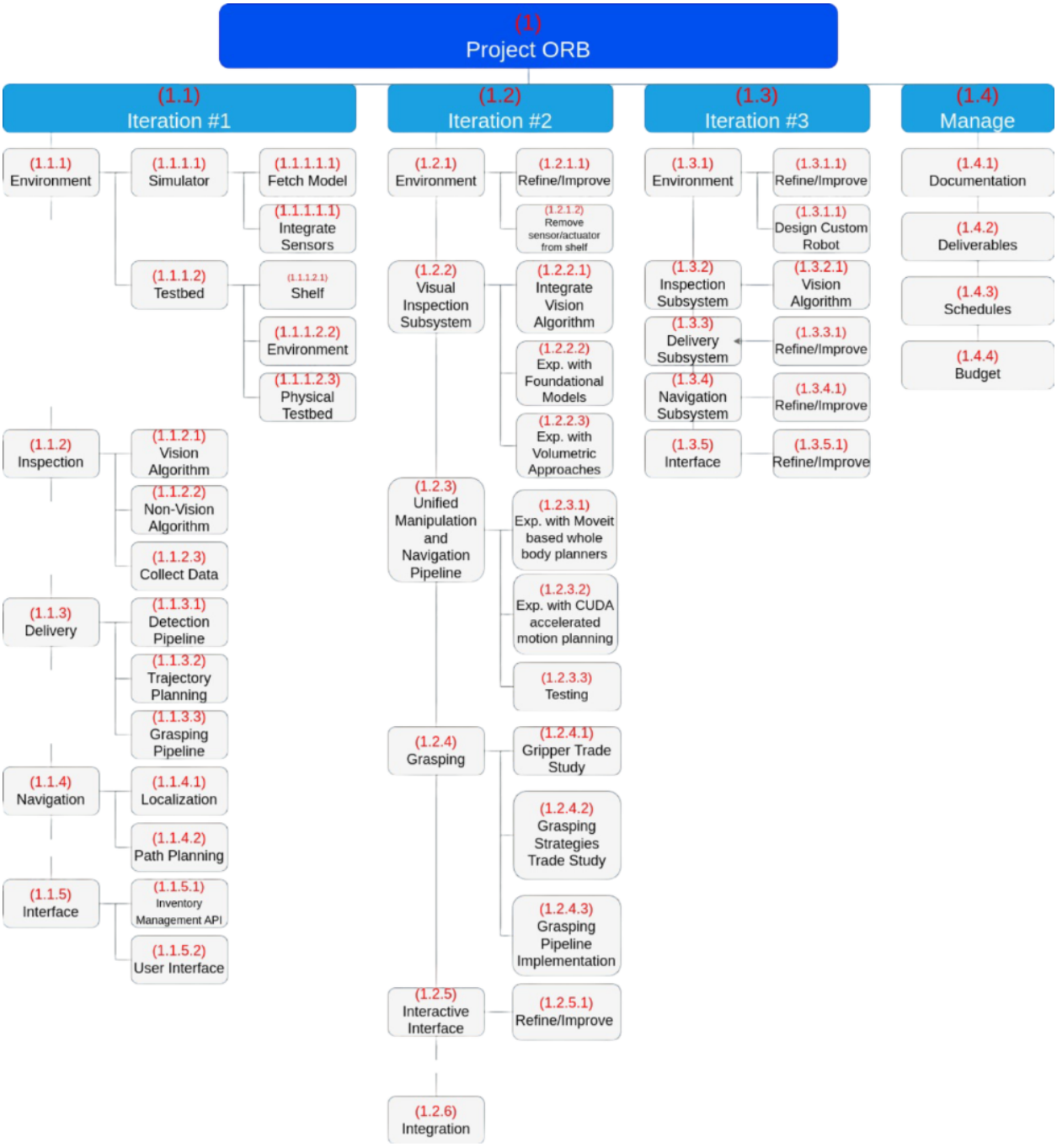


Figure 17: Work Breakdown Structure

7.1 Work Breakdown Structure Explained

We've expanded upon our initial work breakdown structure for the second iteration to align with our objectives. While maintaining the inspection (weight-sensor-based), pickup, and navigation capabilities, we're introducing visual inspection to complement the existing method. Additionally, we're developing a grasping pipeline to enable the robot to grasp items from shelves, as opposed to relying solely on items dispensed by a vending machine. Although we have a user-friendly GUI for interaction,

recognizing the constraints faced by medical staff, we're enhancing the interface further by introducing an interactive system for easier interaction.

Recognizing a decoupling issue between navigation and manipulation leading to compounded imprecisions, we're addressing this by developing a whole-body controller to tightly couple navigation and manipulation, aiming to reduce overall errors.

To address schedule delays observed this semester, we're separating integration and reliability testing into a distinct work package rather than including them within each individual work package.

7.2 Schedule

7.2.1 Biweeklies, Milestones, and Demos

The major milestones for our project, as shown in Table 7, have been strategically planned. Each internal milestone corresponds to the development or integration of specific subsystems. The project is divided into five main parts: **Visual Inspection, Whole Body Controller, Grasping, Interactive Interface, and Integration and Reliability Testing**. The objectives for some of these milestones during the project are tentatively outlined as follows:

Bi-weekly 1 - Generate synthetic data and gather real-world data for visual inspection purposes. This data will subsequently undergo evaluation to assess the effectiveness of different visual inspection methods, including YOLO-v8, foundational models for counting, and volumetric approaches. We parallelly start working on whole-body controllers and grasping pipelines.

Bi-weekly 2 - Over the next two weeks, our focus will be on the development of the whole-body controller. Our strategy involves initial testing with Moveit-based whole-body controllers to ensure their reliability before integration. Additionally, we will explore alternative options, such as CUDA-parallelized whole-body controllers, aiming to enhance the overall speed of the system significantly.

Bi-weekly 3 - Based on the findings from the trade study conducted on gripper and grasping approaches, we will determine the most suitable grasping approach for our specific use case. Subsequently, we will proceed to implement the chosen grasping approach and integrate it seamlessly with the existing system.

System Development Review - A demonstration showcasing all functionalities of the fully integrated system will be conducted using a limited selection of objects, with a focus on grasping a single medical item. The system's capabilities will include navigating the environment autonomously, utilizing the manipulator to grasp the specified item, and updating the central data store based on visual and weight-sensing inputs. This entire process should be executed at least once without the need for manual intervention.

For interface purposes, the current GUI will be utilized during this phase. This comprehensive demonstration serves to provide valuable insights into any potential limitations of the system. Should there be any slack in the schedule, adjustments to our objectives can be made accordingly, based on the observations and feedback gathered from the demonstration.

Bi-weekly 4 - Upgrade the interface to an interactive platform, departing from the current GUI-based interface. Prior to integration with the existing system, the interactive interface will undergo rigorous stress testing to ensure its robustness and reliability under various conditions.

Fall Validation Demonstration (FVD) -A functional demonstration of the fully integrated system, encompassing coupled navigation and manipulation through the whole-body controller, object grasping, and visual inspection, will be showcased. The detailed procedure for the Fall Verification Demonstration (FVD) is outlined in Table 8 below.

7.2.2 Timeline

You can view our Fall schedule displayed in a Gantt chart format in Figure 18. It mirrors what we presented in the CODR, as we're not planning to commit to anything new beyond what was already outlined in the CODR.

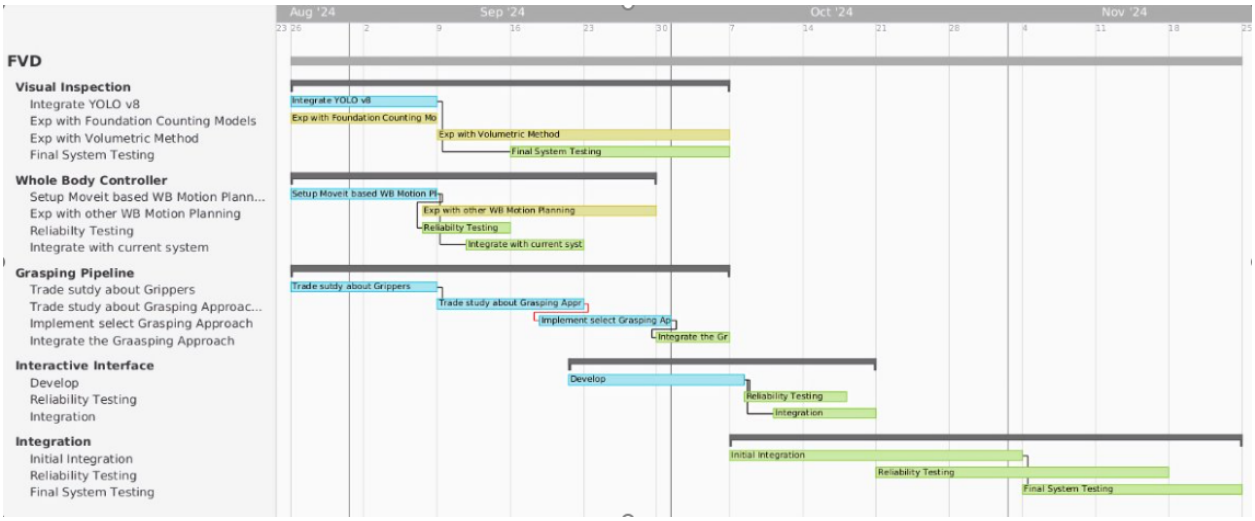


Figure 18: Fall 2024 Schedule and Milestones

7.3 Test Plan

For the fall semester, we have outlined two primary objectives: i) Implement Visual Inspection and Grasping functionalities and ii) Enhance the current system by integrating whole-body controllers and an interactive interface.

Additionally, we aim to improve the system's reliability, ensuring it operates effectively for at least 10 consecutive runs without requiring any manual intervention. This reliability enhancement is crucial for ensuring seamless operation and consistent performance across various scenarios.

7.3.1 Testing Activities

1. **Visual Inspection Accuracy**: Test the accuracy of our visual inspection system by varying:
    - Lightening conditions
    - Distance of the camera from the shelve
    - Shelf location
    - Location of items on the shelve
    - Combination of items kept close to each other
    - Number of items kept on the shelve
2. **Grasping Success Rate Test**: Test the success rate of grasping by varying

- Lightening conditions
- Number of items kept on the shelve
- Object orientation

3. **Latency and Reliability Tests**: We test the latency and reliability by varying
   - Distance between user and input source
   - Number of devices connected to the network
   - Compute utilization

7.3.2 Fall Semester Capability Milestones

Table 7: Fall Capability Milestone

Progress Review	System Capabilities
PR 7	**Visual Inspection**: We would be showcasing our vision based inspection system identifying and counting at-least 3 different classes of medical items on the shelf. The vision algorithm would be integrated with the existing inventory management system
PR 8	**Unified Manipulation and Navigation Pipeline**: This improvement aims to eliminate compounding errors by integrating manipulation and navigation as unified systems. Currently, our Behavior Tree (BT) links these pipelines, but our goal is to establish a whole-body motion planning subsystem capable of resolving these errors over time.
PR 9	**Grasping Pipeline**: We would be conducting trade studies on the gripper design and pipeline, following which we would be finalizing and integrating the best candidate from the studies.
PR 10	**Interactive Interface**: We plan to simplify the user interface to help clinicians order items easily
PR 11	**Integration**: Since we took a lot of time integrating and were even behind schedule, we are making sure that we allocate enough time to integrate.
PR 12	**Integration and Testing**: From our previous experience, we found that once we completed initial integration, a lot of unforeseen issues came up that took a lot of testing and debugging to resolve. Thus we have allocated extra time to conduct testing.

7.3.3 Fall Validation Demonstration

The test conditions are as follows:

- **Location**: AI Makerspace, Tepper
- **Equipment**: Fetch, Five different medical items, PC, Weight sensing shelf
- **Operating area**: Operating wing reconstructed in AI Makerspace

The Fall Demonstration procedure can be found in Table 8.

Table 8: Fall Demo Procedure

Steps	Procedure	Success Criteria	Requirements
1	A clinician requests absent medical supply from an OR corridor	ORB response within 5 seconds ORB response via user interface that alerts user that requested item is missing	**M.P.1, M.P.6** **M.P.6**
2	A clinician requests present medical supply from an OR corridor	ORB makes way to the supply room, reaching room and orienting to correct shelf ORB retrieves requested item by manipulating its end-effector to pre-grasp location and generate grasp pose ORB executes grasping to safely pick the target item and place on a designated bin ORB navigates with item to predefined destination	**M.P.1, M.P.2, M.P.3, M.P.4, M.P.5, D.P.1, D.P.2, D.P.3**
3	Clinician requests inspection sweep	ORB response within 5 seconds ORB makes way to testbed ORB makes sweep of testbed within 15 minutes	**M.P.1, M.P.2, M.P.4, M.P.7, D.P.3, D.P.4**

Other than the pick-and-place capability illustrated several times before, here is a visual depiction of the modalities our Fall Validation Demonstration will feature. (Figures 19, 20, and 21)

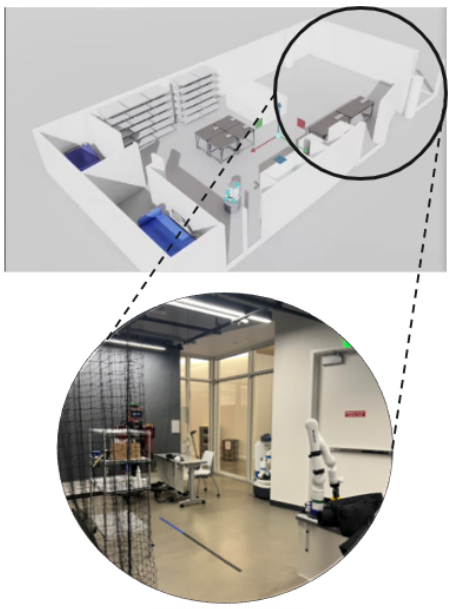


Figure 19: Visualizing A.I Makerspace as testbed for FVD

Figure 20: ORB grasping a medical item

Figure 21: ORB Inspecting and counting items on the shelf

7.4 Budget

The majority of our budget, totaling $388.22 out of $422.56, was allocated for the fabrication of shelves and our vending machine. Notably, the spirals for our vending machine constituted the most significant expense at $115. So far, we have expended 8.45% of our original $5000 allocation. Looking ahead, we anticipate that a significant portion of our remaining budget will be devoted to the design and fabrication of our custom gripper. As we intend to replace the vending machine with a grasping mechanism for fall 2024, we foresee this being a substantial upcoming expense. We plan to update our budget on the website as we advance with the gripper's design, preferably during the summer.

Table 9 provides a comprehensive overview of our expenditure to date.

Table 9: Budget: Expense until May'24

Item No.	Part Number	Description	Quantity	Supply vendor	Date	Subsystem	Price
1	Jetson Xavier board	Nvidia Jetson AGX Xavier Development Kit	1	MRSD Inventory	2024/01/19	Onboard computer	-
2	Vending machine spiral	Used spiral coil for vending machine	5	Ebay	2024/01/19	Inspection	$115
3	Solid State Drive	Samsung 980 PRO 1TB SSD	1	Jinkai	2024/02/13	Onboard computer	-
4	Testbed shelf	Regency 21" x 30" x 54" chrome 4 shelf kit	1	Webstaurant store	2024/02/22	Testbed	$80
5	Load cell (S18X4) and ADC module (HX711)	Digital load cell weight sensor for inspection	4	Amazon	2024/02/22	Inspection	$40
6	Spacers	Spacers for the load cell	4	Amazon	2024/02/22	Inspection	$40
7	DC Motor (SPG30-60K)	DC motor for the vending machine	4	MRSD Inventory	2024/02/22	Retrieval	-
8	DC motor driver (L298N)	L298N Dc motor driver	2	MRSD Inventory	2024/02/22	Retrieval	-
9	NodeMCU	ESP8266 NodeMCU Wifi Module for communication with ROS2	1	MRSD Inventory	2024/02/22	Inspection, Retrieval	-
10	Teensy v3.2	Microcontroller for DC motor PID control	1	MRSD Inventory	2024/02/22	Retrieval	-
11	Small Plastic Storage Bin	Plastic Bin for End-Effector	2	Amazon	2024/03/27	Retrieval	18.98
12	Motor Coupler	6mm to 8mm Shaft Coupling 25mm Length 18mm Diameter	3	Amazon	2024/03/27	Retrieval	15.38
13	Uxcell a14071900ux0060 Load Cell	4 Piece 4x5Kg 3-Wire Half Bridge Load Cell	6	Amazon	2024/03/27	Inspection	86.46
14	SparkFun Load Sensor Combinator	Load Cell Combinator for Half-Bridge Load Cells	4	Sparkfun	2024/03/27	Inspection	7.76
15	M4 Sized Nuts and Bolts Set	300 Pcs M4 Screw Assortment Button Head Socket Cap,M4	1	Amazon	2024/04/09	Inspection, Retrieval	9.99
16	M3 Sized Nuts and Bolts Set	420pcs M3 Screw Kit, Premium M3 Bolts	1	Amazon	2024/04/09	Inspection, Retrieval	8.99
Total Amount Used							$422.56

7.5 Risk Management

Last semester, we identified a total of 7 risks, as detailed in Table 10. Each risk is accompanied by a description and assigned owner. Our primary objective has been, and remains, to pinpoint potential obstacles that could impede the success of our project. Consequently, we continue to update our risk tables accordingly. As of the writing of this report, Tables 11, 12, 13, 14, 15, 15, 16 and 17 encapsulate the identified risks, encompassing those recognized during the Spring semester and those anticipated for the Fall semester.

As anticipated, certain risks proved to be critical to the performance of our system, necessitating multiple risk mitigation strategies. For example, we encountered significant issues with ROS-ROS2

communication, particularly concerning the ROS bridge. This emerged as our primary bottleneck, leading to numerous TF packet drops and causing all subsystems to slow down, with the Moveit pipeline even failing to function due to clock synchronization issues. As the bridge was overloaded with all the manipulation and navigation topics during our integration phase, our immediate mitigation strategy was to offload subsystems that heavily utilized the bridge's resources. Notably, we found that removing the camera's depth nodes alleviated our problems by reducing data transfer volume. However, a more robust mitigation strategy would involve improving the bridge's capabilities, either by increasing its data handling capacity or exploring alternative DDS implementations. Additionally, we are exploring alternatives that eliminate the need for the bridge altogether.

Table 10: Risk Management Table

ID	Risk Name	Requirement	Type	Likelihood	Consequence	Mitigation
R1	Person Unavailable	System-level	Schedule	3	5	• Plan in advance, keep the team informed • Introduce primary and secondary responsibilities
R2	High Network Latency	System-level	Technical	3	5	• Have a dedicated router with high bandwidth capabilities • Use onboard/wired compute
R3	Insufficient Compute Power	System-level	Technical	3	4	• Use additional, more powerful SBCs • Use Cloud
R4	Inaccurate CV Inspection	M.F.6	Technical	5	3	• Fuse weight sensor data with vision • Increase data collection, test foundation models
R5	Sub-optimal Grasps	M.F.3, M.F.5	Technical	5	5	• Fine-tune grasping policy with custom data • Experiment with a different end-effector • Rollback to Vending Machine
R6	Failure in whole body motion plans	M.F.2, M.F.3, M.F.4	Technical	4	1	• Reducing the system dimensionality • Rollback to current stable system
R7	Issue with ROS – ROS2 communication	System-level	Technical	5	5	• Remove data consuming sub-systems from the bridge • Improvise bridge with custom implementation to handle larger volume of data • Experiment with other DDS implementation which natively doesn't support ROS – ROS2 Bridge

8 Conclusions

8.1 Lessons Learned

Integration takes a long, long time

In our project, although each sub-system was developed efficiently and on time, we encountered significant delays during the integration phase. Initially, problems arose with the ROS - ROS2 bridge, followed by issues with the compute on Fetch. As these challenges continued to unfold, it became clear that we needed to reevaluate many components to address the integration difficulties we faced. Over time, our ability to foresee and mitigate potential issues improved. Despite the frustrations this process brought, it also deepened our understanding of the various off-the-shelf software we utilized, enhancing our overall project expertise.

There are more edge cases than you think there are

The old adage from our Systems course, ”prepare for the worst case and hope for the best,” resonated deeply during our recent project. Despite meticulous planning for numerous scenarios, we continually encountered unforeseen edge cases. Each time we addressed an issue and restarted the system, a new, previously unnoticed edge case would emerge. A notable example occurred during the SVD Encore, where an unexpected issue arose with one of our most dependable components. This experience reinforced a valuable lesson: it’s impossible to plan for everything. The most effective strategy is to repeatedly run our system, testing it under various conditions to uncover and resolve all potential edge cases.

We think we’ll remember everything, but it’s just not possible

In a complex system with numerous components, it’s easy to overlook certain aspects that may not seem intuitive. During development, there’s a common reassurance we give ourselves: ”This is tricky, but I’ll remember it.” While initially, we might keep these details fresh, as we delve into other areas of the system, these nuances begin to fade from memory. That’s precisely why thorough documentation is crucial. It alleviates cognitive load and ensures that anyone can understand or pick up the project, even if the original developers are not available. This not only streamlines ongoing project management but also aids in bringing new team members up to speed efficiently.

8.2 Key Fall Activities

Make system more reliable

As we’ve learned, new edge cases continually emerge, and it’s virtually impossible to anticipate every possible scenario. However, the resilience of a system isn’t about accounting for every edge case; it’s about ensuring the system remains functional even in unexpected situations. To achieve this, we must design robust fallback mechanisms. These won’t cover every specific edge case, but they will provide a safety net that keeps the system operational without manual intervention, even under unforeseen circumstances. This approach ensures stability and reliability, maintaining system integrity no matter what challenges arise.

Make the system more user-friendly

While our system features an intuitive GUI that facilitates user interaction, we recognize that medical staff might not always be able to use this interface due to various constraints in their dynamic work environment. To address this, we are planning to introduce an interactive voice interface that allows users to communicate with the system as if they were speaking to another human. This enhancement will make the system more accessible and user-friendly, enabling staff to operate it hands-free and focus more effectively on patient care without compromising on efficiency or functionality.

Multi-Modal Inspection

While our weight-sensing-based inspection system is highly accurate, we acknowledge that relying solely on weight sensing may not address all scenarios effectively. In response to the real-world challenges where additional verification is needed, we are expanding our approach to include multiple sensing modalities, beginning with vision. By integrating vision-based inspection techniques, we can enhance our system's robustness, allowing it to capture more detailed information and perform more comprehensive assessments. This multimodal approach will improve accuracy and reliability, ensuring our system can adeptly handle a broader range of inspection tasks.

References

[1] Scaling up learning across many different robot types. Retrieved from https://deepmind.google/discover/blog/scaling-up-learning-across-many-different-robot-types/.

[2] Understanding Nursing Shortages in the U.S. for 2023, 2023. Available: dailynurse.com/understanding-nursing-shortages-in-the-us-for-2023/.

[3] What Are the Top Challenges Vexing the OR and Surgical Suites?, 2023. Available: www.hpnonline.com/sourcing-logistics/article/21286570/what-are-the-top-challenges-vexing-the-or-and-surgical-suites.

[4] Isaac sim introduction, 2024. Available: https://docs.omniverse.nvidia.com/isaacsim/latest/index.html. Accessed on: May 2, 2024.

[5] Moveit 2 documentation, 2024. Available: https://moveit.picknik.ai/main/index.html. Accessed on: May 2, 2024.

[6] A. Kirillov et al. Segment anything, April 2023. Available: arxiv.org/abs/2304.02643.

[7] Roger A. Light. Mosquitto: server and client implementation of the mqtt protocol. *Journal of Open Source Software*, 2(13):265, 2017.

[8] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7, May 2022.

[9] Steve Macenski, Shrijit Singh, Francisco Martin, and Jonatan Gines. Regulated pure pursuit for robot path tracking. *Autonomous Robots*, 2023.

[10] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.

[11] Steven Macenski, Francisco Martin, Ruffin White, and Jonatan Ginés Clavero. The marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[12] Steven Macenski, Alberto Soragna, Michael Carroll, and Zhenpeng Ge. Impact of ros 2 node composition in robotic systems. *IEEE Robotics and Autonomous Letters (RA-L)*, 2023.

[13] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.

[14] Dillon Reis, Jordan Kupec, Jacqueline Hong, and Ahmad Daoudi. Real-time flying object detection with yolov8, 2023.

[15] DV Lu A. Merzlyakov M. Ferguson S. Macenski, T. Moore. From the desks of ros maintainers: A survey of modern capable mobile robotics algorithms in the robot operating system 2. *Robotics and Autonomous Systems*, 2023.

[16] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.

[17] Y.-C. Tung, G.-M. Chang, and H.-Y. Chang. Associations between types of health insurance and in-hospital mortality in acute myocardial infarction: A nationwide retrospective cohort study in taiwan. *PLoS ONE*, 11(5):e0154949, 2016.

A Appendix

Table 11: Risk 1

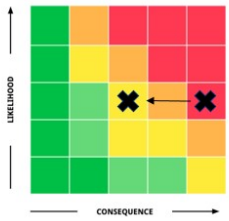
Risk Title	Person unavailable	Consequence matrix
Risk Owner	Project Manager	
Description	Risk Type	
Unavailability due to personal/academic issues	Schedule	
Mitigation Strategy		
Action		Expected Outcome
Plan in advance, keep the team informed		No delays in tasks
Introduce primary and secondary responsibilities		Ensure overlap, remove over-dependence on one person

Table 12: Risk 2

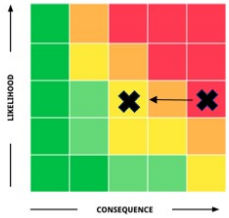
Risk Title	High Network Latency	Consequence matrix
Risk Owner	Software Lead	
Description	Risk Type	
Wi-fi latency issues would cause our entire system to slow down	Technical	
Mitigation Strategy		
Action	Expected Outcome	
Have a dedicated router with high bandwidth capabilities	Ensure less congestion	
Use onboard/wired compute	Reduce network traffic	

Table 13: Risk 3

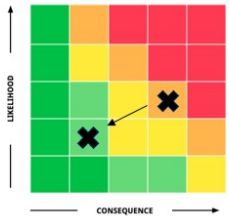
Risk Title	Insufficient compute power	Consequence matrix
Risk Owner	Software Engineer	
Description	Risk Type	
Not enough onboard computation to run heavy algorithms	Technical	
Mitigation Strategy		
Action	Expected Outcome	
Use additional, more powerful SBCs	Increase onboard compute	
Use the cloud	Reduce onboard compute load	

Table 14: Risk 4

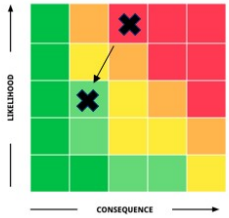
Risk Title	Inaccurate CV Inspection	Consequence matrix
Risk Owner	Perception Lead	
Description	Risk Type	
CV algorithm fails due to - Occlusions - Item Translucency - Poor model performance/insufficient data	Technical	
Mitigation Strategy		
Action		Expected Outcome
Fuse weight sensor data with vision		Increase scene understanding
Increase data collection, test times, test state of the art models		Better model

Table 15: Risk 5

Risk Title	Sub-Optimal Grasps	Consequence matrix
Risk Owner	Manipulation Lead	
Description	Risk Type	
Grasps from the pipeline are often suboptimal, leading to objects being either not gripped or gripped unsafely.	Technical	
Mitigation Strategy		
Action		Expected Outcome
Fine-tune grasping policy with data collected under our setup		Increase scene understanding
Experiment with a different/custom end-effector		Solution Restructured
Rollback to Vending Machine/Experiment with other retrieval mechanisms		Problem Restructured

Table 16: Risk 6

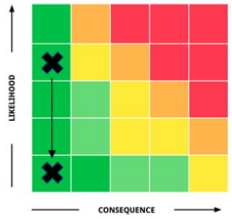
Risk Title	Failure in Computing Whole Body Motion Plans	Consequence matrix
Risk Owner	Manipulation Lead	
Description	Risk Type	
Whole body motion plans cannot be computed due to due to high-DOF nature, stability constraints, and the need for obstacle avoidance and efficient movements.	Technical	
Mitigation Strategy		
Action	Expected Outcome	
Reducing the system dimensionality using constraints	Increase planning efficiency	
Rollback to current stable system	Already working and tested	

Table 17: Risk 7

Risk Title	Issues in ROS – ROS 2 Communication	Consequence matrix
Risk Owner	Software Lead	
Description	Risk Type	
Increased latency and computational demands arise from the necessity to transfer larger volumes of data across the ROS – ROS 2 bridge.	Technical	
Mitigation Strategy		
Action	Expected Outcome	
Remove data consuming sub-systems from the bridge	Reduce volume of data transferred	
Improvise bridge with custom implementation to handle larger volume of data	Enhanced ROS –ROS2 bridge capabilities	
Experiment with other DDS implementation which natively doesn't support ROS – ROS2 Bridge	Enhanced ROS –ROS2 bridge capabilities	