Next-Gen Retail: LALA, a Service Robot for Seamless Navigation and Intelligent Restocking

I-Pei Lee, Chi-Hsiang Lo, Shih-Ting Hsu, Kuan-Ting Lin, Ching-Jui Wang
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, ROC

***Abstract*—This project presents a retail service robot designed to enhance customer experience and optimize inventory management. Built on the Scout Mini platform, the robot integrates advanced navigation, a YOLOv8-powered vision system, and modular APIs for seamless operation. Field evaluations demonstrate the robot's ability to autonomously guide customers, identify products in real-time, and manage restocking tasks efficiently.**

***Index Terms*—service robot, scout mini, retail market, YOLO, SLAM, Flask, navigation, vision, restocking**

I. Introduction

The development of autonomous service robots has garnered significant interest in recent years, especially in applications within retail environments. This paper introduces **"Lala"**, a retail market service robot designed to enhance shopping experiences and streamline store operations. Built on the **Scout Mini four-wheel-drive platform**, Lala is equipped with advanced navigation, vision systems, and interactive capabilities, providing an all-in-one solution for modern retail challenges.

Lala addresses three primary challenges in retail scenarios. First, it assists customers by **guiding them to their desired items** within the store using efficient navigation and mapping systems. Second, it leverages **YOLO v8 for object detection** to **identify products on shelves in real-time**, ensuring accurate product recognition. Lastly, it automates restocking processes by initiating tasks when inventory levels are insufficient. These capabilities are realized through an integration of cutting-edge technologies, including **SLAM for localization and mapping**, **Dijkstra's algorithm for path planning**, and a **Flask-based user interface** to facilitate user interaction.

The hardware design features robust construction using an **aluminum extrusion frame** for structural support, **a Hokuyo URG-04LX-UG01 LiDAR** for obstacle detection, and a **webcam mounted 50 cm above the ground** for optimal shelf detection. The system's modular design ensures ease of maintenance and scalability for additional functionalities. Lala also incorporates a **dynamic restocking system** where tasks are relayed to store personnel or handled autonomously, enhancing operational efficiency.

This paper outlines the comprehensive design and implementation of Lala, covering its hardware architecture, software integration, navigation strategies, and user interface. The system has been rigorously tested in simulated retail environments with diverse scenarios, demonstrating its reliability and adaptability. By bridging customer assistance with inventory management, Lala showcases the potential of service robots to revolutionize retail operations. The source code for this project is available on GitHub[1][2] and the demo video is accessible online[3].

II. Related Work

- **Wistron Corporation** Wistron has introduced service robots aimed at assisting in retail environments, featuring navigation and customer guidance functionalities. Their robots utilize vision systems and LiDAR for real-time mapping and product detection. These robots have been deployed in malls and supermarkets for customer assistance and inventory management.
- **XYZprinting** Known primarily for their 3D printing solutions, XYZprinting has ventured into robotics with models designed to perform shelf scanning and real-time inventory tracking in retail spaces. Their robots are equipped with cameras and AI algorithms to identify low-stock items and notify staff for replenishment.
- **SoftBank Robotics** SoftBank Robotics has developed the "Pepper" robot, widely used in retail stores such as Carrefour and Dufry. Pepper specializes in customer interaction, providing product information and guiding customers to specific sections of the store. Its user-friendly interface and ability to process natural language make it a popular choice for improving in-store engagement.
- **Fetch Robotics** Fetch Robotics has developed autonomous mobile robots (AMRs) like "Freight" and "CartConnect," which are utilized in warehouse-style retail stores like BJ's and Costco. These robots handle repetitive tasks such as inventory transport and shelf restocking, streamlining back-end operations.
- **Geek+** Geek+ is a leader in logistics and retail robotics, producing models such as the "C200M" series, which have been adopted by retail giants like Walmart and Decathlon. These robots use SLAM-based navigation and RFID systems to efficiently manage inventory and transport goods across large store layouts.
- **Omron** Omron has created the "LD Series" of mobile robots, used in retail and manufacturing for autonomous navigation and material handling. In stores, these robots

[1]https://github.com/pei0917/robotics-final
[2]https://github.com/QuantumSpawner/robotics_project
[3]https://youtu.be/Q4yTHIPMvyY

assist with inventory checks and restocking, ensuring shelves are always ready for customers.

The aforementioned products share similar objectives with our robot, "Lala," such as enhancing customer service and automating inventory management. However, Lala integrates unique features, including YOLO v8 for object detection and a customized Flask-based user interface, to offer real-time guidance and dynamic restocking. Unlike most robots that focus solely on either front-end or back-end tasks, Lala bridges the gap by assisting customers directly while simultaneously streamlining inventory workflows.

III. Methodology

A. Navigation

The navigation system is based on the ROS navigation stack [10]. Fig. 1 shows the architecture of the navigation system. The navigation system consists of four main parts: the extended Kalman filter (EKF) **(EKF Localization)**, simultaneous localization and mapping (SLAM) **(SLAM Toolbox)**, path planning **(Global Planner)**, and local controller **(Local Controller)**.

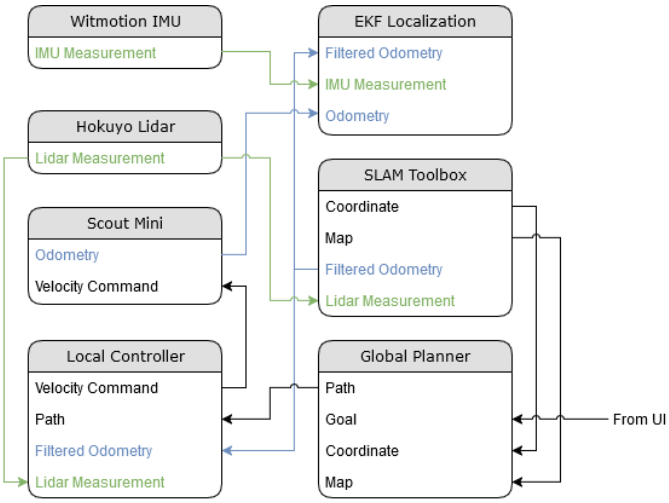


Fig. 1: Architecture of the navigation system.

1) EKF Localization: During development, we discovered that only relying on the Scout Mini's odometry data is not accurate enough owing to the high wheel slippage of the robot in the indoor environment, especially when the robot is turning. Hence, in order to improve the accuracy of the robot's pose estimation, we utilized `robot_localization` [11] ROS package to implement the EKF for localization.

By fusing an additional inertial measurement unit (IMU) that provides accurate orientation data with the odometry data that provides relative accurate translation data, the EKF can provide localization data that is accurate in both orientation and translation.

2) SLAM: The ROS package `slam_toolbox` [12] is selected for the SLAM algorithm over the `gmapping` [13] package ubiquitous in ROS 1 projects. The `slam_toolbox` package can also be used as a localizer without requiring additional packages such as `amcl` provided by the navigation stack that is found not suitable for our inaccurate odometry system during testing, which is required by the `gmapping` package in order to localize the robot in the map.

The shopping scene is first mapped and stored using `slam_toolbox` as shown in Fig. 2, which is used to localize the robot for navigation tasks.

Fig. 2: Map of shopping scene generated by `slam_toolbox`.

Here the pose estimates from the EKF are used as references for the SLAM to produce globally accurate localization data relative to the map.

3) Global Planner: The ROS package `global_planner` provided by the navigation stack is used for path planning to generate a path from the current robot pose to the goal pose. The `global_planner` package is based on the Dijkstra algorithm to generate paths that are collision-free based on the map generated by the SLAM system.

The globally accurate localization data from the SLAM is used as references for the global planner to generate a path that is accurate relative to the map.

4) Local Controller: The ROS package `dwa_local_planner` provided by the navigation stack is used for local controller to generate velocity commands for the robot to follow the path generated by the global planner. The `dwa_local_planner` package is based on the dynamic window approach (DWA) algorithm to generate velocity commands that are collision-free based on the lidar data.

The globally accurate localization data from the SLAM is not suitable for the high-frequency local controller as the localization data has lower update frequency and jumps between estimates. Hence the higher-frequency and smooth pose estimates from the EKF are used as references for the local controller to generate velocity commands.

B. Vision

As we wanted the robot to identify the items on the shelf, vision played a significant role. In fact, the technique we wanted here is object detection, which is a very common technique nowadays as far as computer vision is concerned. Among all the famous AI models for object detection, we chose YOLO-v8 (You Only Look Once version 8) because it not only is a state-of-the-art object detection AI model, it also contains multiple advantages like high accuracy, faster inference, versatility, etc.

The main task for our object detection is to identify the items or products on the shelf, and the identification is for the information of whether the robot is at the right shelf and if

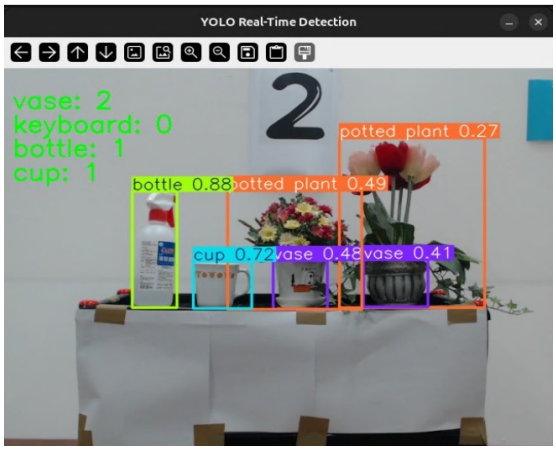


Fig. 3: Demonstration of real-time object detection

the items that the customer bought had any left after the it is taken off the shelf. In this case, the overall algorithm can be separated into three parts: **input**, **processing**, and **output**.

- **Input**: As we want the robot to identify objects on the shelf in real-time, object detection must also be able to run in real-time. Therefore, we used cv2 Image Capture to collect the vision information captured by the webcam we set on the robot. Moreover, there should be data input from API about the name of the item that the robot should detect.
- **Processing**: After the image capturing, we cut the real-time information into frame-by-frame, and let the model of YOLO to process over the frames for object detection. When processing, the model still detects all the objects inside the frame as default but only the number of the item that needs to be detected will be saved inside a dictionary.
- **Output**: Finally, the numbers of the corresponding items inside the dictionary will act as a determination of whether to restock the items on the shelf. While the program returns the item name for restocking if needed, or returns None if the item is still available.

By the algorithm above, the AI model simply detects the objects the camera captured, as shown in Fig. 1. Moreover, it identifies the specific item that the customer wants to buy and reports to the staffs if the item needs to be restocked or not. Under these setups and functions of the model, our robot is available of detect the need of restocking automatically.

C. UI

The user interface (UI) for the retail market service robot is implemented using the Flask web framework. The design incorporates a responsive layout achieved through the integration of Tailwind CSS, while dynamic icons are rendered using Feather Icons. Real-time interactions are facilitated by Socket.IO. The following packages and libraries are utilized in the HTML structure:

The system features four pages: **Home**, **Navigation**, **Restocking**, and an additional **Password** page. The Password page ensures that only authorized clients can access the Restocking page by requiring a correct password.

1) Home Page: The Home page provides two primary buttons: one to access the Navigation page and another to access the Restocking page. If the Restocking button is selected, the user must first authenticate via the Password page.

2) Navigation and Restocking Pages: In the Restocking page, users can create tasks by either selecting a region first (which filters the available products) or directly selecting a product (automatically assigning the region). After task creation, users can click *Start* to initiate the task. The active navigation task display updates with the number of tasks currently in progress.

When the robot arrives at the destination, the system emits a `robot_arrived` event using Socket.IO. This triggers a pop-up menu with the following options:

- Create a new navigation task (closes the pop-up).
- Restock products (invokes `append_restock_deque` API, redirects to the Home page).
- Ask the clerk a question (notifies the clerk, redirects to the Home page).
- "I am fine now" (redirects to the Home page).

In the Home page, the system checks if the restock deque is empty. If not, the `restock()` API is invoked to create a new restock task and redirect the user to the Restocking page. Users can also manually navigate to the Restocking page after password authentication.

3) Task Flow Differences: The Navigation and Restocking pages share similar functionality, with one key distinction: upon the robot's arrival during a Restocking task, no pop-up appears. Instead, the system simply reminds the client to restock products from the robot's basket.

4) User Interface Snapshots: To better illustrate the UI design and functionality, a comprehensive set of snapshots has been provided below. These snapshots showcase the various interface elements and functionalities, such as navigation task creation, task progress, and completion status.

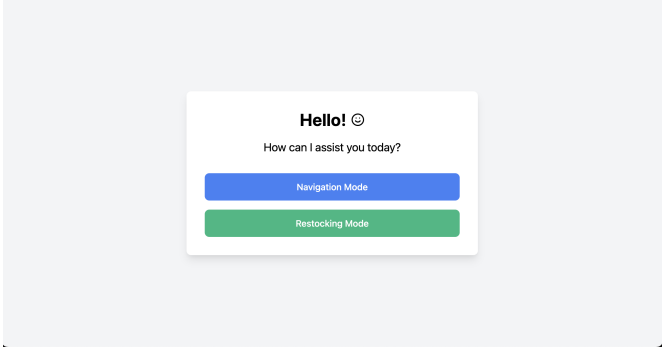


Fig. 4: Home Page

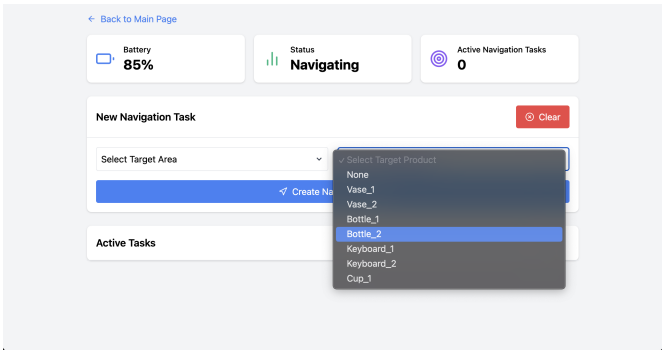


Fig. 5: Navigation Task Creation - Region Selection

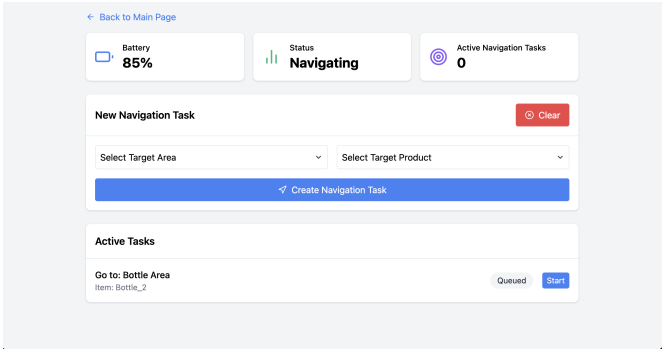


Fig. 6: Navigation Task Created

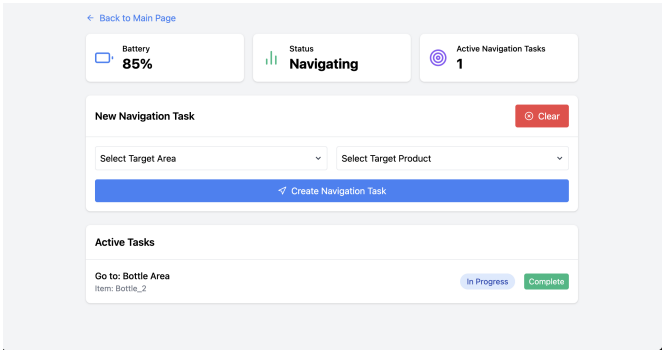


Fig. 7: Navigation Task In Progress

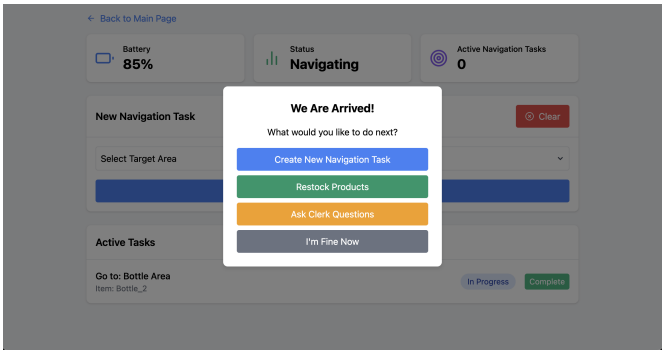


Fig. 8: Robot Arrived Pop Up

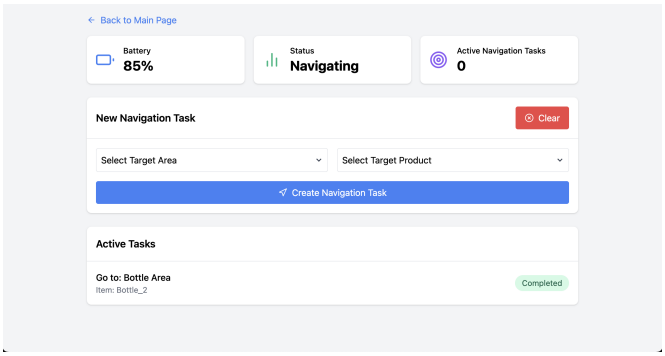


Fig. 9: Restocking Navigation Task Creation - Region Selection

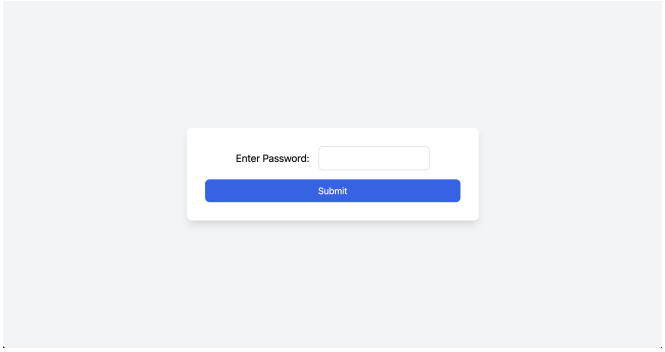


Fig. 10: Task Completed

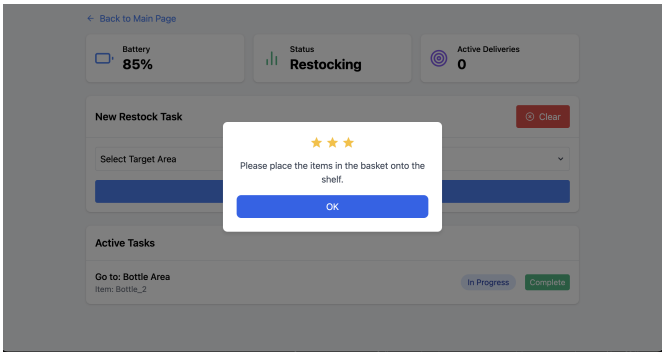


Fig. 11: Restocking Reminder

D. API Connection

1) System Architecture Overview: The system architecture implements a modular microservices pattern with four primary components:

- **UI Module**: Frontend interface containing home, task status and restock functionalities. Socket.IO Handles real-time bidirectional communication
- **Server**: Central controller managing communication between modules
- **Navigation Module**: Handles robot movement and routing
- **Shared Module**: Contains shared resources and configuration

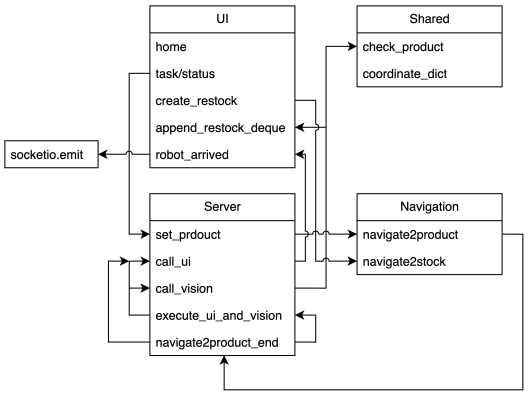


Fig. 12: API connection workflow

2) UI Module Functions:

- **home**: Main entry point that checks restock queue and redirects to restock page if tasks exist, otherwise renders home page
- **task/status**: Updates and displays real-time task execution status throughout operations
- **create_restock**: Initiates restocking operations and robot movement to stock location
- **append_restock_deque**: Manages restocking task queue based on vision system feedback
- **robot_arrived**: Updates UI when robot arrives at designated locations

3) Server Module Functions:

- **set_product**: Processes product selection and retrieves coordinates for navigation
- **call_ui**: Notifies UI system about robot arrival and status changes
- **call_vision**: Triggers vision system to check product stock levels
- **execute_ui_and_vision**: Simultaneously executes UI updates and vision system checks
- **navigate2product_end**: Processes robot arrival and initiates subsequent operations

4) Navigation Module Functions:

- **navigate2product**: Controls robot movement to selected product locations using coordinate data
- **navigate2stock**: Directs robot to stock location for restocking operations

5) Shared Module Resources:

- **check_product**: Product validation utilities for stock level verification
- **coordinate_dict**: Stores and provides mapping of all product and location coordinates

6) Product Processing Flow: The product processing flow is initiated when a user selects the "navigation mode" option on the UI home page. This action transitions the interface to the navigation page, where the user specifies their desired product. Upon clicking the "start" button, the UI updates the task status and sends a request to the server's `set_product` function.

The server retrieves the corresponding product coordinates from the shared `coordinate_dict` and instructs the navigation module to execute the `navigate2product` function. This initiates the robot's movement to the designated product location.

Upon reaching the destination, the navigation module signals the server via the `navigate2product_end` function. The server then triggers the `execute_ui_and_vision` function, which initiates two parallel processes:

- `call_ui`: Updates the UI by notifying the `robot_arrived` function.
- `call_vision`: Performs a stock-level verification of the selected product.

If the vision system detects a need for restocking, it activates the UI's `append_restock_deque` function to enqueue a restocking task. This task is processed by the UI's `create_restock` function, which coordinates with the navigation module's `navigate2stock` function to guide the robot to the stock location. The final restocking operation is performed by on-site staff.

7) Stock Management Flow: The stock management flow begins when staff complete a restocking operation and click the "start" button. This action prompts the UI to update the task status and notify the server via the `set_product` function, similar to the product processing flow.

The server retrieves the relevant location coordinates from the shared `coordinate_dict` and directs the navigation module to execute the `navigate2product` function, guiding the robot to the specified destination.

Upon arrival, the navigation module signals completion to the server through the `navigate2product_end` function. The server then triggers the `call_ui` function, prompting the UI's `robot_arrived` function to update the system status. This completes the stock management cycle, ensuring accurate synchronization between the physical operation and system updates.

IV. Experimental Setup

A. Hardware

In our hardware setup, we utilized an **Inertial Measurement Unit (IMU)**, a **webcam**, a **Hokuyo URG-04LX-UG01 LiDAR**, an **aluminum extrusion frame**, **dense boards**, and a **laptop computer**. The **aluminum extrusion frame** served as the primary structural component, providing robust support and modularity for mounting additional equipment. The frame was constructed in two 25 cm layers, resulting in a total height of 50 cm, with the **vehicle base** itself measuring approximately 25 cm in height. This design ensured stability and compatibility with the **compact Scout Mini four-wheel-drive platform.**

To accommodate various sensors, **dense boards** were cut and affixed using screws, enabling precise placement of the **webcam** and **LiDAR**. The **webcam** was strategically mounted

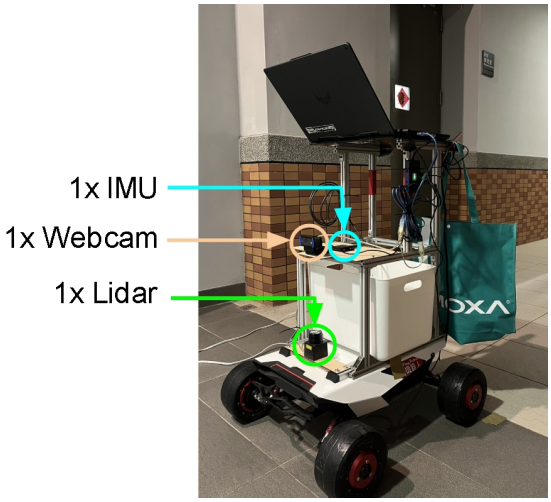


Fig. 13: LALA Hardware Setup

on a platform 50 cm above ground level, a height optimized for detecting and identifying items on shelves during experiments. The **LiDAR**, a **Hokuyo URG-04LX-UG01**, was installed on the front of the vehicle to facilitate obstacle detection and environment mapping. This configuration provided the robot with a 2D spatial awareness capability essential for navigation in retail scenarios.

The **laptop computer**, responsible for processing vision and navigation algorithms, was positioned at the topmost level of the structure. This placement allowed for easy access during debugging and experiments, as well as convenient interaction by customers and operators. To enhance functionality, a **metal frame handle** was incorporated into the design, enabling the attachment of shopping bags for customers or storage for restocked items.

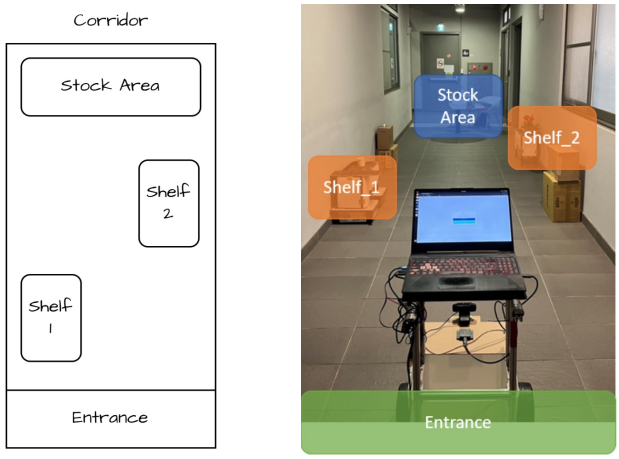
Each component was integrated to maximize efficiency and reliability, creating a cohesive hardware architecture tailored for retail applications. This setup provided the foundation for advanced capabilities such as real-time object detection, SLAM-based navigation, and dynamic task execution, which are discussed further in the subsequent sections.

B. Scene Set

To run a complete demonstration of the robot's functions, we built a scene to imitate the scenario of a market. To ensure all of our robot's functionalities can be demonstrated well, there are several crucial parts of our scene set. First are the shelves, we prepared two plastic shelves as the representation of the actual shelves in the market, with several items on the shelves. Second, the items we chose to put on the shopping list for demonstration are keyboards, cups, bottles, and vases. However, to increase the authenticity of the demo, we added more items on the shelves, as a mimic scenario inside a market.

Next are the roles of the demonstration, there are three roles in total of the demo, a customer who comes to the market to buy the items she wants, using the robot for navigation and assistant, clerk 1 who works at the restocking area, puts the items inside the box on the robot when the robot comes to the area for restocking, and clerk 2 who works at the lane area of the market, puts the items contained inside the box of the robot onto the corresponding shelves.

Last but not least, we set the whole scene at the lane outside our laboratory, the schematic diagram and the actual scene photo are shown in Fig. 2 and Fig. 3. Under this scene set, the robot can be able to demonstrate all its functionalities clearly, including navigation, object identification, restocking detection etc.



(a) The schematic diagram of the scene set

(b) The actual photo of the scene set

Fig. 14: Comparison between the schematic diagram and the actual scene set, aligned to the same height.

V. Results

The shopping assistance robot developed in this project demonstrates its ability to address complex retail scenarios through seamless integration of navigation, vision-based product detection, and inventory management functionalities. By combining these capabilities, the robot effectively enhances customer service while streamlining operational processes in retail environments.

The robot performs the following key tasks:

1) Navigates customers to their desired products with high precision.
2) Detects product availability in real-time using a vision system and identifies when restocking is required.
3) Waits until the customer no longer needs assistance before autonomously initiating restocking tasks.
4) Responds to user inputs through a pop-up menu offering options such as requesting assistance from a clerk, initiating manual restocking, or autonomously navigating to the stock area.

These features enable the robot to handle complex service scenarios efficiently. For example, consider the following use case:

A customer wishes to purchase two **bottles**. The robot guides the customer to the designated **bottle area** but identifies

only one **bottle** in stock. The customer selects the **restock option** and decides to browse nearby. The robot autonomously navigates to the **stock area**, where **clerk_1** restocks the required **bottles** into the robot's storage compartment. The robot then returns to the **bottle area**, where **clerk_2** places the restocked items on the shelf. The **customer**, now finding two **bottles** available, successfully completes the purchase.

Later, the same **customer** decides to buy a **keyboard**. Upon navigating to the **keyboard area**, only one **keyboard** is available. Since the **customer** needs only one, they choose to proceed without triggering an immediate restock. However, the system automatically creates a **restock task** for **keyboards**. After the **customer** completes the navigation task by clicking the "**I'm fine**" option at the **store entrance**, the robot returns to the **stock area** to execute the previously queued **restock task**. The robot follows the same workflow to ensure the **keyboard area** is replenished for future customers.

These results highlight the robot's ability to handle real-world scenarios that combine navigation, real-time decision-making, and task coordination. The lightweight and adaptable nature of the system ensures it can be deployed on various robotic platforms at a low cost. By reducing the reliance on large employee teams and ensuring timely restocking, the robot significantly improves operational efficiency and customer satisfaction in retail environments.

VI. Conclusions

In this project, we have developed a shopping assistance robot that effectively addresses critical challenges in the retail industry while enhancing the overall shopping experience. By seamlessly integrating navigation, real-time inventory detection, and autonomous restocking capabilities, the robot ensures timely product availability and efficient task execution.

The system's user-friendly graphical interface fosters seamless interaction for both customers and clerks, demonstrating the practicality of robotics in diverse retail settings. This robot reduces reliance on large employee teams, minimizes customer frustration in locating products, and maintains a lightweight, cost-effective design adaptable to various platforms.

Through its demonstrated ability to navigate complex tasks and provide personalized assistance, the robot exemplifies the potential of robotics to transform retail operations. By improving customer satisfaction and operational efficiency, this project represents a significant step toward the future of automated service in retail environments.

Appendix A
Work Division

The contributions of each team member are detailed in the table I.

Name	**Report (related to code)**	**Others**
I-Pei Lee	UI, API Connection, Results, Formatting	Video Recording
Chi-Hsiang Lo	Navigation, Conclusion	Video Recording
Shih-Ting Hsu	Abstract, API Connection	Video Recording
Kuan-Ting Lin	Introduction, Related Work, Hardware	Video Editing
Ching-Jui Wang	Vision, Scene Set	Demo Presentation

TABLE I: Work Division Among Team Members

References

[1] Wistron Corporation. (n.d.). *Autonomous Mobile Robot Solution - Wifundity*. Retrieved from https://wifundity.wistron.com/en/index.html

[2] ADLINK Technology. (n.d.). *Shopping Mall Service Robot — ROS 2 Use Case*. Retrieved from https://www.adlinktech.com/tw/ros2-wistron-service-robot

[3] ITHome. (2016, August 4). Retrieved from https://www.ithome.com.tw/newstream/103417

[4] 3ders.org. (2016, January 26). *XYZPrinting launches 3D printing creative space and service robots in Taiwanese store*. Retrieved from https://www.3ders.org/articles/20160126-xyzprinting-launches-3d-printing-creative-space-and-service-\protect\penalty\z@robots-in-taiwanese-store.html

[5] Robotics On The Runway. (n.d.). *XYZPrinting — NERA*. Retrieved from https://roboticsontherunway.com/portfolio-items/xyzprinting-nera/

[6] SoftBank Robotics. (n.d.). *Pepper the humanoid and programmable robot*. Retrieved from https://www.softbankrobotics.com/emea/en/pepper

[7] Fetch Robotics. (n.d.). *Autonomous Mobile Robots for Material Handling*. Retrieved from https://fetchrobotics.com/

[8] Geek+. (n.d.). *Robotic Solutions for Logistics*. Retrieved from https://www.geekplus.com/

[9] Omron. (n.d.). *LD Series Mobile Robots*. Retrieved from https://industrial.omron.us/en/products/ld

[10] S. Macenski, F. Martín, R. White, J. Clavero. "The Marathon 2: A Navigation System," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.

[11] T. Moore and D. Stouch, "A Generalized Extended Kalman Filter Implementation for the Robot Operating System," in Proc. 13th Int. Conf. Intell. Auton. Syst. (IAS-13), July 2014.

[12] Macenski, S., Jambrecic I., "SLAM Toolbox: SLAM for the dynamic world," Journal of Open Source Software, 6(61), 2783, 2021.

[13] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard, "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters," IEEE Transactions on Robotics, Volume 23, pages 34-46, 2007.