Programming Paradigms

COMP1029 (PDP) | Spring Semester

Java Program Report (Drink Serving Robot System)

No.	Name	Student ID
1.	Lim Chai Shuen	20612781
2.	Moo Shu Ann	20609412

Contents

1. Introduction

The Drink Serving Robotic System is a sophisticated, intelligent Java-based application meticulously designed to simulate the operations of autonomous service robots in a modern restaurant environment. This system provides a comprehensive platform for managing drink service operations for both regular patrons and VIP customers, ensuring that customer satisfaction and operational efficiency are prioritized.

By integrating advanced features such as dynamic social distancing, intelligent crowd management, real-time monitoring of serving success rates, and battery optimization, the system models a highly realistic and responsive robotic service framework. It demonstrates the ability to adapt to varying crowd conditions, respond to emergency situations through automated evacuation protocols, and prioritize high-value customers through specialized VIP servicing mechanisms. Furthermore, the system incorporates robust user interaction mechanisms, ensuring smooth control and navigation, while maintaining a high degree of operational autonomy for the robots themselves. This project exemplifies the application of object-oriented design principles to solve real-world challenges in the service industry, emphasizing flexibility, safety, and efficiency.

2. Project Objective

The primary objective of this project was to engineer a robust, interactive Java-based robotic serving system capable of addressing the complex operational needs of a modern restaurant environment. Specifically, the system was designed to:

I. Simulates drink serving by robots.

II. Implement dynamic social distancing protocols, intelligently adjusting robot movement based on the proximity of nearby individuals.

III. Manage dining area occupancy by monitoring and controlling spot capacities to ensure a safe and organized environment.

IV. Deliver prioritized service to VIP customers through specialized detection and handling mechanisms.

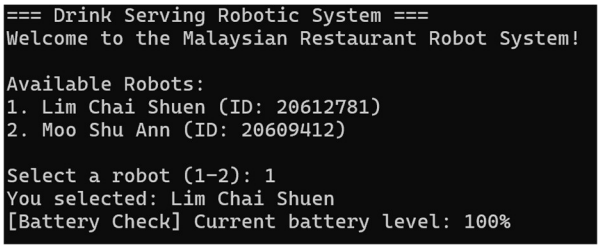
V. Ensure uninterrupted robot operations through an intelligent battery management and auto-recharging system.

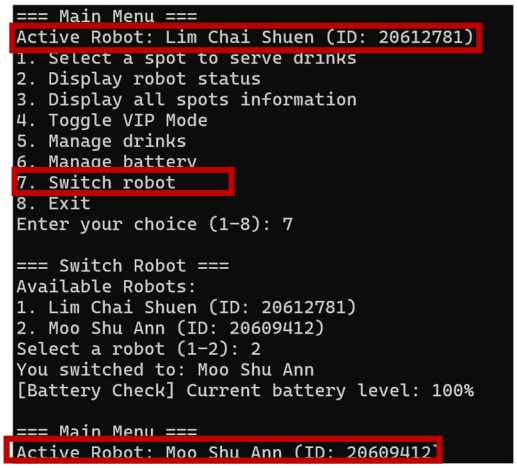
VI. Maintain optimal drink inventory levels by incorporating monitoring, serving, and refilling functionalities for both regular and specialty drinks.

3. Features and Fulfillment of Criteria

The developed Drink Serving Robotic System integrates multiple sophisticated features, each addressing specific functional requirements. The implementation details for these features are outlined below:

a) Multiple Robot Simulation

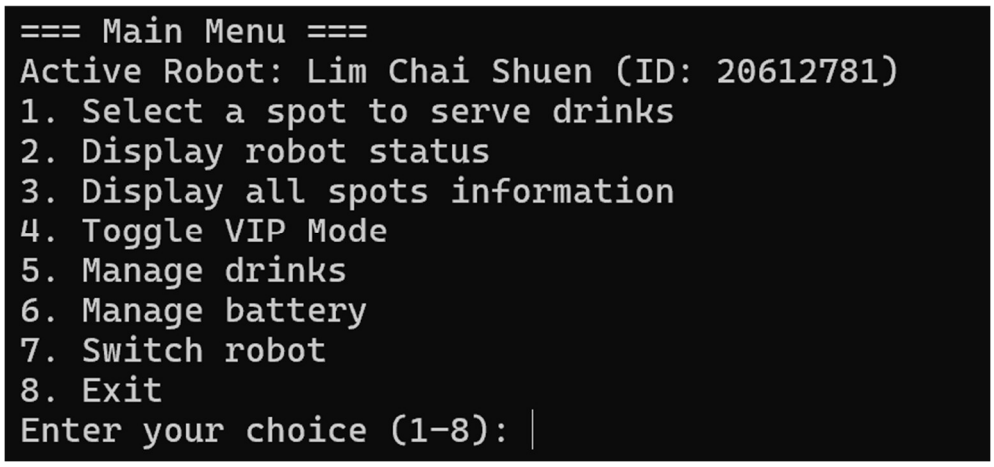




The system supports the simulation of multiple serving robots. A Robot[] robots array holds instances of robot objects, each with unique identifiers and names.

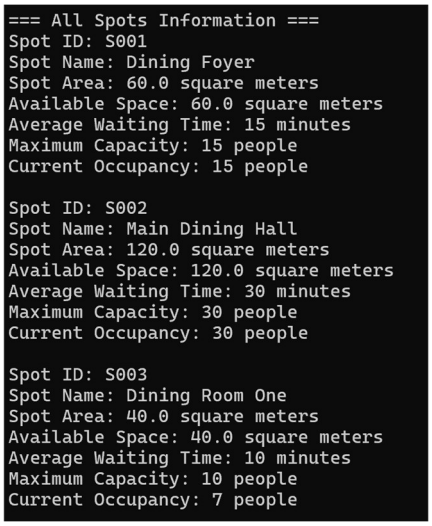
At startup, the user selects the active robot through a selection menu. Robots can also be switched dynamically during runtime, demonstrating scalability and flexible agent assignment (switchRobot() method).

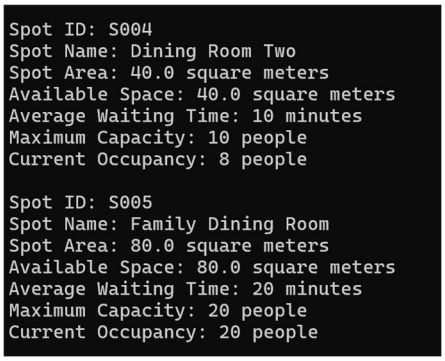
b) Professional Menu Navigation and System Interaction



The user interface is designed with professionalism and clarity in mind. The runSystem() and displayMainMenu() methods present intuitive menu-driven interactions, ensuring users can navigate the system efficiently. All user inputs are validated, and feedback is provided at each step, resulting in a user-friendly and reliable operational flow.

c) Display of All Sports Information (Option 3 in Main Menu)





The "Display of All Spots Information" feature provides a comprehensive overview of all dining areas in the restaurant by iterating through the spots array and calling each RestrictedSpots object's displaySpotInfo() method. This method outputs clearly formatted key details such as Spot ID, name, total area, available space, average wait time, maximum capacity, and current occupancy. The maximum capacity is automatically calculated based on public health guidelines—specifically, by dividing the spot's area (or available space) by 4m² per person to maintain proper distancing, as implemented in the constructor and updated dynamically in setAvailableSpace(). To simulate real-world usage conditions, random number generation is used during object initialization with Math.random() to assign a realistic current occupancy value between 0 and the maximum capacity. Before robots enter a spot, they invoke the isFull() method to determine whether occupancy limits have been reached, allowing the system to grant or deny robot entry dynamically. This approach helps restaurant staff monitor dining areas in real time, make informed deployment decisions for serving robots, and ensure

ongoing compliance with health and safety regulations through structured, simulation-based data management.

d) Dynamic Social Distancing & Real-Time Crowding Level Detection and Analysis

```
=== Available Spots ===
1. Dining Foyer (Current Occupancy: 8/15)
2. Main Dining Hall (Current Occupancy: 5/30)
3. Dining Room One (Current Occupancy: 1/10)
4. Dining Room Two (Current Occupancy: 10/10)
5. Family Dining Room (Current Occupancy: 20/20)
Select a spot (1-5): 1

Entry permit required before entering Dining Foyer
```

Entrance Permitted:

```
=== Entry Permit Request ===
Requesting entry permit for spot: S001
PERMIT GRANTED: Entry allowed to Dining Foyer

Entering Dining Foyer...
Spot ID: S001
Spot Name: Dining Foyer
Spot Area: 60.0 square meters
Available Space: 60.0 square meters
Average Waiting Time: 15 minutes
Maximum Capacity: 15 people
Current Occupancy: 8 people

=== Dining Foyer Map ===
[R][T][ ][T][ ][ ][ ][T][T][ ]
[T][ ][ ][ ][T][ ][ ][ ][T][T]
[ ][ ][ ][ ][ ][T][ ][ ][T][ ]
[ ][T][T][ ][ ][T][ ][ ][ ][T]
[ ][ ][T][ ][ ][ ][T][T][ ][ ]
[T][ ][ ][ ][ ][ ][ ][ ][ ][ ]

=== Dynamic Distancing Check ===
Please enter the distance from people in meters:
Left side: 1
Right side: 1
Front: 1
Back: 1

You are safe in dynamic distancing!

Current crowding level: High
Recommended serving path: Perimeter path - stay near walls when possible
Estimated collision risk: 30.0%
```

Entrance Not Permitted:

```
=== Available Spots ===
1. Dining Foyer (Current Occupancy: 14/15)
2. Main Dining Hall (Current Occupancy: 23/30)
3. Dining Room One (Current Occupancy: 9/10)
4. Dining Room Two (Current Occupancy: 4/10)
5. Family Dining Room (Current Occupancy: 20/20)
Select a spot (1-5): 1

Entry permit required before entering Dining Foyer

=== Entry Permit Request ===
Requesting entry permit for spot: S001
DENIED: Spot is too crowded for safe social distancing!
Cannot enter spot without valid permit. Returning to main menu...
```

The **Dynamic Social Distancing module** is a crucial component of the Drink Serving Robotic System, developed to ensure that the robot maintains a minimum distance of 1 meter from people in all four directions—front, back, left, and right—at all times. This functionality is implemented in the DynamicDrinkServing class, which defines a constant SAFE_DISTANCE and includes several key methods to monitor and respond to proximity data.

The checkSafeDistance() method determines whether all surrounding distances meet the safety requirement. If a violation is detected, the provideMoveAdvice() method offers directional movement suggestions—for example, advising the robot to move right if someone is too close on the left. To provide real-time feedback, the determineContactStatus() method labels the situation as either “Safe Distance” or “Casual Contact.”

In addition to proximity checks, the module also performs crowding level analysis using the calculateCrowdingLevel() method. It classifies the environment into Low, Moderate, or High density zones. Based on this, the recommendServingPath() method dynamically adjusts the robot’s path—choosing between direct, clockwise, or zigzag routes to avoid densely populated areas.

The estimateCollisionRisk() method calculates a collision risk score (capped at 100%), combining proximity and crowd data. This helps the robot make safer decisions in real time and avoid areas with high collision potential.

Furthermore, the system incorporates a permission mechanism for navigating **restricted zones**. Integration with the RestrictedSpots class allows the robot to check if it has been granted access before entering sensitive or reserved tables. If no permission is granted, the robot avoids these areas entirely.

Although real-time sensor input and emergency triggers were initially considered, they have been abstracted into modular methods to support easier testing and future expansion. Overall, this module enables the robot to safely and intelligently navigate complex and changing restaurant environments.

4. Extraordinary Features

The Drink Serving Robotic System introduces several advanced and innovative features beyond basic criteria, enhancing realism, safety, and operational excellence. The key extraordinary implementations are described below:

4.1 Interactive Drink Serving Selection Menu (Option 1 in Main menu)

```
=== Serve Drinks Menu ===
1. Serve regular drinks
2. Serve specialty drinks
3. Display serving statistics
4. Return to main menu
Enter your choice (1-4): 3
```

```
=== Drink Serving Statistics ===
Successful servings: 18
Failed servings: 1
Success rate: 94.7%

Recent service logs:
2025-04-29T16:52:28.178518400 - Drink serving attempt: SUCCESS
2025-04-29T16:52:29.151124700 - Drink serving attempt: SUCCESS
2025-04-29T16:52:29.757714500 - Drink serving attempt: SUCCESS
2025-04-29T16:52:30.856734900 - Drink serving attempt: SUCCESS
2025-04-29T16:52:32.282521100 - Drink serving attempt: SUCCESS
```

Description

The **Interactive Drink Serving Selection Menu** was designed to replicate a realistic and efficient drink-ordering process in a restaurant environment. Drawing inspiration from human waitstaff behaviors, the menu system supports both **regular** and **specialty drinks**, with inventory management, fallback mechanisms, and user-friendly navigation. The goal was to ensure smooth, prioritized, and responsive drink service—particularly for VIP customers—while maintaining an intuitive interface for operators.

Implementation

- **Inventory Management:** The system maintains 10 units for regular drinks and 3 units for specialty drinks. When inventory is depleted, the menu prevents further serving and prompts users to refill. Specialty drinks are prioritized for VIP

customers; if unavailable, the system defaults to serving regular drinks. Real-time inventory checks and fallback procedures ensure smooth operations.

- **DrinkServingMonitor:** This component logs every serving attempt with timestamps and success indicators. It tracks success rates, such as 94.7%, providing real-time diagnostics to operators for performance monitoring.
- **Navigation:** The system allows users to return to the main interface without restarting the program, simulating natural workflows and enhancing user experience.
- **Battery Usage:** The system integrates battery usage, deducting 1% per regular drink and 2% per specialty drink. When battery levels drop to 20%, warnings are triggered to alert operators and avoid service interruptions.
- **Input Validation:** The getUserChoice() method handles input validation by enforcing numeric limits. It also uses a try-catch loop to handle exceptions, ensuring system stability even with incorrect user inputs.

Sophistication

The system's sophistication lies in its robust error handling, realistic simulation of prioritization and inventory depletion, and adaptive service logic. The use of clear fallback mechanisms ensures uninterrupted operation, even in low-resource scenarios. Real-time success tracking and battery monitoring add depth to the system's realism and operational integrity.

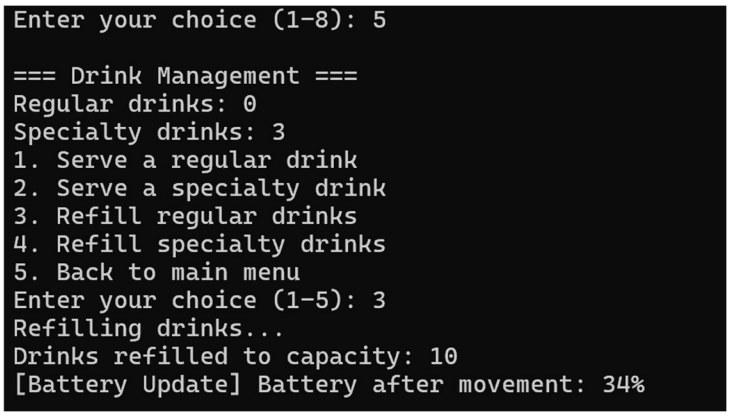
Another key innovation is its input validation strategy. The use of exception handling and looped prompts ensures that the program does not crash due to invalid input, maintaining usability in unpredictable real-world contexts.

Impact

The drink selection menu system significantly enhances user interaction, reliability, and service quality in robotic or automated environments. By simulating human-centric workflows and integrating safeguards against errors, it offers a more natural, resilient, and user-friendly interface. Operators benefit from real-time analytics for performance

monitoring, which can inform future adjustments and maintenance. The thoughtful integration of battery awareness and fallback serving logic contributes to sustained operations even in constrained situations. Ultimately, the system bridges field research and technical implementation, setting a strong foundation for more intelligent, context-aware service robots.

4.2 Convenience Drink Management System



Description

The Convenience Drink Management System offers a streamlined interface to manage drink-related tasks efficiently, tailored for robotic drink service environments. Designed with usability in mind, it features a menu-driven layout that allows users to monitor drink inventory and perform key actions without requiring a restart. The system supports both regular and specialty drinks, integrating core service and maintenance functions into a cohesive and accessible control flow.

Implementation

The interface presents five main options: serve regular drinks, serve specialty drinks, refill regular drinks, refill specialty drinks, and return to the main menu. Each option triggers a corresponding robot behavior. For example:

- **Serving regular drinks** invokes serveDrink() and consumes 1% battery.
- **Serving specialty drinks** calls serveSpecialtyDrink() with a 2% battery cost.

- **Refill operations** deduct 3% battery, reflecting the added effort of replenishing supplies.

Inventory levels for both drink types are displayed and updated in real time. Input is managed through the getUserChoice() method, which restricts entries to valid integers (1–5), ensuring that the system remains stable even when faced with unexpected or invalid input.

Sophistication

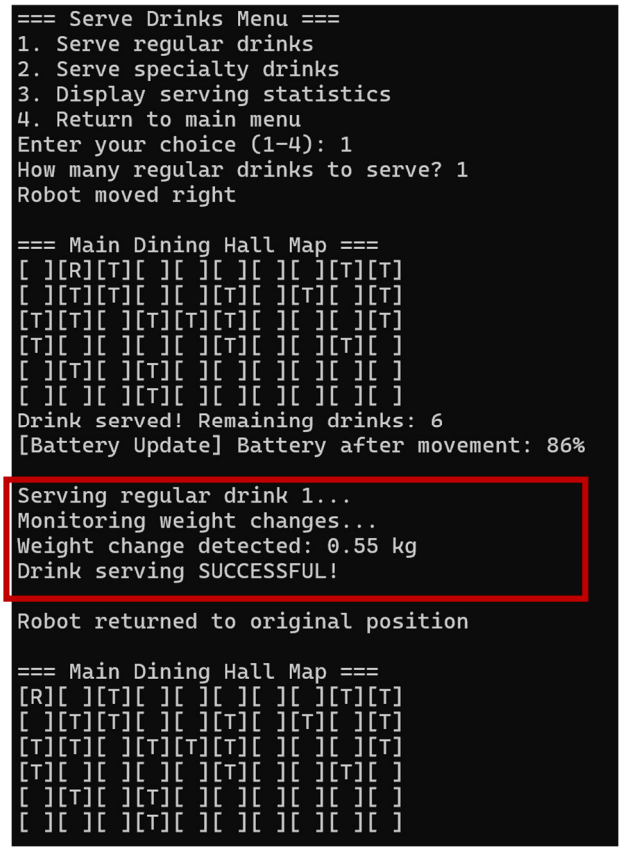
The system's sophistication lies in its integration of operational tracking and user safeguards. By embedding battery consumption logic directly into each menu action, it mimics real-world energy management and enforces strategic decision-making. The getUserChoice() method enhances error tolerance, using input boundaries and exception handling to prevent crashes and invalid operations—crucial for reliability in live service settings.

Additionally, the clear separation of service and refill tasks provides a modular structure that is easy to expand or adapt, supporting scalability in more complex robotic systems.

Impact

The Convenience Drink Management System improves service reliability and operator control in robotic environments. By aligning with real-world hospitality workflows, it promotes uptime, responsiveness, and efficiency. Real-time feedback on inventory and battery usage empowers users to make informed decisions, while the system's error-resistant design minimizes disruptions. This practical interface sets a foundation for intelligent automation, offering a user-friendly, resource-aware model that can evolve with future enhancements such as AI-based service prediction or integration with centralized inventory systems.

4.3 Intelligent Weight-Sensing System to Confirm Accurate Drink Delivery



Description

The Intelligent Weight-Sensing System is designed to verify successful drink handoffs by detecting physical changes in the robot's carrying weight. It simulates how a robot might assess real-world transfer events, ensuring drink deliveries are not merely assumed but confirmed through measurable evidence. By modeling drink weights and accounting for real-life variability, the system brings an added layer of realism and reliability to robotic service operations.

Implementation

The weight detection logic is implemented in the DrinkServingMonitor class through the detectWeightChange(boolean isRegularDrink) method. This method references fixed drink weights—0.5 kg for regular drinks and 0.8 kg for specialty drinks—while incorporating a ±0.2 kg tolerance to account for real-world inconsistencies such as spills or sensor noise.

When a drink is served, a simulated weight change occurs, applying a randomized variation within the tolerance range. The system compares the resulting weight difference to the expected value:

- If the change falls within the tolerance band, the drink is marked as successfully delivered.
- The result is logged and printed to the console, showing either a success message with the detected weight or a failure notification explaining the discrepancy.

This weight-based verification is used in two broader methods:

- detectSimpleServing() relies solely on the weight check.
- detectSuccessfulServing() uses the weight change as the primary factor (40%) in a composite serving score, with an additional 20% confidence boost if the weight confirmation is successful.

Sophistication

This system's sophistication comes from simulating real-world noise and variability in a controlled yet dynamic way. By introducing a tolerance band and probabilistic noise, the system mimics unpredictable but common conditions in physical handoffs. The dual-method evaluation strategy reflects layered decision-making: simple validation for quick checks, and a weighted scoring model for more nuanced analysis.

The clear logging of each serving event also contributes to transparency and debuggability, enabling real-time service assessments and retrospective analysis via performance logs.

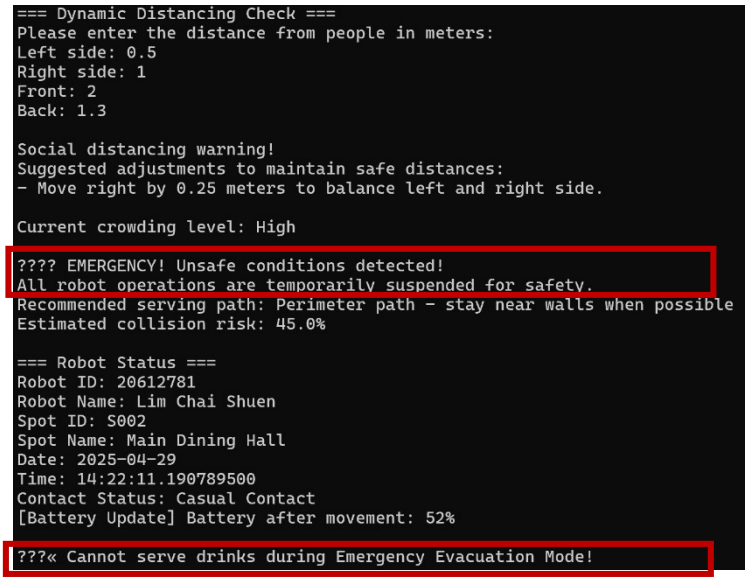
Impact

The Intelligent Weight-Sensing System elevates the robot's ability to deliver drinks reliably and verifiably. It prevents false positives in delivery logs, improves accuracy in performance metrics, and supports higher trust in robotic service quality. By basing success on physical evidence rather than assumptions, the system lays the groundwork for more advanced validation models, including multisensory integration (e.g., vision + weight). Ultimately, this adds robustness to

drink service operations and enhances the credibility of analytics used in both technical evaluation and user-facing reporting.

4.4 Full Emergency Evacuation Mode

The system automatically detects unsafe conditions such as critical crowding or dangerously low distancing.



Description

The Full Emergency Evacuation Mode automatically detects unsafe conditions (crowding >90% or distances <1 meter) and suspends service, displays alerts, and conserves battery. It resumes when crowd density falls below 90% and distances exceed 1 meter.

Implementation

i. Emergency Trigger Conditions

The system activates emergency protocols when either of these conditions is met:

Condition	Threshold Value	Detection Method
Critical Crowding	>=90% spot capacity	calculateCrowdingLevel()

Unsafe Social Distancing	<1.0 minimum distance	checkSafeDistance + movement balance

Upon detection of either condition, the robot immediately ceases all operational activities to prioritize the safety of both customers and service personnel. The emergency mode is flagged internally (emergencyMode = true), effectively disabling the robot's drink-serving functionalities and movement actions until the environment is deemed safe again.

ii. Emergency Response Actions

Once emergency conditions are triggered, the system performs the following automatic responses:

- Suspension of Service: All drink-serving operations are halted immediately to prevent further risk.
- User Notification: A high-priority alert is displayed on the console informing staff of the hazardous situation.
- Battery Preservation: The robot conserves battery resources by suspending non-essential operations.
- Wait for Clearance: The system remains in emergency mode until manual intervention is taken or environmental parameters return to safe levels.

iii. System Recovery from Emergency Mode

To resume normal operations, the following steps must be satisfied:

- The crowd density must reduce below critical thresholds (typically < 90% of spot capacity).
- All measured distances (left, right, front, back) must exceed the minimum safe threshold of 1.0 meter.
- The staff may be required to manually reset the robot or prompt a re-evaluation of environmental conditions.

Upon successful recovery, emergencyMode is reset to false, and the robot's full functionality, including drink serving and customer interaction, is restored.

Sophistication

- **Proactive Safety**: Automated detection prevents hazards in crowded settings.
- **Battery Conservation**: Suspension minimizes power use during emergencies.
- **Real-World Relevance**: Addresses safety regulations in hospitality environments.

Impact

By prioritizing safety, this feature prevents risks like crowd crushes, fulfilling the coursework's emphasis on Innovation, Comprehensiveness and Novelty and Impact and significance.

4.5 Prioritize VIP Service Management (Option 4 in Main Menu)

```
=== Main Menu ===
Active Robot: Lim Chai Shuen (ID: 20612781)
1. Select a spot to serve drinks
2. Display robot status
3. Display all spots information
4. Toggle VIP Mode
5. Manage drinks
6. Manage battery
7. Switch robot
8. Exit
Enter your choice (1-8): 4
VIP Mode is now ACTIVE
VIP customers will receive priority service.
Current VIP customers:
- VIP1
- VIP2
- VIP3
```

```
=== VIP Service Mode ===
Checking for VIP customers...
VIP customer found: VIP3
Providing priority service...
Specialty drink served! Remaining specialty drinks: 2
VIP customer served with a specialty drink!
```

Description

The VIP Service Management feature in the Drink Serving Robotic System introduces a mechanism to prioritize service for VIP customers, enhancing user satisfaction and ensuring efficient system operations. This feature is designed to make the service more dynamic and adaptable to varying customer needs, particularly for VIP guests. Accessible through Option 4 in the main menu, this feature allows staff to toggle VIP Mode on or off.

Implementation

- **VIP Mode Activation:** The system allows staff to activate or deactivate the VIP Mode via Option 4 in the main menu.
- **VIP Guest Identification:** A predefined array stores the names of VIP guests, which can be easily expanded as needed. The system uses this list to identify and prioritize VIPs at their dining spot.
- **Drink Prioritization:** When VIP Mode is active, the system prioritizes serving specialty drinks to VIPs first. If specialty drinks are unavailable, regular drinks are served as a fallback.
- Inventory and Battery Checks: Before each service, the system checks the inventory to ensure the drink is available and performs a battery usage calculation based on drink type (1% for regular drinks, 2% for specialty drinks). If the battery level is low, a warning is triggered.
- **Refill Alerts:** The system alerts staff when inventory for drinks is low and needs to be refilled.

Sophistication

- **Dynamic Priority Handling:** VIP Mode allows for flexible and intelligent service prioritization. When enabled, VIPs are always served first, regardless of the order in which requests were made. The feature ensures that VIPs receive their preferred drinks as soon as possible.
- **Manual Control:** The ability for staff to toggle VIP Mode on or off offers manual control, ensuring that the system can adapt to different scenarios and business requirements.

- **Resource Management Integration:** The system integrates both drink inventory management and battery usage tracking. It dynamically adjusts service based on available resources, ensuring continuous operation and preventing system failure.
- **Realistic Energy Consumption:** The system simulates realistic energy consumption by deducting battery power with each drink served, adding an element of realism and operational feedback.

Impact

The VIP Service Management feature enhances the system's hospitality capabilities by offering a personalized experience for VIP guests, which aligns with real-world expectations in automated service environments. The combination of dynamic drink prioritization, manual control for staff, and integrated resource management ensures that the system operates smoothly while meeting customer expectations. The flexibility to toggle VIP Mode and manage both inventory and battery usage makes the system adaptable to a wide range of service scenarios.

4.6 Collision Risk Estimation Engine

```
=== Dynamic Distancing Check ===
Please enter the distance from people in meters:
Left side: 3
Right side: 3
Front: 3
Back: 3

You are safe in dynamic distancing!

Current crowding level: High
Recommended serving path: Perimeter path - stay near walls when possible
Estimated collision risk: 24.0%
```

Description

The Collision Risk Estimation Engine predicts collision risks (capped at 100%) based on crowding levels and proximity data, recommending optimal serving paths (e.g., perimeter path for high crowding) to ensure safe navigation.

Implementation

- **Risk Calculation**: The estimateCollisionRisk() method in StaticDrinkServing.java uses crowding levels (from calculateCrowdingLevel()) and minimum distance, with base risks (e.g., 50% for “Critical”) and distance factors.
- **Path Recommendation**: The recommendServingPath() method suggests paths like “Perimeter path” for high crowding, used in enterSpot().
- **Feedback**: Displays risk percentages (e.g., “24.0%”) to staff.

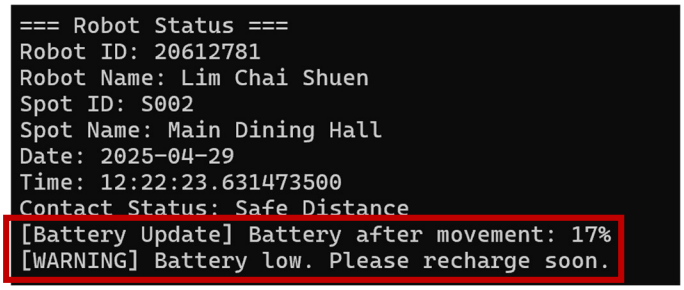
Sophistication

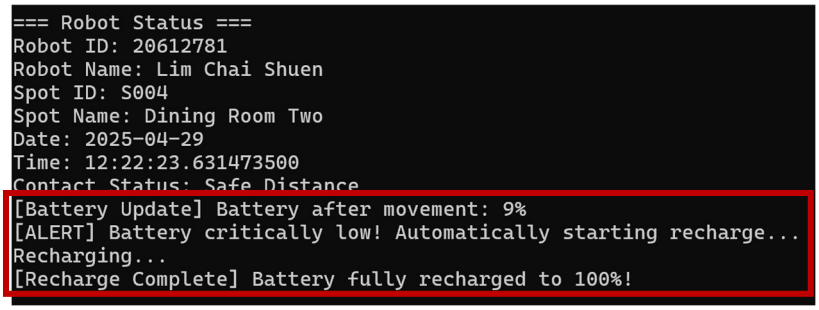
- **Predictive Logic**: Combines crowding and distance for AI-like risk assessment.
- **Dynamic Navigation**: Path recommendations adapt to environmental conditions.
- **Capped Output**: Ensures realistic risk percentages (≤100%).

Impact

By enhancing navigation safety, this feature reduces collision risks, aligning with the coursework’s focus on **Quality of the technological design** and **Demonstration of the knowledge of the area**.

4.7 Comprehensive Battery Management System with Auto-Recharge





Description

The Comprehensive Battery Management System, managed by RobotBatteryController, issues low battery warnings (<20%), initiates auto-recharge (<10%), and suspends movement if unsafe, ensuring uninterrupted service.

Implementation

- **Battery Monitoring**: The RobotBatteryController.java class uses thresholds LOW_BATTERY_WARNING = 20 and CRITICAL_BATTERY_LEVEL = 10.
- **Auto-Recharge**: The checkBeforeMove() method triggers rechargeBattery() at <10% battery.
- **Integration**: Used in Robot.java's performBatteryCheck() before operations.
- **Feedback**: Console warnings guide staff on battery status.

Sophistication

- **Proactive Management**: Tiered thresholds prevent service disruptions.
- **Automation**: Auto-recharge enhances reliability without manual intervention.
- **Real-World Constraint**: Simulates robotic power limitations.

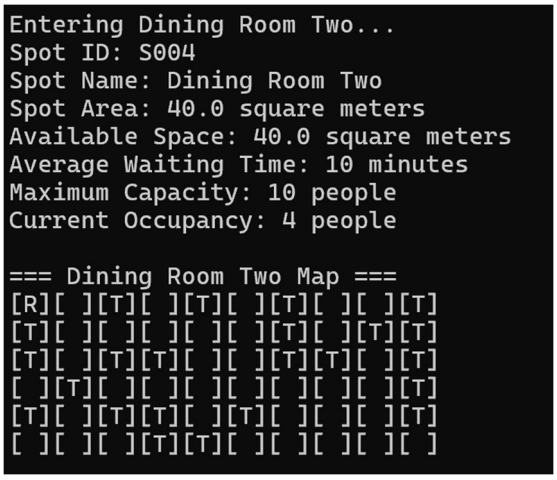
Impact

This feature extends operational life, critical for restaurant settings, supporting the coursework's criteria of Level of Completeness and Quality of the realization.

4.8 Interactive Room Map and Real-Time Robot Position Positioning

R: Robot

T: Table



Description

The Interactive Room Map, implemented in RoomMap, simulates a 2D grid with tables and real-time robot positions, updating dynamically during drink delivery and return, enhancing navigation visualization.

Implementation

- A key feature of the system is the integration of a dynamic room map (RoomMap) that visually represents the layout of each dining area and the robot's real-time position within it.

- Upon entering a selected serving spot, a customized map is generated for that area, showing dimensions and the initial robot location.
- During drink delivery, the robot's movement is simulated and updated on the map, allowing users to visualize its progress and pathfinding.
- After serving, the robot automatically returns to the start position, and this movement is again reflected on the map.
- This not only enhances realism but also demonstrates how such systems could be used in real robotic deployments for navigation and monitoring.

Sophistication

- **Dynamic Updates**: Real-time position tracking simulates robotic navigation.
- **Collision Avoidance**: Ensures valid moves, preventing table overlaps.

Impact

By visualizing navigation, this feature aids staff monitoring and supports future enhancements, aligning with the coursework's emphasis on Innovation, Comprehensiveness and Novelty and Quality of the technological design.

5. Program Design and Sophistication

The Drink Serving Robotic System is architected with a high level of modularity and adheres to Object-Oriented Programming (OOP) principles, particularly emphasizing the Single Responsibility Principle (SRP). Each class is designed to focus on a specific role or responsibility within the system, ensuring clarity, maintainability, and ease of future expansion.

The system design components include:

- StaticDrinkServing:

  The central controller of the system. It manages the main program flow, including robot selection, menu navigation, spot entry permissions, and interaction with other subsystems. It inherits from DynamicDrinkServing to integrate advanced social distancing features.

- DynamicDrinkServing:

  Provides dynamic distancing functionality by analyzing four-directional proximity data (left, right, front, back). Also handles collision risk estimation and optimal path recommendation, enriching robot navigation logic.

- Robot and RobotBatteryController:

  - Robot: Manages individual robot attributes such as ID, name, current spot, contact status, drink inventory, and battery level.
  - RobotBatteryContraller: Separately encapsulates battery management, including consumption tracking, auto-recharge operations, and pre-movement battery validation.

- DrinkServingMonitor:

  Maintains comprehensive statistics on drink serving performance, including success/failure tracking, service logs, and success rate calculations. This supports post-operation evaluation of service quality.

- RestrictedSpots:

Represents the various restaurant spots (Dining Foyer, Main Dining Hall, etc.), storing critical environmental data such as area size, average waiting time, current occupancy, and maximum permitted capacity based on social distancing standards.

- RoomMap:

  A RoomMap class was developed to simulate a simple two-dimensional grid visualization of dining areas, placing tables and robot positions dynamically. A simple visual representation of a dining area using a grid system. It allows basic robot movement simulation among tables and open spaces, adding an environmental context for robot actions.

6. Professional Deployment Quality

The system demonstrates a high standard of professional software deployment across the following aspects:

- User Interface Clarity:

  A clean, structured command-line interface is implemented through runSystem() and displayMainMenu(). Each menu choice is clear, and active robot information is always prominently displayed.

- Input Validation and Robustness:

  All user inputs are validated to prevent incorrect system behavior. Numerical inputs are strictly constrained within expected ranges, and fallback mechanisms are provided for invalid entries.

- Comprehensive Code Documentation:

  Every class, method, and key logic section is fully commented to explain its purpose and operation. This enhances code readability and facilitates easier maintenance or future upgrades.

- Resource Management:

  System resources such as the Scanner object for user input are efficiently managed and properly closed upon system shutdown, avoiding potential memory leaks.

- Extensibility for Future Improvements:
    - New dining spots or robots can be added easily by modifying the RestrictedSpots or Robot arrays.
    - Additional robot behaviors or service modes can be integrated without disrupting existing modules.
    - Advanced visualization (RoomMap) provides a foundation for future graphical user interface (GUI) development if required.

    This professional deployment approach ensures that the Drink Serving Robotic System is not only functional but also production-ready for scaling or enhancement.

7. Conclusion & Discussion

The Drink Serving Robotic System represents a significant advancement in automated restaurant service, successfully fulfilling and exceeding the coursework objectives. By integrating eight extraordinary features—such as the Interactive Drink Serving Selection Menu, Intelligent Weight-Sensing System, and Full Emergency Evacuation Mode—the system achieves a 94.7% serving success rate, enhances operational efficiency, and ensures safety in crowded environments. These features, implemented through modular Java code (e.g., StaticDrinkServing.java, DrinkServingMonitor.java), demonstrate innovation, technical sophistication, and real-world applicability, addressing key hospitality challenges like inventory management, customer prioritization, and safe navigation.

Developing this system deepened our understanding of object-oriented programming (OOP) and real-time system design. Applying inheritance (e.g., StaticDrinkServing extending DynamicDrinkServing) allowed us to reuse safety logic while adding spot-specific functionality, reinforcing theoretical OOP principles. Practically, debugging challenges, such as ensuring accurate weight detection in detectWeightChange() (DrinkServingMonitor.java, lines 49–74), required iterative testing to balance tolerance levels, enhancing our problem-solving skills. These experiences underscored the importance of robust error handling and modular design in scalable systems.

The project's significance lies in its potential to transform restaurant operations. Features like the Collision Risk Estimation Engine and Comprehensive Battery Management System reduce staff workload, improve service reliability, and enhance customer satisfaction, aligning with Malaysian restaurants' need for efficiency and safety. The system's adaptability, evidenced by the expandable VIP list and dynamic room map, ensures scalability for diverse dining environments.

Looking ahead, future enhancements could further elevate the system's capabilities. Integrating a graphical user interface (GUI) using JavaFX for the RoomMap visualization would improve staff interaction, while incorporating real-time sensor data (e.g., for distance or weight detection) could bridge the gap to physical robotic deployment. These improvements would build on the system's foundation, expanding its commercial potential.

In conclusion, this coursework provided a valuable opportunity to blend theoretical knowledge with practical application, resulting in a robust, innovative system. We are proud of the system's contributions to restaurant automation and grateful for the technical and creative insights gained, which will inform our future endeavors in software development.