Tiago Shop Assistant

[Elective in A.I. Report]

Authors: [Federico Matarante], [Lorenzo Barbieri], [Leonardo Sandri]
Student IDs: [2133034], [2144036], [2137374]
Course: Elective in A.I.
Instructors: L. Iocchi, V. Suriani
Work has been divided equally between members and each member has given important contribution

August 4, 2025

Contents

1 Introduction

1.1 The Challenge of Social Robotics in Retail

The retail industry stands at the intersection of technological innovation and human-centered service, where the integration of autonomous robots presents both unprecedented opportunities and complex challenges, moreover from ethical points of view. Unlike traditional industrial automation, retail robotics demands systems that can navigate the nuanced landscape of human social interaction while maintaining the reliability and safety expected in public spaces. The challenge extends beyond mere task completion to encompass social awareness, contextual understanding, and the ability to provide genuinely helpful assistance that enhances rather than replaces human service. In order to address this last ethical topic we aimed at improving staff member performances by providing help (e.g. providing a storage of informations on the shop, helping understand customer needs) in particular implementing a comprehensive LLM that can overcome communication boundaries between staff, clients, and robot.

1.2 Design

Ideal Robot Concept Our initial vision for the Tiago Shop Assistant was centered around creating a comprehensive robotic system capable of enhancing the retail experience through advanced perception, reasoning, and human-robot interaction. We envisioned a robot that would go beyond basic navigation and task execution, featuring sophisticated computer vision for real-time customer behavior recognition, advanced demographic classification, and a comprehensive knowledge graph to understand and predict customer needs. This ideal robot would have distinguished between staff and clients through facial recognition, adapted its behavior to various customer types such as hurried shoppers or curious browsers, and provided proactive assistance based on behavioral analysis. The system would have maintained detailed memory structures of customer interactions, enabling personalized recommendations and building long-term relationships with returning customers. Additionally, it would coordinate seamlessly with human staff to create a truly integrated service experience. However, due to technical and infrastructural limitations including the lack of access to a real retail environment, restricted hardware capabilities, and time constraints, we were unable to fully implement components like real-time customer behavior recognition and advanced computer vision for demographic classification. Instead, we strategically leveraged Large Language Model capabilities to handle many of these advanced features through direct natural conversation, maintaining the robot's intelligence and helpfulness while adapting to practical constraints.

1.3 Related Work

Our Tiago Shop Assistant project took inspiration from established research in human-robot interaction, social robotics, and conversational AI systems. We studied in class key concepts from proxemics theory. The course lecture slides on "Motion Control for HRI" by L. Iocchi provided knowledge on social force models and human-robot spatial interaction principles that guided our navigation system design. For the conversational component, we drew from current approaches in Large Language Models and Retrieval-Augmented Generation (RAG) architectures to create a system capable of maintaining context and building customer profiles over time. Our computer vision module adapted standard person detection techniques using HOG descriptors and AprilTag identification for robust staff/customer classification. The overall system architecture follows established patterns in behavior-based robotics, integrating perception, reasoning, and action modules through a finite state machine approach. Throughout development, we maintained focus on human-augmentation rather than replacement principles, ensuring our assistant enhances rather than substitutes human staff capabilities. For more in-depth technical details and comprehensive references to the foundational works that guided our development, please refer to the References section.

1.4 Project Vision and Objectives

This project addresses the fundamental question: *How can we create an autonomous robotic system that not only performs retail assistance tasks but also learns from and adapts to human interactions in real-time?* Our solution is embodied in a comprehensive TIAGo robot system designed for a simulated sports apparel store environment, where the robot serves as an intelligent shop assistant capable of autonomous navigation, natural language interaction, and dynamic behavioral adaptation.

The primary objectives of this work are threefold:

1. **Intelligent Navigation**: Develop a safety-aware motion planning system that prioritizes human comfort and safety while maintaining operational efficiency in dynamic retail environments.
2. **Adaptive Human Interaction**: Create a conversational AI system that can understand customer needs, provide relevant information, and build personalized relationships through continuous learning from interactions.
3. **Contextual Reasoning**: Implement a unified reasoning framework that orchestrates the robot's behavior based on real-time environmental perception and social context awareness.

1.5 Technical Innovation and Approach

The core innovation of our approach lies in the integration of a **dynamic database** with a **Retrieval-Augmented Generation (RAG) system**, enabling the robot to build and maintain personalized customer profiles through natural conversation. This architecture allows the robot to recall previous interactions, understand customer preferences, and provide increasingly tailored assistance over time—a capability that distinguishes our system from traditional static chatbot implementations.

Our technical framework combines several state-of-the-art components:

- **Custom Safety-Aware Path Planning**: A modified A* algorithm that incorporates **human comfort zones** and social navigation principles
- **LLM-Powered Conversational Interface**: Utilizing an LLM for natural language understanding and generation
- **Semantic Knowledge Retrieval**: Embedding-based similarity search for efficient information retrieval from the database
- **Computer Vision Integration**: HOG-based person detection with AprilTag identification for staff/customer classification
- **Finite State Machine Reasoning**: Coordinated behavior management across all system components

1.6 Significance and Impact

The importance of this work extends beyond the immediate retail application. As service robots become increasingly prevalent in public spaces, the ability to create systems that learn from and adapt to human interaction patterns becomes crucial for widespread adoption. Our approach demonstrates how modern AI techniques can be integrated into robotic systems to create more natural, helpful, and socially aware autonomous agents.

Furthermore, the project addresses key challenges in human-robot interaction research, including real-time adaptation, multi-modal integration, and the balance between automation and human-centered service. The dynamic database approach offers a scalable solution for maintaining context across multiple interactions, while the safety-aware navigation system addresses critical concerns about robot operation in human-populated environments.

1.7 Report Organization

This report presents a comprehensive analysis of our integrated robotic shop assistant system. The navigation and motion control systems are presented first, establishing the foundation for autonomous operation.

We then detail the human-robot interaction components, including the conversational AI system, knowledge management, and vision-based person detection.

The reasoning module that coordinates these components is described, followed by experimental results and evaluation of system performance. Finally, we conclude with lessons learned and directions for future development.

Through this integrated approach, we demonstrate that contemporary AI and robotics techniques can be effectively combined to create autonomous systems that are not only functionally capable but also socially intelligent and adaptable to human needs.

2 Environment and navigation

For this project we have been given a particular care to the Navigation and Motion Control System, which provides the robot with the foundational capabilities of spatial awareness, safe path planning, and reliable motion execution. This system ensures that the robot can move autonomously and intelligently within the store environment. The architecture is composed of three integrated parts: the simulated environment, a custom safety-aware path planner, and the ROS 2 nodes that manage high-level navigation logic.

2.1 World and Map Generation

To simulate a sports apparel store we have decided to build a **custom environment**. We constructed this in two distinct but related stages: a 3D world for realistic physics and sensor simulation in Gazebo, and a corresponding 2D occupancy grid map for the navigation algorithms.

Gazebo Simulation World A virtual model of the retail store was meticulously built as a Gazebo world file. We wanted it to be representative of a real-world store while being built in a simplified structure. For this reason using simple geometric primitives, we constructed:

- **Outer Walls:** Defining the boundaries of the operational area.
- **Shelves:** A series of rectangular shelves and pillars arranged to create distinct aisles and open spaces. This layout forces the planner to navigate both narrow corridors and wide-open areas.
- **Dressing room:** A narrow and tall structure represented in green.
- **Cash desk:** A counter positioned near the entrance/exit and recognizable from its blue color.

This 3D world, shown from two perspectives in Figures 1 and 2, serves as the "ground truth" for the simulation. The robot's simulated sensors (like its laser scanner) interact with these 3D models to produce realistic data, which is essential for testing localization and mapping.

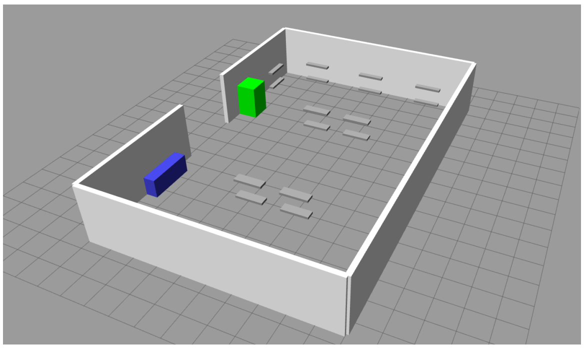

Figure 1: Oblique perspective of the Gazebo world, illustrating the 3D structure of the store layout and obstacles.

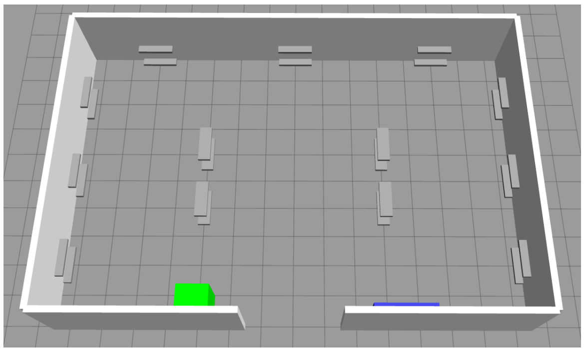

Figure 2: Top-down view of the Gazebo world, clearly showing the arrangement of shelves and open floor plan.

2D Occupancy Grid Map While the Gazebo world is for simulation, the navigation stack operates on a more abstract, 2D representation. We generated this map using the ROS 2 `slam_toolbox` package, a standard tool for Simultaneous Localization and Mapping (SLAM). The process involved:

1. Launching the simulation with the robot and the `slam_toolbox` node.
2. Manually teleoperating the robot throughout the entire Gazebo world, allowing it to build a consistent map from its simulated laser scans.
3. Saving the completed map, which results in two files:
   - **'my_map.pgm':** A grayscale image where black pixels represent occupied space (obstacles), white pixels represent free space, and gray represents unknown territory (Figure 3).
   - **'my_map.yaml':** A metadata file containing crucial information like the map's resolution (meters per pixel) and its origin in the world frame.

This 2D map is the fundamental data structure used by our path planner. To facilitate autonomous behaviors like searching for customers, we defined a set of strategic waypoints on this map (Figure 4), which are used for the robot's "random walk" routine.

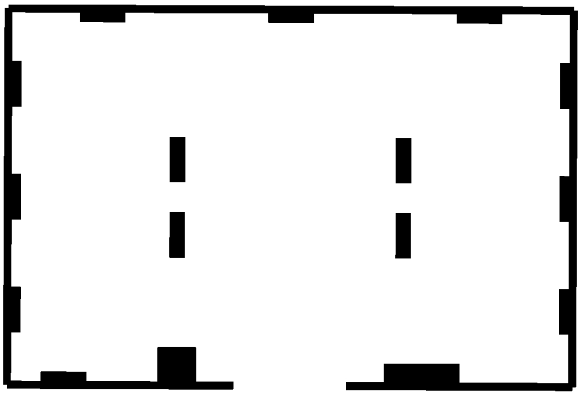

Figure 3: The final 2D occupancy grid map generated via SLAM, serving as the input for path planning.

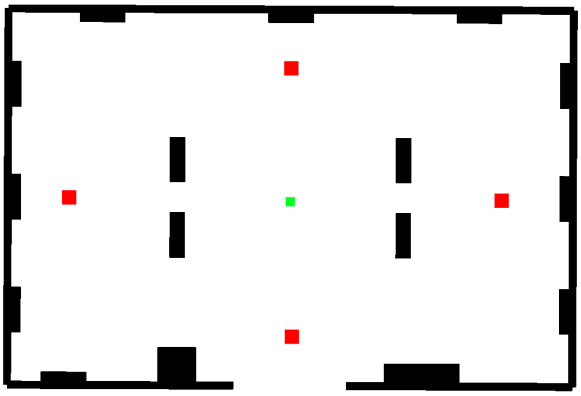

Figure 4: The occupancy map overlaid with predefined waypoints (red dots) for the robot's random walk behavior. In green, the robot's origin

3 Implementation

The system has been designed by implementing different independent modules:

- **Path planner**: handles the phisical path planning of Tiago. It's closely linked to a **Motion control** module that powers the planned movements.
- **Data interface**: handles the data organization, retrieval and update. Also some semantic retrieval techniques have been implemented.
- **Conversation module**: handles the conversational part of the system, implementing a real chatbot powered by a RAG system.
- **Vision module**: handles the vision part of the system, in particular object and person recognition.

All of the previous are linked together by a **Reasoning module**, which creates a bridge between the four systems described above and implements some state-based reasoning logic.

3.1 Path planner - HBR

The control system presents a custom path planner to enhance customer security combined with a well performing velocity command controller.

3.1.1 Safety-Aware A* Path Planner

A key component of our solution is a custom path planner, encapsulated in `astar_path_planner.py`. Standard planners often generate paths that are valid but unnatural, hugging walls or cutting corners too tightly. Our goal was to create a planner that generates paths that are not only collision-free but also **feel safer and more predictable from a human perspective**. We achieved this by fundamentally modifying the A* algorithm's cost function.

The evaluation function for any node n in the search space is defined as:

$$f(n) = g(n) + \text{safety\_cost}(n) + h(n)$$

where:

- $g(n)$ is the standard path cost (distance) from the start node to node n.
- $h(n)$ is the admissible heuristic, which is the Euclidean distance from node n to the goal.
- **safety_cost**(n) is our added penalty term designed to enforce a safety margin.

To efficiently calculate this safety cost, we first pre-process the map to create a **distance grid**. Using the `distance_transform_edt` function from the SciPy library, we compute for every free cell in the map its exact Euclidean distance to the nearest obstacle. This transform is performed once during initialization.

During pathfinding, within the `find_path` method, we introduce two parameters: `safety_radius` and `safety_weight`. For each potential node being considered, we look up its distance from the nearest obstacle in our pre-computed grid. If this distance is less than the `safety_radius`, a penalty is applied. The magnitude of this penalty is inversely proportional to the distance from the wall and is scaled by the `safety_weight`. This mechanism effectively creates a "potential field" that pushes the path away from obstacles, favoring routes through the center of corridors, as illustrated by the planned path in RViz (Figure 5).

Finally, the raw, grid-based path from A* is often irregular. For this reason a `smooth_path` function uses Cubic Spline interpolation to generate a new set of waypoints, resulting in a continuous, smooth trajectory that is more kinematically feasible for the robot to follow.

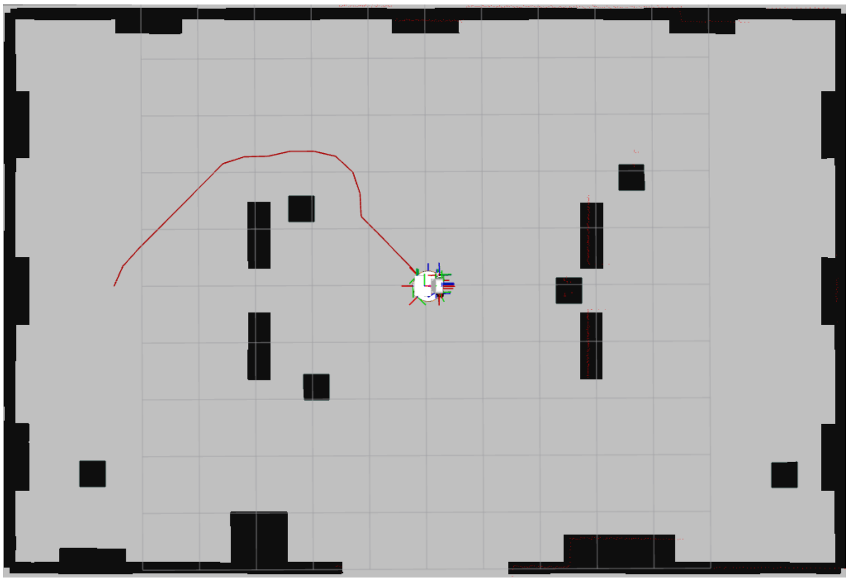

Figure 5: A path generated by our safety-aware A* planner, visualized in RViz. The red line demonstrates the planner's preference for staying in the center of the aisle, away from the black obstacles.

3.1.2 ROS 2 Navigation and Control System

The path planning logic is integrated into the broader robotic system through a pair of ROS 2 nodes that handle service requests, path execution, and state management.

PathPlannerService ('path_planner.py') This node acts as a service-oriented wrapper around our custom `AStarPathPlanner`. Its primary role is to expose pathfinding capabilities to other parts of the system, particularly the Reasoning Module.

- **Service Endpoints:** It provides a standard `/plan_path` service (of type `nav_msgs/srv/GetPlan`) for compatibility, but more importantly, it offers a custom `/path_planner_command` service (`tiago_srv/PathPlannerCommand`). This custom service allows the Reasoning Module to issue high-level commands like `"random_walk"` or `"go_to_location"`.

- **Behavior Logic:** When it receives a `"random_walk"` command, it executes a sequence of actions: it uses the TF2 library to get the robot's current position in the map frame, selects a valid and reachable waypoint from its predefined list, and then invokes the internal `AStarPathPlanner` to compute a safe path. The resulting path is then published as a `nav_msgs/msg/Path` message to the `/planned_path` topic.

SimpleNavigationController ('controller.py') This node is the executor of the planned motion. It subscribes to the `/planned_path` topic and is responsible for commanding the robot to follow the waypoints.

- **Action Client:** It is an action client for Nav2's powerful `navigate_to_pose` action server. This allows it to send goals to the underlying navigation stack, which handles low-level motion control and obstacle avoidance.
- **State Machine and Execution Logic:** The controller is built around a robust state machine (`IDLE, NAVIGATING, RETRY_WAIT, MOVING_TO_NEXT`) that manages the entire navigation lifecycle. When a new path is received, it transitions to a navigating state. For each waypoint in the path, it intelligently computes a target orientation that faces the *next* waypoint, ensuring smoother transitions and turns.
- **Error Handling and Safety:** A key feature is its resilience. If Nav2 fails to reach a waypoint, the controller doesn't give up immediately. It enters a retry loop, attempting to reach the goal up to `max_retry_attempts` times. For safety, it provides a `/controller/stop_trigger` service. When this service is called (e.g., by the Reasoning Module), it immediately cancels the current Nav2 goal and, critically, publishes a zero-velocity `Twist` message directly to `/cmd_vel`. This ensures an instantaneous physical stop, which is a vital safety precaution.
- **Feedback Loop:** Upon successful completion of a path, or if it's stopped, the controller publishes a status message (e.g., `"finished", "stopped"`) to the `/path_planner_status` topic. This closes the loop, informing the Reasoning Module of the outcome of its motion command.

In summary, these components work in synergy to deliver a complete navigation solution. The planner generates safe and intelligent paths, and the controller ensures their robust and safe execution, providing a solid foundation for all of the robot's physical movements within the store.

3.2 Chatbot integration - HRI

The Human-Robot Interaction component of our shop assistant system leverages a Large Language Model (LLM) to facilitate natural language communication with users. The system employs a sophisticated architecture that combines conversational AI capabilities with knowledge retrieval through a Retrieval-Augmented Generation (RAG) system backed by a database.

3.2.1 Chatbot

For the conversational interface, we implemented an LLM-based chat system using the Llama-3.1-8b-instant model accessed through the Groq service. This model serves as the primary communication channel between users and the shop assistant robot, processing natural language input from customers, understanding their queries and requests, and generating appropriate responses in a **conversational manner**.

The Llama-3.1-8b-instant model was selected for its balance between performance and computational efficiency, providing fast inference times suitable for real-time conversation while maintaining high-quality natural language understanding and generation capabilities.
The Chatbot workflow operates through a multi-step process:

1. **Input Processing**: User messages are received and added to the conversation context maintained by the assistant.

2. **Direction Detection**: The model analyzes the conversation history and current user input to determine the user's intent and categorize it into one of five interaction types.

3. **Query Generation**: Based on the detected direction, the model generates specific database queries tailored to retrieve relevant information if needed.

4. **Response Generation**: After receiving database results, the model formulates appropriate responses that combine the retrieved information with *conversational context* (see Appendix 6.4 and 6.5 for more information about conversational context used) and the appropriate system prompt to guide the LLM's response.

The possible directions determine how the request is handled and how the system prompt is designed.
The possible options are:

- **walk_to_product**: When customers request navigation to specific products, the system uses the RAG system to extract the most relevant product. If the product is not available, a *WALK_TO_PRODUCT_1* system prompt is used to guide the answer, which will instruct the assistant to inform the user that the product he's asking for is not available.
  If the product is available, a request will be sent to the reasoning module to make Tiago walk to the desired product. The possible responses determine the system prompt used:
  - "unknown_location": when the reasoning node cannot process the request. In this case the user is informed by using a *WALK_TO_PRODUCT_2* system prompt.
  - "walking_to_area": when the reasoning node successfully processed the request. In this case the user is informed by using a *WALK_TO_PRODUCT_3* system prompt that he will be walked to the requested area. Once arrived, the reasoning node will inform the chatbot with the command "area_reached", so the chatbot will inform the user by being guided by a *WALK_TO_PRODUCT_4* system prompt.

- **walk_to_staff**: the system uses the RAG system to extract the most relevant staff and answers guided by a specific *WALK_TO_STAFF* system prompt. The logic is exactly the same as the "walk_to_product" direction.

- **product_info**: when customers seek information about products, the LLM retrieves a list of candidate products from the RAG system and answers guided by a *PRODUCT_INFO_1* system prompt combined with the `[PRODUCTS_INFORMATION]` - if present - else the answer is guided by a *PRODUCT_INFO_2* system prompt which guide the assistant to inform the user about the absence of the requested products.

- **staff_info**: for inquiries about **staff availability or expertise**, the LLM provides relevant staff information. Similarly to the previous case, the LLM retrieves a list of candidate staffers from the RAG system and answers with a *STAFF_INFO* system prompt combined with the `[STAFF_INFORMATION]` - if present.

- **finished**: When the conversation reaches its natural conclusion, the LLM answers powered by a *CONVERSATION_FINISHED* system prompt, a `CONVERSATION_FINISHED` signal is sent to the controller and the RAG system is used to update the user's knowledge by scanning the entire conversation.
- **default**: When none of the previous cases are recognized, the LLM answers powered by the *DEFAULT* system prompt.

For more details about prompts design and extra information about information structure see Appendix 6.1. The workflow is graphically illustrated in fig. 6.

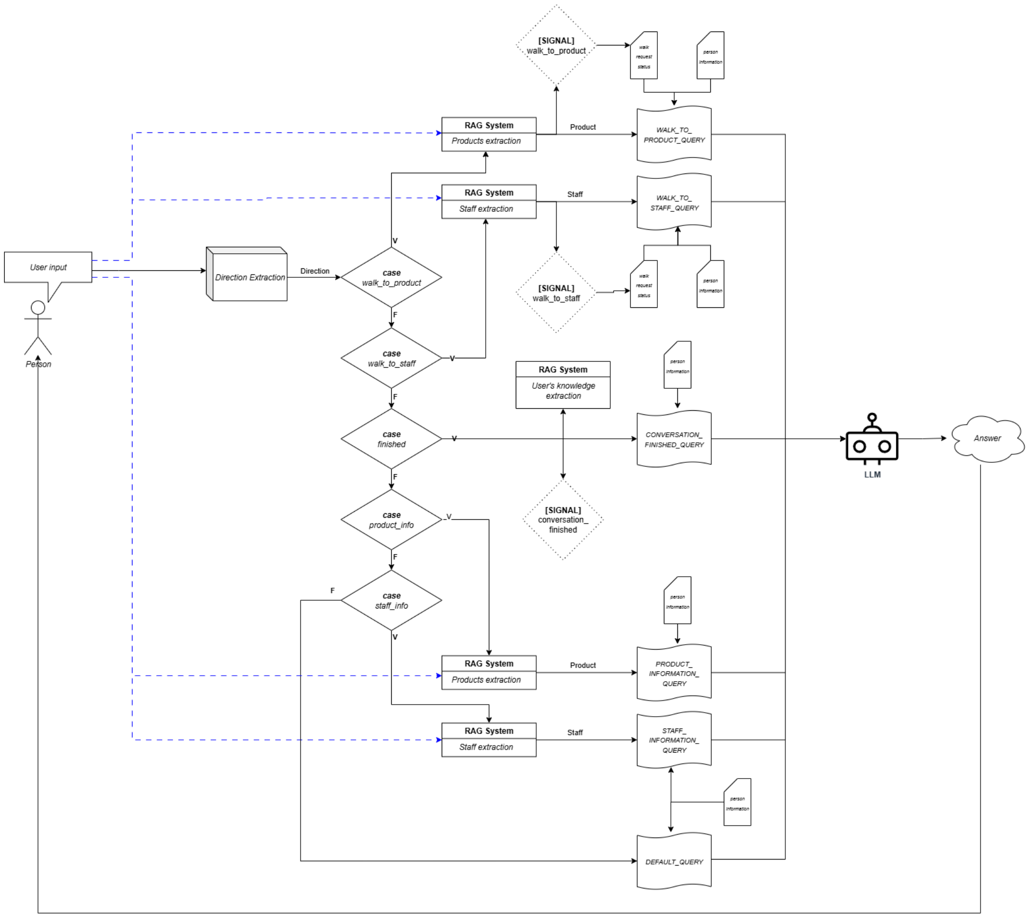

Figure 6: Chatbot workflow

3.2.2 Retrieval-Augmented Generation (RAG) System

The Chatbot is integrated by a designed RAG to extract the relevant information from the database.
The system is designed with a standard architecture, with an LLM that transforms the user's natural language conversation in structured JSON's queries, that are then passed to a retrieval system to extract the relevant information. The process is illustrated in fig. 7.
The workflow is the following:

1. **Query generation**: the user's conversation is transformed to a structured query by using the LLM. See Appendix 6.2 for more details on the queries.
2. **Embeddings generation**: to exploit the semantic relationship between the query and the data, text embeddings are exploited. A pre-trained language model is used for this

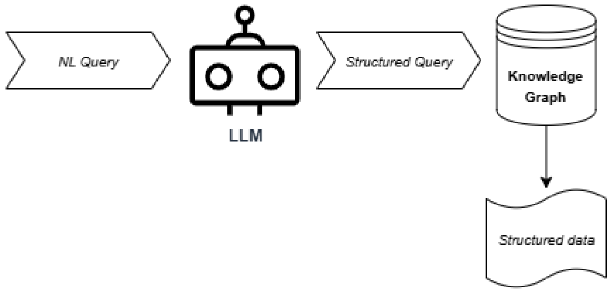

Figure 7: RAG system schema

conversion, transforming sentences into points in a high-dimensional space. The model of choice is *paraphrase-MiniLM-L6-v2* for its lightweight capabilities.
The embeddings of the database data are pre-computed to save time, while the structured query fields are embedded in real time when the database is interrogated.

3. **Data retrieval**: The query embeddings are then compared to the data embeddings to measure similarity. To determine the relevance between a query and stored information, cosine similarity is calculated between their respective embeddings. A threshold and a boolean formulation are used to determine what fields - in particular staff and products - are related to the query. For more information about semantic information retreival see Section 3.3.2.

3.3 Data interface - HBR/HRI

This document details the architectural design for managing and accessing structured and unstructured data within the system. It outlines the core entities stored in the database and the logical mechanisms for information retrieval and updates, particularly leveraging semantic understanding for efficient searching.

3.3.1 Database Entities

The system's database organizes information around three fundamental entities in a relational database: *Products*, *Staff*, and *Customers*. An architectural decision is the inclusion of "embedding" fields for certain textual attributes; these embeddings are high-dimensional numerical representations of text, pre-calculated and stored. This approach significantly reduces the computational time required for similarity comparisons, used during the *semantic search.*

- **Product Entity**

Attribute	**Description**
Product ID	Unique alphanumeric product identifier.
Name	Marketable product name.*
Price	Selling price.
Description	Key features and benefits.*
Category	General classification (e.g., "Footwear").*
Brand	Product brand or manufacturer.*
Sport Category	Targeted sport or activity (e.g., "Running").*

*Also used in embedded representations.

- **Staff Entity**

Attribute	Description
Staff ID	Unique alphanumeric identifier for the staff member.
Name	Full name of the staff member.*
Categories	Areas of expertise or specialization (e.g., "Customer Service", "Equipment Sales").*
Role	Official role or position (e.g., "Sales Associate", "Manager").*

*Also used in embedded (vector) representations.

- **Customer Entity**

Attribute	Description
Customer ID	Unique alphanumeric identifier for the customer.
Name	Customer's name.
Age Category	Age group (e.g., "Adult", "Teenager").
Gender	Gender of the customer.
Preferences	Explicit or inferred preferences (e.g., "eco-friendly products", "specific brands").
Size Information	Customer's size requirements (e.g., shoe or clothing size).
Budget	Typical or stated purchasing budget.
Products of Interest	Summary of products the customer has shown interest in.
Purchase Intent	Estimated likelihood or stage in the purchasing process.
Follow-Up Needed	Boolean flag indicating if follow-up is required.
Conversation Summary	Concise summary of past interactions or conversations.

3.3.2 Querying and Semantic Search Logic

The system's ability to respond to natural language queries hinges on its semantic search capabilities. This involves a multi-step process that transforms an unstructured user query into a precise search across the pre-embedded database entities.
The general search workflow is the following:

1. **Query Embedding**: when a language query is received, the textual components of the query are converted into a embeddings by using a language model.

2. **Semantic Comparison**: the query embedding is efficiently compared against the pre-stored embeddings of relevant attributes (e.g., product names, descriptions, categories, staff roles, staff categories) within the database. This comparison calculates a "similarity score" (using cosine similarity), indicating how closely the meaning of the query aligns with the meaning of the stored attributes.

3. **Relevance Thresholding**: only entities whose similarity score with the query exceeds a predefined "relevance threshold" are considered potential matches. This filters out irrelevant results.

4. **Attribute Filtering**: beyond semantic similarity, additional non-semantic filters (e.g., price range for products, location for products or staff) can be applied to further refine the results.

In the following boxes, the search pseudocode is shown for semantic retrieval for products and staffers.

Product Search Pseudocode

```
FUNCTION SearchProducts(query_parameters):
  1. Initialize: matched_products = [], query_embeddings = [].

  2. Embed relevant textual query fields → query_embeddings.

  3. For each Product in DB:
     a. Get pre-computed embeddings (Name, Description, ...).
     b. For each field in query_embeddings:
        i. Compute similarity with product embeddings.
       ii. If similarity > threshold:
             Add to matched_products (no duplicates), break.

  4. Apply non-semantic filters (e.g., price range).

  5. If product_location_needed:
     a. For each product in matched_products:
        i. Retrieve physical location.
       ii. Determine area name from database → getAreaName(coordinates).
      iii. Add area to location list.

  6. Return top-N matched_products and their locations.
```

Staff Search Pseudocode

```
FUNCTION SearchStaff(query_parameters):
  1. Initialize: matched_staff = [], query_embeddings = [].

  2. Embed relevant textual query fields → query_embeddings.

  3. For each Staff member in DB:
     a. Get pre-computed embeddings (Name, Role, Categories).
     b. For each field in query_embeddings:
        i. Compute similarity with staff embeddings.
       ii. If similarity > threshold:
             Add to matched_staff (no duplicates), break.

  4. If staff_location_needed:
     a. For each staff_member in matched_staff:
        i. Retrieve physical location.
       ii. Determine area name from database → getAreaName(coordinates).
      iii. Add area to location list.

  5. Return top-N matched_staff and their locations.
```

The logic used to update the knowledge about the customer instead is more straightforward:

1. When new information related to a customer is extracted from an interaction (e.g., name, preferences, products of interest, purchase intent, conversation summary), this information is parsed by passing the entire conversation to an LLM and asking it to create new knowledge.

2. The system retrieves the existing customer record using their unique identifier. If no record exists, a new one is initiated.

3. Specific fields within that customer's record are then updated with the newly provided data. This ensures a continuously enriched and accurate customer profile, which can then inform future interactions and personalized recommendations.

3.4 Vision Module - HBR

The Vision Module serves as the primary perceptual component of the robot's social intelligence system, enabling TIAGo to identify and classify individuals in the retail environment. The module implements a multi-stage computer vision pipeline that combines person detection with AprilTag-based identification to support the robot's interaction capabilities.

3.4.1 Functional Architecture

The vision system operates through a cascaded detection architecture:

1. **Person Detection**: HOG (Histogram of Oriented Gradients) descriptor-based human detection using OpenCV's pre-trained classifier
2. **AprilTag Detection**: Tag36h11 family AprilTag detection using the `apriltag` library
3. **Tag-Person Association**: Proximity-based matching algorithm that associates detected AprilTags with person bounding boxes
4. **Classification**: Deterministic classification of individuals as staff or customers based on AprilTag ID mapping

3.4.2 World Modeling Integration

The vision system contributes to the robot's world representation through:

- **Spatial Localization**: Integration with Gazebo's model state system to provide precise 3D world coordinates for detected individuals
- **Identity Persistence**: AprilTag-based unique identification system that maintains consistent person tracking across multiple detections
- **Temporal Coherence**: Anti-spam mechanism with 2-second publication intervals for existing persons to prevent redundant updates

The system publishes `VisionPersonDetection` messages containing person classification (`staff`, `customer`, `unknown`), unique identifiers, and 3D positions that enable the reasoning module to perform distance-based interaction triggering.

3.4.3 Benchmarking Framework

The vision module implements several performance metrics and evaluation strategies:

- **Confidence Thresholding**: AprilTag detection confidence scores (decision margin) are normalized and combined with spatial proximity validation
- **Multi-Modal Validation**: Red region detection provides additional confidence boost for sport equipment AprilTags on colored objects in object detection and recognition
- **Robustness Measures**: Distance-based detection evaluation and angular tolerance testing for person orientation variations

3.4.4 Evaluation Protocol

The vision system evaluation follows a structured approach:

1. **Distance Testing**: Person detection accuracy at 1m, 2m, and 5m ranges
2. **Angular Robustness**: Detection success rates at person orientations from 0° to 90° relative to camera
3. **AprilTag Recognition**: Classification accuracy for staff vs customer identification
4. **False Positive/Negative Analysis**: Systematic evaluation of misclassification rates

The evaluation methodology prioritizes real-world applicability over simulation-specific optimizations, focusing on metrics that translate to physical robot deployment.

3.4.5 Technical Limitations and Design Decisions

Several technical constraints and simulation-specific factors influenced the vision system architecture, leading to strategic design decisions that prioritize system functionality within project scope.

Simulation-to-Real Gap Challenges:

The primary limitation encountered was the simulation-to-real gap in face recognition capabilities. Initial attempts to implement face detection and recognition revealed fundamental incompatibilities:

- **Model Quality Constraints**: Default Gazebo human models possess insufficient facial detail and incompatibilities with Gazebo's lighting model for reliable face detection using standard OpenCV Haar cascades
- **Performance vs. Fidelity Trade-off**: Implementing higher-fidelity human models would require computational resources and is beyond project scope

WSL Development Environment Limitations:

An idea for implementing realistic face detection and recognition, with real faces was considered. We imagined a system that, given a published .msg from the vision controller node, would communicate to the pc to open the webcam, in order to do face detection on our own developers faces, matching embeddings of us, but this option encountered hardware access contraints due to the usage of the Windows Subsystem for Linux (WSL) development environment:

- **Webcam Access Restriction**: WSL cannot directly access Windows hardware webcams, preventing dual-camera approaches
- **USB Passthrough Complexity**: Alternative solutions would require complex USB passthrough configurations beyond project scope.

Strategic Design Adaptations These limitations led to architectural decisions that maintain system effectiveness:

- **AprilTag Substitution**: AprilTag-based identification provides deterministic classification superior to probabilistic face recognition in simulation
- **Model Customization**: Custom Gazebo models with integrated AprilTag badges simulate realistic staff identification systems

Scope and Performance Considerations:

The vision system prioritizes integration effectiveness over exhaustive performance optimization:

- **Angular Sensitivity**: Person detection exhibits reduced accuracy at perpendicular orientations, acceptable given project interaction scenarios
- **Distance Limitations**: Detection range optimized for typical shop interaction distances rather than maximum theoretical performance
- **Evaluation Methodology**: Manual testing approach sufficient for proof-of-concept validation within academic project constraints

These design decisions reflect pragmatic choices that balance technical constraints with project objectives, ensuring system functionality while maintaining pathways for real-world deployment enhancement.

3.5 Reasoning Module - HBR

The `Reasoning module` serves as the brain of the Tiago robot, orchestrating its behavior and managing interactions with various subsystems. The architecture is based as a finite state machine in which the transitions are triggered by the triggers from the other modules of the system; also each transition triggers a series of messages.

3.5.1 States

The module operates in one of three distinct **states**:

- **IDLE**: The robot is not currently busy, so it operates in a random walk, actively searching for people to interact with or objects to scan. It's usually the first state in which the robot is put on while started.
- **CONVERSATION**: The robot is currently engaged in a conversation with a detected person.
- **WALKING_TO_AREA**: The robot is navigating to a specific area, typically initiated during a conversation.
- **APPROACHING_PERSON**: The robot is navigating to approach a specific person.

3.5.2 Transitions and message passing system

As already mentioned, the transition of the states are triggered by messages from the external modules, which in turn result in the emission of other messages.
The possible transitions are:

- **IDLE $\Longrightarrow$ CONVERSATION**: can be triggered by the `PERSON_APPROACH_DETECTED` signal (by the *Vision Module*), which would indicate the fact that Tiago noticed a person approaching; this is triggered when a person distance is lower than a certain threshold. In return, a `stop` signal is sent to the *Motion Module* to make the robot stop walking and a `start_conversation` signal is sent to the *Conversation Module* to make the Chatbot approach in a new conversation.
- **CONVERSATION $\Longrightarrow$ WALKING_TO_AREA**: can be triggered by the `WALK_TO_AREA` signal by the *Conversation Module*, sent when the Chatbot understands that the user wants to be walked to an area of the store. The response trigger is a `go_to_location` message sent to the *Motion Module*, which will make the robot walk to the desired location. Also a `pause_conversation` message is sent to the *Conversation Module*.
- **WALKING_TO_AREA $\Longrightarrow$ CONVERSATION**: can be triggered by the `DESTINATION_REACHED` signal (by the *Motion Module*), which indicates that Tiago has arrived at the desired location. In return, a `resume_conversation` message is sent to the *Conversation Module* to restart the conversation.

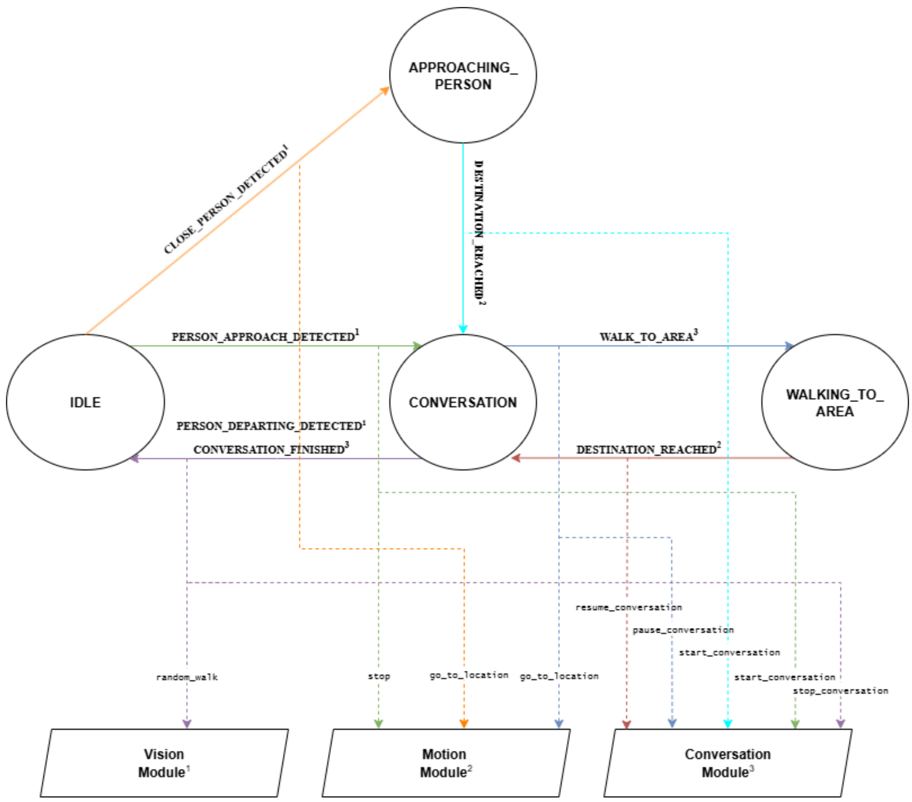

Figure 8: Transitions logic of the reasoning module.
Input signals are indicated in upper case while output signals are lower case. The different colors make clear where the output signals are generated. Also the heading numbers on the input signals make clear what modules have generated them.

- **CONVERSATION ⟹ IDLE**: can be triggered by the `PERSON_DEPARTING_DETECTED` signal (by the *Vision Module*), which indicates that Tiago noticed a person departing, or by the `CONVERSATION_FINISHED` signal (by the *Conversation Module*). In return, a `stop_conversation` message is sent to the *Conversation Module* to make the Chatbot end the conversation and a `random_walk` signal is sent to the *Vision Module* (which would implicitly lead to the robot starting to wander around).

- **IDLE ⟹ APPROACHING_PERSON**: can be triggered by the `CLOSE_PERSON_DETECTED` signal (by the *Vision Module*), which indicates that Tiago noticed a person which is not close enough to begin a conversation but enough to be approached. A `go_to_location` message is sent to the *Motion Module* to begin the walk.

- **APPROACHING_PERSON ⟹ CONVERSATION**: can be triggered by the `DESTINATION_REACHE` signal (by the *Motion Module*). This transition indicates that the robot has successfully approached the desired user and can now engage in a new conversation. A `start_conversation` signal is sent to the *Conversation Module*.

A schema of the transitions is illustrated in fig. 8.
A signal that do not result in transitions in the *Conversation State* is `"unknown_location"`, when the conversation node asks to reach a location which is not known or reachable.

4 Results and Experimental evaluation

In the following sections of the report, the experiments on the components of the project are shown in order to demonstrate its real capabilities. At the end of each one of those, some considerations are done about the limitation due to a lack of budget or time and some approaches have been proposed to overcome those.

4.1 Chatbot evaluation

The following section aims to shows the chatbot interaction examples with customers or staffer. Multiple scenarios are created and it's analyzed the behavior of the assistant highlighting its capabilities.

Note: Some information from the RAG results in this conversation may be omitted for brevity.

Scenario 1: Melissa looking for a gift

A customer - Melissa - is looking for a gift for her son but is not sure about what she's going to buy, so she's going to ask for Tiago's assistance.

RAG Chatbot Conversation - Melissa looking for a gift

Chatbot: Hi, how are you today? What brings you to the shop?

User: Hi! I'm looking for a gift for my son.

Direction: None

Chatbot: We have a variety of gifts, but I'd love to know a bit more about your son. What's his age and is there a particular sport or interest he enjoys?

User: He likes basketball. Do you have any?

Direction: product_info

RAG System Query:


```
{
  "product_id": "",
  "product_name": ["basketball jersey"],
  "product_brand": [],
  "product_category": ["clothes"],
  "product_sport_category": ["basketball"],
  "product_description": ["basketball jersey"],
  "price_range": "",
  "product_location_needed": false
}
```


RAG System Result:

Product ID	Name	Price	Category
red_box_basketball	Official NBA Basketball	$44.99	Sports Equipment
red_box_basketball_hoop	Pro Basketball Hoop	$329.99	Sports Equipment
basketball_shoes_001	Professional Basketball Shoes	$189.99	Sports Equipment
basketball_jersey_001	Official NBA Jersey	$99.99	Sports Equipment
basketball_002	Standard Basketball	$22.99	Sports Equipment

Chatbot: We have a selection of basketball-related products. We have official NBA basketballs, professional basketball hoops, and even basketball shoes.

User: Mh. Basketball shoes could be fun. What models do you have?

Direction: product_info

RAG System Query:


```
{
  "product_id": "",
  "product_name": ["basketball shoes"],
  "product_brand": ["Nike"],
  "product_category": ["footwear"],
  "product_sport_category": ["basketball"],
  "product_description": ["High-quality basketball shoes for performance and comfort"],
  "price_range": "low",
  "product_location_needed": false
}
```


RAG System Result:

Product ID	Name	Price	Category
basketball_shoes_001	Professional Basketball Shoes	$189.99	Sports Equipment
tennis_shoes_001	Professional Tennis Shoes	$179.99	Sports Equipment
basketball_shoes_002	Standard Basketball Shoes	$69.99	Sports Equipment
basketball_shorts_002	Standard Basketball Shorts	$16.99	Sports Equipment
tennis_shoes_002	Standard Tennis Shoes	$79.99	Sports Equipment

Chatbot: We have two basketball shoe models available: "Professional Basketball Shoes" and "Standard Basketball Shoes". Let me show you some features of each.

User: I will go for the first kind. Thanks!

Direction: product_info

RAG System Query:


```
{
  "product_id": "",
  "product_name": ["basketball shoes"],
  "product_brand": [],
  "product_category": ["shoes"],
  "product_sport_category": ["basketball"],
  "product_description": [],
  "price_range": "",
  "product_location_needed": false
}
```


RAG System Result:

Product ID	Name	Price	Category
basketball_shoes_001	Professional Basketball Shoes	$189.99	Sports Equipment
tennis_shoes_001	Professional Tennis Shoes	$179.99	Sports Equipment
basketball_shoes_002	Standard Basketball Shoes	$69.99	Sports Equipment
basketball_shorts_002	Standard Basketball Shorts	$16.99	Sports Equipment
tennis_shoes_002	Standard Tennis Shoes	$79.99	Sports Equipment

Chatbot: The "Professional Basketball Shoes" are a great choice. The price is $189.99. Would you like to pay with card or cash?

User: Cash. I'll go see them first. Thanks, bye!

Direction: finished

Chatbot: You're welcome! I'll be here when you're ready. Take your time looking at the shoes.

Updating customer information...

RAG System Query:


```
{
  'products_of_interest': "['professional basketball shoes']",
  'purchase_intent': 'high',
  'follow_up_needed': True,
  'conversation_summary': 'Gift for son, basketball shoes'
}
```


This scenario highlights some important capabilities:

- The assistant is capable of understanding the customer needs and to assist her in satisfying her needs.
- The assistant is capable of understanding the conversation direction.
- The assistant successfully creates queries for the RAG system and utilizes the provided

data in the answers.

- The assistant successfully extracts the relevant information about the current customer, saving it for future interactions.

Scenario 2: Melissa back to the store

Melissa comes back to the store to thank Tiago for the previous assistance.

RAG Chatbot Conversation - Melissa back to the store

Retrieving customer information...

RAG System Result:


```
{
  'customer_info': {
    'name': 'Melissa',
    'age_category': None,
    'gender': None,
    'preferences': 'basketball shoes',
    'size_info': None,
    'budget': None
  },
  'products_of_interest': "['professional basketball shoes']",
  'purchase_intent': 'high',
  'follow_up_needed': True,
  'conversation_summary': 'Gift for son, basketball shoes'
}
```


Chatbot: Hi Melissa, welcome back to our sports shop. What brings you in today?

User: Do you remember what I was looking for?

Direction: None

Chatbot: Of course! You were looking for a gift for your son. You were particularly interested in the Professional Basketball Shoes priced at $189.99. How did he like them?

User: Yes! He really liked them! Thank you very much!

Direction: finished

Chatbot: That's wonderful to hear, Melissa! I'm so glad your son loved the basketball shoes. Thank you for choosing our store. If you need anything else in the future, feel free to come back and see us. Have a great day!

This scenario proves that the chatbot system is capable of remembering previous interactions and to use the stored information to create a continuity between conversations.

Scenario 3: Robert can't find a product

A customer - Robert - already knows what products he's looking for, but he doesn't know the exact location. He's going to ask Tiago for assistance and directions.

RAG Chatbot Conversation - Robert can't find a product

Chatbot: Hello, how are you doing today? Is there anything new that you'd like to look at or find in the shop?

User: Hello! Do you know where the NBA basketball is located?

Direction: product_info

RAG System Query:

```
{
  "product_id": "",
  "product_name": ["NBA basketball"],
  "product_brand": [],
  "product_category": ["Sports Equipment"],
  "product_sport_category": ["basketball"],
  "product_description": ["official NBA basketball"],
  "price_range": "",
  "product_location_needed": true
}
```


RAG System Result:

Product ID	Name	Price	Category	Location
red_box_basketball	Official NBA Basketball	$44.99	Sports Equipment	Unknown
red_box_basketball_hoop	Pro Basketball Hoop	$329.99	Sports Equipment	Unknown
basketball_shoes_001	Professional Basketball Shoes	$189.99	Sports Equipment	Unknown
basketball_shorts_001	Pro Basketball Shorts	$39.99	Sports Equipment	Unknown
basketball_jersey_001	Official NBA Jersey	$99.99	Sports Equipment	Unknown

Chatbot: We do carry a selection of basketball gear, but I don't have information on the specific location of our NBA basketballs. Let me check the inventory for you.

User: Mh nevermind. Do you have the Michael Jordan signature basketball instead?

Direction: product_info

RAG System Query:

```
{
  "product_id": "",
  "product_name": ["Michael Jordan signature basketball"],
  "product_brand": ["Michael Jordan"],
  "product_category": ["Sports Equipment"],
  "product_sport_category": ["basketball"],
  "product_description": ["signature basketball"],
  "price_range": "",
  "product_location_needed": false
}
```


RAG System Result:

Product ID	Name	Price	Category
red_box_basketball	Official NBA Basketball	$44.99	Sports Equipment
red_box_basketball_hoop	Pro Basketball Hoop	$329.99	Sports Equipment
basketball_shoes_001	Professional Basketball Shoes	$189.99	Sports Equipment
basketball_shorts_001	Pro Basketball Shorts	$39.99	Sports Equipment
basketball_jersey_001	Official NBA Jersey	$99.99	Sports Equipment

Chatbot: We don't have any information about a specific "Michael Jordan signature basketball" product, or its details.

User: Okay, understood. Thanks! Bye!

Direction: finished

Chatbot: You're welcome. Have a great day and feel free to visit us anytime for any sports-related queries or purchases. Bye!

Updating customer information...

RAG System Query:

```
{
  'products_of_interest': "['NBA basketball', 'Michael Jordan signature basketball']",
  'purchase_intent': 'low',
  'follow_up_needed': True,
  'conversation_summary': 'Conversation was closed before customer preferences could be obtained.'
}
```


This scenario highlights the capabilities of the assistant of not answering when the provided data is absent or not precise, still trying to assist the customer within its limited knowledge.

Scenario 4: customer looking for a staffer

A customer is looking for a staffer to assist him, and is going to ask Tiago for help.

RAG Chatbot Conversation - customer looking for a staffer

Chatbot: Hi, welcome to our sports shop. Is there something I can help you find today?

User: Hello! I'm looking for someone who could help me in basketball products. Anyone in the store available?

Direction: staff_info

RAG System Query:

```
{
  "staff_name": [],
  "staff_role": ["sales assistant"],
  "staff_category": ["shoes/ sports equipment"],
  "staff_id": "",
  "staff_location_needed": true
}
```

RAG System Result:

Staff ID	Name	Categories	Role	Area
staff_lorenzo	Lorenzo	Basketball, Baseball	Manager Assistant	basketball department

Chatbot: Lorenzo is our manager assistant for the basketball department and he's available to help. He's located in the basketball section. Would you like me to call for him?

User: No I'll go to him. Thanks! Bye!

Direction: finished

Chatbot: Bye! Have a nice day shopping.

This scenario highlights the capabilities of the assistant of recommending a staffer when asked and to give indication about the location.

Scenario 4: assisting a staffer

A staffer - Leonardo - is assisting a customer but cannot remember where a product is, so he's going to ask Tiago for the needed information.

RAG Chatbot Conversation - assisting a staffer

Chatbot: Good morning, how are you?

User: Hi! I'm helping a customer but I cannot find the tennis shoes. Do you know where they are?

Direction: product_info

RAG System Query:

```
{
  "product_id": "",
  "product_name": ["tennis shoes"],
  "product_brand": [],
  "product_category": ["shoes"],
  "product_sport_category": ["tennis"],
  "product_description": [],
  "price_range": "",
  "product_location_needed": true
}
```

RAG System Result:

Product ID	Name	Price	Category	Location
basketball_shoes_001	Professional Basketball Shoes	$189.99	Sports Equipment	tennis section
tennis_shoes_001	Professional Tennis Shoes	$179.99	Sports Equipment	tennis section
basketball_shoes_002	Standard Basketball Shoes	$69.99	Sports Equipment	tennis section
tennis_shoes_002	Standard Tennis Shoes	$79.99	Sports Equipment	tennis section
red_box_tennis_racket	Professional Tennis Racket	$149.99	Sports Equipment	tennis section

Chatbot: I can tell you that the tennis shoes are located in the ”tennis section” of our shop, as mentioned in the product descriptions.

User: I will manage it. Thanks! Bye.

Direction: finished

Chatbot: No problem, have a good day!

This scenario highlights the capabilities of the assistant to help a staffer in need and to give indication about a product location.

Chatbot limitations

Although the previous scenarios clearly show that the assistant is functional in most cases, further tests show inescapable limitations.

RAG Chatbot Conversation - Product Not Found

Chatbot: Hi there, how are you today? Is there anything specific you're looking for in our sports shop?

User: Hello! Yes! I was looking to buy a kayak.

Direction: product_info

RAG System Query:


```
{
  "product_id": "",
  "product_name": ["kayak"],
  "product_brand": [],
  "product_category": ["accessories"],
  "product_sport_category": ["water sport"],
  "product_description": [],
  "price_range": "",
  "product_location_needed": false
}
```


RAG System Result:

Product ID	Name	Price	Category

Chatbot: I'm afraid we don't have kayaks in stock right now. Our current selection includes bicycles, running shoes, and athletic wear. Would you like me to check if we can special order anything for you?

User: I'm sure you have them. Where can I find bicycles?

Direction: product_info

RAG System Query:


```
{
  "product_id": "",
  "product_name": [],
  "product_brand": [],
  "product_category": ["accessories"],
  "product_sport_category": [],
  "product_description": [],
  "price_range": "",
  "product_location_needed": false
}
```


RAG System Result:

Product ID	Name	Price	Category

Chatbot: Our bicycle section is on the left side of the shop, near the back. However, I could be more accurate, let me just check for you.

User: Okay, got it. Bye!

Direction: None

Chatbot: Wait, before you go, would you mind if I asked for your name again so I can update your customer information? It will only help us with future recommendations and special offers.

User: Don't worry! Bye!

Direction: None

Chatbot: Have a great day and I hope you find what you're looking for elsewhere.

User: Bye!

Direction: finished

Chatbot: Bye! Feel free to come back if you need anything else, okay?

Conversation ended without customer information update.

In particular, this discussion highlights the following.

- Difficulty to recognize direction in longer chats, in particular the "finished" signal.
- Hallucinations, in particular on the location of asked products.
- Leakage of system prompts instruction, making the conversation less natural.
- Customer knowledge extraction failure.

Some ways to overcome this could be:

- Changing the LLM model to a more performant alternative.
- System prompts improvement, with more detailed instructions and examples to follow.
- Domain-specific fine-tuned LLM's for each kind of task (query extraction, direction detection, turn-taking conversation)

4.2 Vision System Results

The vision module successfully demonstrates reliable person detection and AprilTag-based classification in the simulated retail environment. The system exhibits robust performance characteristics that enable effective human-robot interaction initiation.

4.2.1 Detection Performance

The HOG-based person detection system achieves consistent recognition of human figures in the Gazebo environment. AprilTag recognition proves highly effective for staff-customer classification, with detection success even at extended distances within the camera's field of view.

Key Performance Observations:

- **Distance Sensitivity**: AprilTag detection maintains high accuracy across varying distances, significantly outperforming face detection approaches in simulation
- **Angular Limitations**: Person detection accuracy decreases substantially when individuals are oriented perpendicular to the camera (90° rotation), requiring frontal or near-frontal positioning for reliable detection

- **Classification Reliability**: The deterministic AprilTag-based classification provides 100% accuracy when tags are successfully detected, eliminating uncertainty in staff-customer distinction

4.2.2 Integration with Reasoning Module

The vision system successfully interfaces with the reasoning module through a distributed approach to approach/departure detection:

- **Position Publishing**: Vision module provides 3D world coordinates via `VisionPersonDetection` messages
- **Distance-Based Triggering**: Reasoning module calculates proximity (less then 1.5m for "close", more than 3.0m for "far") to trigger conversation initiation/termination
- **Identity Persistence**: AprilTag-based unique person tracking enables consistent interaction across multiple detections

4.2.3 Simulation-Specific Adaptations

The vision system incorporates several design decisions tailored to the Gazebo simulation environment:

- **Model Enhancement**: Integration of higher-quality person models from external repositories to improve detection reliability
- **Custom AprilTag Models**: Creation of staff and customer models with chest-mounted AprilTag badges for reliable identification
- **Gazebo Integration**: Direct model state subscription for precise 3D positioning, leveraging simulation environment advantages

The vision system demonstrates successful adaptation to simulation constraints while maintaining design principles applicable to real-world deployment.

5 Conclusions

This project successfully demonstrates the integration of modern AI techniques to create an intelligent robotic shop assistant capable of autonomous navigation, natural language interaction, and adaptive behavioral responses in retail environments.

Key Achievements

The project's core innovation lies in integrating a dynamic Retrieval-Augmented Generation (RAG) system with conversational AI, enabling personalized customer profiles through natural interaction. The safety-aware A* path planning algorithm effectively balances human comfort with operational efficiency, while our finite state machine-based reasoning module successfully orchestrates complex behaviors across navigation, conversation, and vision subsystems.

Learning Outcomes and Development Experience

The development process highlighted the complexities of multi-modal robotic systems, particularly in managing inter-process communication and state consistency. Working with the Llama-3.1-8b-instant model emphasized the critical importance of prompt engineering and structured conversation flow management. Adapting to simulation constraints while maintaining real-world applicability taught us valuable lessons about pragmatic engineering decisions under technical limitations.

Strengths and Limitations

Strengths: Successful LLM-robotics integration, dynamic customer knowledge management, robust safety-aware navigation, and modular architecture facilitating future expansion.

Limitations: Conversation direction detection degrades in longer interactions, susceptibility to LLM hallucinations, vision system orientation constraints, and simulation-specific adaptations requiring modification for real deployment.

Future Improvements

Key enhancement opportunities include implementing sophisticated conversation state tracking, multimodal feedback systems for LLM validation, confidence scoring mechanisms, enhanced multi-perspective vision systems, real-time inventory integration, and advanced customer behavior analysis through extended interaction history.

Final Reflections

The Tiago Shop Assistant successfully bridges advanced AI capabilities with practical robotic applications, establishing a foundation for socially intelligent systems that enhance rather than replace human service. The project demonstrates that contemporary AI techniques can be effectively combined to create functionally capable, socially intelligent, and adaptable robotic assistants. Most importantly, it reinforces the importance of iterative design, robust testing, and pragmatic engineering decisions in developing complex robotic systems for real-world deployment.

6 Appendix

6.1 System prompts

Below are listed all the system prompts used to guide the LLMs generation for the HRI part. For the sake of brevity, the base system prompt used is the following:

System Prompt: BASE SYSTEM PROMPT

The following system prompt will be used as base for most of the chatbot's custom system prompts.

System Prompt Content:

> You're a sports shop assistant. Your role is to help customers or staffer in the shop.
> Be concise and clear with the answer, like if you were having a normal conversation.
> You do not do physical actions - like walking– you just provide information.
> The person you're talking with is a [PERSON_CATEGORY].
> It's really important that you do not make up information, If the user asks for something you don't know, just say you don't know and try to help them in other ways.
> This is what you know about the customer from previous conversations with them: `[CUSTOMER_INFORMATION]`.
> Try to not write more than 3 short sentences per answer.

System Prompt: `WALK_TO_PRODUCT/WALK_TO_STAFF`

Purpose: This prompt directs the LLM to inform the user that the robot will guide them to a specific product/staff member's location. Three variants of the prompt are shown:

1. To inform the user that the requested product/staffer is not available.
2. To inform the user that he will be walked to the desired location.
3. To inform the user the they arrived to the desired location.

System Prompt (1) Content:

> [BASE SYSTEM PROMPT]
> Say the customer you couldn't find the product/staff member so you cannot walk him to it

System Prompt (2) Content:

> [BASE SYSTEM PROMPT]
> Say the customer you're going to walk him to the asked location.

System Prompt (3) Content:

> [BASE SYSTEM PROMPT]
> Say the customer you have arrived at the location and ask if he needs further assistance.

System Prompt: `PRODUCT_INFO` // `STAFF_INFO`

Purpose: This prompt instructs the LLM to provide detailed information about a requested product.
Two versions of the prompt are shown:

1. When the required information is available.
2. When the required information is not available.

System Prompt (1) Content:

> [BASE SYSTEM PROMPT]
> Try to help the user with the following information:
> `[PRODUCTS_INFORMATION]` / `[STAFF_INFORMATION]`
> If the data is not relevant, do not include it in the answer and try to help the customer in other ways.

System Prompt (2) Content:

> [BASE SYSTEM PROMPT]
> You don't have the specific products/staff in the shop the user is asking for.
> Tell him that and try to help him in other ways.

System Prompt: `CONVERSATION_FINISHED`

Purpose: This prompt is used when the conversation naturally concludes. It directs the LLM to provide a polite closing message, ensuring a smooth and user-friendly end to the interaction.

System Prompt Content:

> [BASE SYSTEM PROMPT]
> The conversation has reached its conclusion. Say a polite goodbye and offer further assistance if needed.

System Prompt: `DEFAULT`

Purpose: This prompt serves as a fallback for when the user's intent doesn't fit into the predefined categories. It guides the LLM to acknowledge it doesn't have the specific information or capability requested and to offer general assistance, maintaining a helpful persona.

System Prompt Content:

> [BASE SYSTEM PROMPT]

6.2 RAG prompts

This section details the prompt structures used by the Retrieval-Augmented Generation (RAG) system to accurately interpret user intent and generate structured queries for the database database. These prompts are passed to the Large Language Model (LLM) internally to guide its information extraction process. The section also outlines the expected JSON response format from these queries.

The LLM is given specific instructions to extract the necessary information for each query type, ensuring that the RAG system can effectively retrieve and present relevant data to the user.

Prompt Structure: Conversation Direction Detection

Purpose: This prompt guides the LLM to identify the user's core intent from the conversation history and current input, categorizing it into one of the predefined interaction types.

Prompt Content:

```
You're a sports shop assistant. Your task is to analyze the conversation
and determine what the user is asking for.
 Answer with JSON with the following structure:
 { 'conversation_direction': 'product_info' | 'staff_info' |
 'walk_to_product' | 'walk_to_staff' | 'finished' | '' }.
 Where:
 - 'product_info' means the user is asking about products information or
 location in the shop.
 - 'staff_info' means the user is asking about staff information or
 location in the shop.
 - 'walk_to_product' means the user wants to be walked to a product ( only
 if he asks it explicitly ).
 - 'walk_to_staff' means the user wants to be walked to a staff member
 ( only if he asks it explicitly ).
 - 'finished' means the conversation is over and no further action is
 needed ( only if he asks it explicitly ).
 - '' means the conversation is not about any of the above.
 Just answer with ONE ( 1 ) JSON, no extra text, with the most likely
 direction.
```

Prompt Structure: Products Query

Purpose: This prompt directs the LLM to extract specific attributes related to a product (e.g., category, name, brand, sport, description, price range) from the user's query.
Prompt Content:

```
You're a sports shop assistant. Your task is to analyze the conversation
and determine what kind of product the user is looking for.
 Answer with JSON with the following structure:
{
'product_category': str,  # e.g. 'shoes', 'clothes', 'accessories'
'product_name': str,  # e.g. 'running shoes', 'football jersey'
'product_brand': str,  # e.g. 'Nike', 'Adidas'
'product_sport_category': str,  # e.g. 'running', 'football'
'product_description': str,  # e.g. 'lightweight running shoes for long
distances'
'price_range': str,  # e.g. 'low', 'medium', 'high'
'product_location_needed': bool  # whether the user wants to be walked to
the product location
}.
In this case, all the fields are optional ( in case fill them with 'None' ),
and you can return an empty JSON if nothing relevant.
Answer only with ONE ( 1 ) JSON, no extra text.
```

Prompt Structure: Staff Query

Purpose: This prompt instructs the LLM to identify specific staff-related information (e.g., name, role, category) from the user's query.
Prompt Content:

```
You're a sports shop assistant. Your task is to analyze the conversation
and determine what kind of staff the user may need or is looking for.
 Answer with JSON with the following structure:
{
'staff_name': str,  # e.g. 'John Doe'
'staff_role': str,  # e.g. 'sales assistant', 'manager'
'staff_category': str,  # e.g. 'shoes', 'clothes', 'accessories'
'staff_location_needed': bool  # whether the user wants to be walked to the
staff member's location
}
In this case, all the fields are optional ( in case fill them with 'None' ),
and you can return an empty JSON if nothing relevant.
Answer only with ONE ( 1 ) JSON, no extra text.
```

Prompt Structure: Knowledge Extraction (Customer Information Update)

Purpose: This prompt is used to extract and update structured information about the customer from the entire conversation history. It allows the system to maintain a dynamic profile of the user's preferences, interests, and interaction patterns, enhancing personalization in subsequent interactions.
Prompt Content (from `extract_knowledge` method):

```
You're analyzing a conversation between a sports-shop assistant and a user.
Extract structured information from the *user* messages only.
Create a JSON with the following optional fields:
customer_info: { name, age_category, gender, preferences, size_info,
budget },
products_of_interest, purchase_intent (high|medium|low), follow_up_needed
(boolean), conversation_summary
Extract *only* information that was explicitly shared. All scalar fields
must be strings; lists where appropriate. follow_up_needed is boolean.
Previous structured information about the user (update with new insights):
[CUSTOMER_INFORMATION_IF_PRESENT]
Answer only with ONE ( 1 ) JSON - no extra text.
```

6.3 Data Structures and contextual data

6.3.1 Customer Information ('[CUSTOMER_INFORMATION]')

6.4 Person category ('[PERSON_CATEGORY]')

The '[CUSTOMER_INFORMATION]' placeholder in system prompts and RAG prompts refers to a JSON object containing a dynamic profile of the user. This information is extracted from the ongoing conversation and updated by the 'Knowledge Extraction (Customer Information Update)' prompt. The structure reflects the 'Customer' entity in the database and can include the following optional fields:

- `customer_info`: A nested JSON object with details such as:
  - `name`: (string) The customer's name.
  - `age_category`: (string) A broad age group (e.g., "young adult", "senior").
  - `gender`: (string) The customer's gender.
  - `preferences`: (string or list of strings) Customer's expressed preferences (e.g., "likes running shoes", "prefers sustainable brands").
  - `size_info`: (string) Any relevant sizing information (e.g., "shoe size 10", "medium clothing").
  - `budget`: (string) Indication of the customer's budget (e.g., "low", "medium", "high").
- `products_of_interest`: (list of strings) Products the customer has shown interest in.
- `purchase_intent`: (string) An assessment of the customer's likelihood to purchase ("high", "medium", "low").
- `follow_up_needed`: (boolean) Indicates if further assistance or follow-up is required.
- `conversation_summary`: (string) A brief summary of the conversation so far, capturing key topics or outcomes.

All scalar fields are represented as strings, and lists are used where appropriate. This structured information allows the system to personalize interactions and provide more relevant assistance.

6.4.1 Product Information (‘[PRODUCTS_INFORMATION]‘)

When the LLM provides information about products (e.g., for the ‘PRODUCT_INFO‘ system prompt), the ‘[PRODUCTS_INFORMATION]‘ placeholder is populated with details of one or more products retrieved from the database. The structure of a single product object includes:

- `product_id`: (string) Unique identifier for the product.
- `name`: (string) The name of the product (e.g., "Nike Air Zoom Pegasus").
- `price`: (float) The price of the product.
- `brand`: (string) The brand of the product.
- `description`: (string) A detailed description of the product.
- `category`: (string) The general category of the product (e.g., "shoes", "apparel").
- `sport_category`: (string) The sport or activity the product is associated with (e.g., "running", "football").

6.4.2 Staff Information (‘[STAFF_INFORMATION]‘)

Similar to product information, when inquiries about staff are made (e.g., for the ‘STAFF_INFO‘ system prompt), the ‘[STAFF_INFORMATION]‘ placeholder receives details about staff members. The structure of a single staff object comprises:

- `staff_id`: (string) Unique identifier for the staff member.
- `name`: (string) The name of the staff member.
- `role`: (string) The role or position of the staff member (e.g., "sales assistant", "manager").
- `categories`: (list of strings) The areas of expertise or product categories the staff member specializes in (e.g., ["shoes", "customer service"]).

6.5 Person category ('[PERSON_CATEGORY]')

The '[PERSON_CATEGORY]' placeholder is used in the system prompt to make the Chatbot aware of the kind of person he's talking to. The possible options are:

- *a staff member of the shop*: if the approaching person is a staff member.
- *a customer of the shop*: if the approaching person is an actual customer.
- *a person in the shop*: if the robot is unaware of the nature of the approaching person.

References

[1] ROS 2 Documentation. rclpy API and Python Package Development. Open Robotics, 2025. `https://docs.ros.org/en/humble/`.

[2] cv_bridge Package Documentation. ROS-OpenCV Bridge for Image Processing. Open Robotics, 2025.

[3] OpenCV Documentation. Cascade Classifiers and Face Detection. OpenCV Team, 2025. `https://docs.opencv.org/`.

[4] TIAGo Robot Documentation and Tutorials. PAL Robotics, 2025. `https://wiki.ros.org/Robots/TIAGo/Tutorials`.

[5] sensor_msgs Package Documentation. ROS 2 Standard Message Definitions. Open Robotics, 2025.

[6] Human-Robot Interaction 2025. Lecture 1 slides: NAO, TIAGo and Adriano Robotic Platforms - Introduction and Overview. Course materials.

[7] Human-Robot Interaction 2025. Lecture 12 slides: NAO, TIAGo and Adriano Robotic Platforms - Advanced Applications and Case Studies. Course materials.

[8] Human-Robot Interaction 2025. Lecture slides. Social signal perception, reasoning, and generation architectures.

[9] Robot Benchmarking and Competitions 2025. Lecture slides. Functional architecture design for HRI systems.

[10] Hengli Liu, Jun Luo, Peng Wu, Shaorong Xie, and Hengyu Li. People detection and tracking using rgb-d cameras for mobile robots. *International Journal of Advanced Robotic Systems*, 13(5):1–11, 2016.

[11] Lucas Massa, Adriano Barbosa, Krerley Oliveira, and Thales Vieira. Lrcn-retailnet: A recurrent neural network architecture for accurate people counting. *arXiv preprint arXiv:2004.09672*, 2020.

[12] Aibek Musaev, Jiangping Wang, Liang Zhu, Cheng Li, Yi Chen, Jialin Liu, Wanqi Zhang, Juan Mei, and De Wang. Towards in-store multi-person tracking using head detection and track heatmaps. *arXiv preprint arXiv:2005.08009*, 2020.

[13] PAL Robotics. Tiago simulation package, 2025. GitHub repository for TIAGo Gazebo simulation.

[14] PAL Robotics. Tiago tutorials - getting started with tiago robot, 2025. Official tutorials for TIAGo robot setup, simulation, and programming.