Software Engineering Application Requirements Document (ARD) - AOS Middleware

Student Name	ID
Majd Atwan	211709407
Michel Mousa	314811589
Natalie Ahmad	206692295
Moamen Mawasi	206604555

Abstract

This document outlines the requirements for the development of a new middleware for the Autonomous Operating System (AOS). The middleware will feature both a graphical and RESTful user interface, facilitating the management and control of robot capabilities. The operations include receiving configuration files, automatic code generation, listening to public communication in the robot, and reporting results.

The system will operate in two modes:

- Configuration mode involves receiving configuration files and dynamically generating code. This generated code, when executed, enables the activation of robot capabilities, creation of notifications, and more.
- Execution mode, where the middleware will receive commands to execute capabilities, perform them, and update the planning engine.

Contents

Chapter 1

Introduction

1.1 The Problem Domain

The current landscape of Autonomous Operating System (AOS) usage presents multifaceted challenges that demand a comprehensive solution. Users and developers encounter a myriad of technical and operational difficulties in effectively managing autonomous robots. These challenges include:

Fragmented Tooling: Users and developers are compelled to rely on a disparate array of tools for various tasks such as robot activation, skill documentation, and debugging. This fragmented approach not only complicates workflows but also introduces the risk of inconsistencies and errors in documentation and monitoring.

Operational Cohesion: The separation of debugging and monitoring from the broader AOS system leads to inefficiencies and a lack of cohesion in the troubleshooting process. This disjointed approach hampers the ability to effectively debug and monitor the behavior of autonomous robots.

1.2 Context

In response to these pressing challenges, our project aims to revolutionize the AOS ecosystem by introducing a middleware solution that streamlines operations and enhances user experience. By seamlessly integrating with existing planning engines and robot infrastructures, particularly ROS2, our middleware will bridge the gap between disparate components, fostering operational cohesion and simplifying development workflows.

1.3 Vision

Our vision is to significantly enhance the development, management, and usability of autonomous robots within the AOS framework. We aim to achieve this by:

Streamlined Workflow: Developing a robust middleware solution that acts as an intermediary between the AOS planning engine and robot infrastructure, thereby simplifying development processes and fostering operational efficiency.

Enhanced Usability: Providing intuitive user interfaces for monitoring system status, ensuring that users can easily manage and monitor autonomous robots without significant technical expertise.

1.4 Stakeholders

Stakeholders in our project include developers, system administrators, and end-users involved in autonomous robot development. Additionally, our collaborating lab, Prof. Barfman's lab, underscores the importance of delivering a modular, easily maintainable, and flexible product to support future development endeavors.

1.5 Software Context

High-Level Description of the Software System:

Our planned software system is a middleware solution tailored for the Autonomous Operating System (AOS) environment, with a primary focus on enhancing usability, streamlining development processes, and facilitating effective communication between components. Emphasizing modularity, ease of maintenance, and flexibility for future enhancements, the system will feature graphical and RESTful user interfaces to cater to diverse user needs and preferences.

Major Inputs:

1. Metadata Configuration: User-provided metadata describing robot capabilities, environment, and goals.
2. User Commands: Commands from users through the graphical and RESTful interfaces, specifying actions such as activating robot capabilities or querying system status.

Outputs

1. System Notifications: Notifications and alerts informing users about the status of executed commands, system events, and potential issues.
2. Real-time Reports: Real-time reports and visualizations displaying the current state of robot capabilities, environment variables, and system alerts.

Common Usage Scenarios (Use-Cases):

1. Configuration Mode
    - User Action: Provide metadata about robot capabilities.
    - System Response: Process metadata, generate activation code, and listen to internal robot communication.
2. Activation Command:
    - User Action: Issue a command to activate specific robot capabilities.
    - System Response: Execute the specified capabilities, update the planning engine, and generate real-time reports.
3. User Interface Interaction:
    - User Action: Interact with the graphical interface to monitor robot status.
    - System Response: Display real-time reports, visualize robot activities, and provide alerts if necessary.
4. RESTful Command:
    - User Action: Send a command through the RESTful interface to query robot parameters.
    - System Response: Process the command, retrieve relevant data, and provide a response through the interface.

Chapter 2

Usage Scenarios

2.1 User Profiles — The Actors

Autonomous Robot Developers:

- *Characteristics:*
  - Highly skilled professionals in robotics.
  - Proficient in configuring and programming autonomous robots.
  - Well-versed in metadata specifications and system integration.
- *Role:*
  - Engage in configuring the system by providing metadata about robot capabilities.
  - Activate and manage robot functionalities through the middleware.
  - Monitor real-time reports for performance optimization.

System Administrators:

- *Characteristics:*
  - Expertise in system management and maintenance.
  - Familiarity with middleware architecture and configuration.
  - Skillful in addressing system-related issues.
- *Role:*
  - Manage and maintain the middleware system.
  - Ensure system reliability and performance.
  - Address configuration and maintenance requirements.

External Systems:

- *Characteristics:*
  - Non-human entities, such as monitoring tools.
  - Capable of sending and receiving data programmatically.
  - Automated systems seeking data and control functionalities.

- *Role:*
    - Interact with the middleware through the RESTful interface.
    - Send commands to query robot parameters or initiate specific actions.
    - Receive data and responses from the middleware for external system integration

2.2 Use-Cases

- **Use Case 1**: Configure Robot Capabilities
    - Description:

      A user, typically an autonomous robot developer or a system administrator, engages in configuring the system by providing metadata about the robot's capabilities. This metadata includes specifying the types of tasks the robot can perform, its sensor capabilities, and other relevant details necessary for its operation within the autonomous operating system (AOS) framework.

    - Actors:
        - Autonomous Robot Developer
        - System Administrator
    - Pre-Conditions:

      None.

    - Post-Conditions:

      After successful configuration, the robot's capabilities are stored in the system, making them accessible for further utilization.

    - Main Success Scenario:
        1. The user accesses the configuration interface provided by the system.
        2. Inputs metadata describing various aspects of the robot's capabilities, such as its motion control abilities, sensor types, and communication protocols.
        3. The system processes and validates the configuration data provided by the user.
        4. Upon successful validation, the system stores the configuration data in the appropriate database or configuration files.
        5. The system confirms to the user the successful configuration, indicating that the robot's capabilities are now configured within the system and ready for use.

- **Use Case 2:** Activate Robot Capability
  - Description:

    An operator or developer initiates the activation of a specific robot capability, enabling the robot to execute a designated task. This activation process can be performed either through the graphical interface or by sending RESTful commands to the middleware system.

  - Actors:
    - Autonomous Robot Developer
  - Pre-Conditions:

    Before activating a robot capability, it is essential that the relevant capabilities have been previously configured within the system.

  - Post-Conditions:

    Upon successful activation, the specified robot capability becomes active and ready for execution.

  - Main Success Scenario:
    1. The user selects the desired robot capability for activation, either through the graphical user interface or via RESTful commands.
    2. The user initiates the activation process, signaling to the system their intention to enable the selected capability.
    3. The system interprets the activation request and translates it into executable tasks for the robot.
    4. Upon successful translation, the system confirms to the user that the activation process has been completed.
    5. The activated capability is now ready for execution, and the robot can perform the designated task as intended.

- **Use Case 3**: Monitor Real-time Robot Behavior
  - Description:

    Users, typically developers or operators, access the real-time monitoring interface provided by the system to observe the behavior and status of the autonomous robot. This includes monitoring sensor data, task execution progress, and any system alerts or notifications.

  - Actors:
    - Autonomous Robot Developer

- Pre-Conditions

  Before monitoring the robot's behavior in real-time, its capabilities must be activated within the system.

- Post-Conditions:

  Upon accessing the monitoring interface, users gain insights into the robot's current behavior and operational status.

- Main Success Scenario:
  1. The user accesses the real-time monitoring interface provided by the system.
  2. The interface displays relevant information such as sensor readings, ongoing tasks, and any alerts or notifications.
  3. The system continuously updates the interface with real-time data, ensuring that users have access to the latest information about the robot's behavior.
  4. The user gains insights into the robot's behavior and operational status, allowing for informed decision-making and troubleshooting as necessary.

- **Use Case 4**: Send RESTful Command
  - Description:

    External systems or users interact with the middleware system by sending RESTful commands. These commands can include queries about robot parameters, initiation of specific actions, or retrieval of system information.

  - Actors:
    - External Systems
    - API Consumers
  - Pre-Conditions:

    The middleware system must be operational and accessible to external systems.

  - Post-Conditions:

    Upon processing the RESTful command, the system responds accordingly by either retrieving the requested data or initiating the specified action.

  - Main Success Scenario:
    1. An external system or API consumer sends a RESTful command to the middleware system.
    2. The middleware system receives and processes the command.

3. Depending on the nature of the command, the system either retrieves relevant data from its database or initiates the specified action, such as activating a robot capability or retrieving sensor readings.
4. The system sends a response back to the external system or API consumer, confirming the successful processing of the command and providing any requested data or acknowledgments.

- **Use Case 5:** Initialize Project Request
  - Description:

    A user, typically an autonomous robot developer or a system administrator, initiates a project request within the system. This involves providing detailed information about the goals, scope, and requirements of the autonomous robot project. The project request functionality supports the system in managing and organizing project-related data efficiently.

  - Actors:
    - Autonomous Robot Developer
    - System Administrator
  - Pre-Conditions:

    None.

  - Post-Conditions:

    Upon successful initialization, the project request information is stored in the system for further processing and reference.

  - Main Success Scenario:
    1. The user accesses the project initialization interface provided by the system.
    2. The user provides comprehensive details about the project goals, scope, and specific requirements, ensuring clarity and alignment with project objectives.
    3. The system stores the project request information in its database, associating it with relevant project identifiers and metadata.
    4. The system confirms to the user the successful initialization of the project request, indicating that the provided information has been recorded and is available for further processing.

- **Use Case 6:** Graphical Interface Access
    - Description:

      Users, including autonomous robot developers and system administrators, access the graphical interface provided by the system to interact with its functionalities. The graphical interface serves as the primary means for users to configure, activate, monitor, and manage autonomous robot capabilities, providing an intuitive and user-friendly experience.

    - Actors:
        - Autonomous Robot Developer
    - Pre-Conditions:

      The middleware system must be operational and accessible via the graphical interface.

    - Post-Conditions:

      Upon successful access, users gain the ability to interact with the system's functionalities through the graphical interface.

    - Main Success Scenario:
        1. The user opens a web browser or application and navigates to the graphical interface provided by the system.
        2. The user logs in with appropriate credentials, ensuring secure access to system functionalities.
        3. The graphical interface displays relevant options and features for system interaction, such as configuration settings, activation controls, monitoring tools, and management functions.
        4. The user successfully interacts with the middleware system through the graphical interface, performing tasks such as configuring robot capabilities, activating specific functionalities, monitoring real-time data, and managing system settings.

- **Use Case 7:** Update Robot Configuration
    - Description:

      A user, either an autonomous robot developer or a system administrator, modifies the existing configuration of the autonomous robot within the system. This involves updating metadata or parameters related to the robot's capabilities, such as adding new skills or adjusting sensor configurations, to reflect changes or improvements in its functionality.

- Actors:
  - Autonomous Robot Developer
  - System Administrator
- Pre-Conditions:

  The robot's configuration must have been previously established within the system.

- Post-Conditions:

  After successful modification, the updated configuration is saved in the system, ensuring that the robot's capabilities are up-to-date and accurately represented.

- Main Success Scenario:
  1. The user accesses the configuration interface provided by the system, which allows for the modification of robot configurations.
  2. The user identifies the specific aspects of the robot's capabilities that require updating, such as adding new skills or adjusting sensor parameters.
  3. The user modifies the relevant metadata or parameters within the configuration interface, ensuring that the changes accurately reflect the desired updates.
  4. The system processes and validates the updated configuration data provided by the user, ensuring its consistency and compatibility with existing system components.
  5. Upon successful validation, the system saves the updated configuration in its database, confirming to the user the successful completion of the configuration update process.

- **Use Case 8:** Deactivate Robot Capability
  - Description:

    An operator or developer deactivates a specific robot capability within the system, temporarily disabling its functionality. This action may be necessary for maintenance purposes, troubleshooting, or to prevent the execution of certain tasks under specific conditions.

  - Actors:
    - Operator
    - Autonomous Robot Developer
  - Pre-Conditions:

    The robot capability to be deactivated must have been previously activated within the system.

- Post-Conditions:

  Upon successful deactivation, the specified robot capability is temporarily disabled within the system, preventing its execution until reactivated.

- Main Success Scenario:
  1. The user selects the desired robot capability for deactivation within the system's graphical interface or through RESTful commands.
  2. The user initiates the deactivation process, signaling to the system their intention to temporarily disable the selected capability.
  3. The system processes the deactivation request and updates its internal state to reflect the disabled status of the specified capability.
  4. Upon successful processing, the system confirms to the user the successful deactivation of the specified robot capability, indicating that it is now inactive and unavailable for execution.

- **Use Case 9:** Query Robot Parameters through RESTful Command
  - Description:

    External systems or users retrieve specific parameters or information about the autonomous robot by sending a RESTful query command to the middleware system. This functionality enables seamless communication between external entities and the middleware, facilitating data exchange and retrieval.

  - Actors:
    - External Systems
    - API Consumers
  - Pre-Conditions:

    The middleware system must be operational and accessible to external systems.

  - Post-Conditions:

    Upon processing the RESTful query command, the system responds with the requested robot parameters or information, providing the external entity with the required data.

  - Main Success Scenario:
    1. An external system or API consumer sends a RESTful query command to the middleware system, specifying the parameters or information they wish to retrieve.
    2. The middleware system receives and processes the query command, identifying the requested data and accessing the relevant information within its database.

3. The system retrieves the requested robot parameters or information as per the query, ensuring its accuracy and relevance to the specified criteria.
4. Upon successful retrieval, the system sends a response back to the external system or API consumer, containing the requested data or information, and confirming the completion of the query process.

- **Use Case 10:** Add New Robot Skill
  - Description:

    A developer, either an autonomous robot developer or a system administrator, adds a new skill to the capabilities of the autonomous robot within the system. This involves defining the behavior and functionality of the new skill through the graphical interface, enhancing the robot's capabilities and versatility.

  - Actors:
    - Autonomous Robot Developer
    - System Administrator
  - Pre-Conditions:

    The robot capabilities must have been previously configured within the system.

  - Post-Conditions:

    Upon successful addition, the new skill is integrated into the robot's capabilities within the system, expanding its range of functionalities and potential applications.

  - Main Success Scenario:
    1. The user accesses the skill creation interface provided by the system, which allows for the definition of new robot skills.
    2. The user defines the behavior and functionality of the new skill, specifying its intended purpose, input-output behavior, and any associated parameters or requirements.
    3. The system processes and validates the definition of the new skill provided by the user, ensuring its compatibility with existing system components and configurations.
    4. Upon successful validation, the system adds the new skill to the robot's capabilities within its database, confirming to the user the successful creation and integration of the new skill.

- **Use Case 11:** Retrieve Real-time Alerts
    - Description:

      Users, such as developers or operators, receive real-time alerts through the graphical interface regarding critical events or issues related to the autonomous robot. This functionality enables users to stay informed about the robot's status and promptly address any emerging issues or concerns.

    - Actors:
        - Autonomous Robot Developer
        - Operator
    - Pre-Conditions:

      The robot capabilities must be activated within the system.

    - Post-Conditions:

      Upon receiving real-time alerts, users are informed about critical events or issues related to the autonomous robot, allowing them to take appropriate actions as needed.

    - Main Success Scenario:
        1. The user monitors the real-time alerts interface provided by the system, which displays notifications about critical events or issues related to the autonomous robot.
        2. The system continuously updates the interface with real-time alerts, ensuring that users receive immediate notifications about any emerging issues or concerns.
        3. Upon receiving an alert, the user acknowledges the notification and reviews the details provided by the system, gaining insights into the nature and severity of the issue.
        4. Based on the information received, the user takes appropriate actions to address the issue, such as initiating troubleshooting procedures, adjusting system configurations, or requesting assistance from other stakeholders.

- **Use Case 12:** Add New ROS-Compatible Robot Infrastructure
    - Description:

      A system administrator adds a new robot infrastructure compatible with the Robot Operating System (ROS) to the middleware system. This functionality enables the integration of diverse robot infrastructures into the system, enhancing its versatility and compatibility with different hardware platforms.

- Actors:
  - System Administrator
- Pre-Conditions:

  The middleware system must be operational and accessible to the system administrator.

- Post-Conditions:

  Upon successful addition, the new ROS-compatible robot infrastructure is seamlessly integrated into the middleware system, allowing for its utilization in autonomous robot development and operation.

- Main Success Scenario:
  1. The system administrator accesses the infrastructure management interface provided by the system, which allows for the addition of new robot infrastructures.
  2. The administrator provides details of the new ROS-compatible robot infrastructure, including its specifications, capabilities, and any required configuration parameters.
  3. The middleware system processes and validates the information provided by the administrator, ensuring compatibility with existing system components and configurations.
  4. Upon successful validation, the system integrates the new ROS-compatible robot infrastructure into its database, confirming to the administrator the successful addition and availability of the new infrastructure for use.

- **Use Case 13:** Update System Parameters
  - Description:

    A system administrator updates parameters related to the middleware's operation, ensuring optimal performance and adaptability. This functionality allows administrators to fine-tune system configurations based on changing requirements or environmental conditions.

  - Actors:
    - System Administrator
  - Pre-Conditions:

    The middleware system must be operational and accessible to the system administrator.

- Post-Conditions:

  Upon successful update, the system parameters are adjusted according to the new configurations, optimizing system performance and adaptability.

- Main Success Scenario:
  1. The system administrator accesses the system parameters interface provided by the system, which allows for the modification of various parameters.
  2. The administrator identifies the parameters requiring modification and adjusts their values based on the desired changes.
  3. The middleware system adapts to the updated parameters, incorporating the new configurations into its operation and adjusting its behavior accordingly.
  4. Upon successful update, the system confirms to the administrator the completion of the parameter modification process and the implementation of the new configurations.

- **Use Case 14:** Retrieve Historical Robot Data
  - Description:

    Users, such as operators or developers, retrieve historical data about the autonomous robot's performance and behavior through the graphical interface. This functionality enables users to analyze past data for insights, trend analysis, and decision-making purposes.

  - Actors:
    - Operator
    - Autonomous Robot Developer
  - Pre-Conditions:

    The robot capabilities must be activated within the system.

  - Post-Conditions:

    Upon retrieving historical robot data, users gain access to valuable insights into the robot's past performance and behavior, facilitating informed decision-making and analysis.

  - Main Success Scenario:

1. The user accesses the historical data retrieval interface provided by the system, which allows for specifying the time range and parameters for data retrieval.
2. The user defines the criteria for data retrieval, including the time range and specific parameters of interest.
3. The system retrieves and displays the historical performance and behavior data of the autonomous robot based on the specified criteria, presenting it to the user for analysis.
4. The user analyzes the retrieved historical data, gaining insights into the robot's past behavior, performance trends, and any recurring patterns or anomalies.
5. Based on the analysis, the user makes informed decisions and takes appropriate actions to optimize the robot's operation or address any identified issues or concerns.

These use cases provide a comprehensive overview of the functionalities and interactions within the system, covering various scenarios and user roles. Let me know if you need further elaboration or if there are additional use cases to address!

2.3 Special Usage Considerations

Special Requirements:

1. Scalability:

   Description: The system should be able to scale effectively to accommodate a growing number of autonomous robots and diverse capabilities. It should handle increased traffic and data exchange without compromising performance.

2. Compatibility with ROS Versions:

   Description: Ensure compatibility with multiple versions of the Robot Operating System (ROS) to facilitate integration with different robot infrastructures.

3. Real-time Data Processing:

   Description: The system must be capable of processing real-time data efficiently, ensuring timely responses and decision-making capabilities.

4. Interoperability with External Systems:

   Description: Ensure seamless interoperability with external systems, allowing the middleware to integrate data from various sources.

5. User Interface Responsiveness:

Description:The graphical and RESTful user interfaces should be responsive, providing users with a smooth and efficient interaction experience.

Chapter 3

Functional Requirements

1. System Initialization and Configuration
    - Description: The system must initialize and configure settings to ensure proper functionality
    - Priority: Must-Have
    - Risk: Low
2. Metadata Reception and Processing
    - Description: The system must receive and process metadata for configuring and connecting to robots.
    - Priority: Must-Have
    - Risk: Medium
3. Communication Monitoring
    - Description: The system must continuously monitor communication within the robot environment
    - Priority: Must-Have
    - Risk: Medium
4. Running Mode
    - Description: The system must activate and execute robot capabilities based on configured settings.
    - Priority: Must-Have
    - Risk:High
5. Robot Infrastructure Compatibility
    - Description:The system must ensure compatibility with various robot infrastructures, with ROS2 as the default
    - Priority: Must-Have
    - Risk: Medium.
6. Dynamic Parameter Listening
    - Description: The system must dynamically listen to changes in ROS topics and parameters.
    - Priority: Must-Have
    - Risk: Medium

7. Event-Based Notification System
    - Description: The system should implement a notification system to alert users about significant events.
    - Priority: Should-Have

- Risk: Low

8. Remote User Interface
    - Description: The system must provide a web-based interface for remote monitoring and control.
    - Priority: Must-Have
    - Risk:Medium
9. Integration with AOS Planning Engine
    - Description:The system must seamlessly integrate with the AOS planning engine for coordinated decision-making.
    - Priority: Must-Have
    - Risk: Medium
10. System Interface with External Devices
    - Description:The system could allow interfaces with external devices for extended functionality.
    - Priority: NTH
    - Risk:Low

Chapter 4

Non-functional Requirements

4.1 Implementation Constraints:

- **Performance:**

  Description: The middleware system should ensure fast transaction completion times to enhance overall performance**.**

- **Reliability & Stability:**

  Description: The middleware system needs to support data recovery and error-correction mechanisms to ensure reliability and stability.

- **Safety & Security:**

  Description: The middleware system should prioritize confidentiality, implementing encryption for data security and controlling user access based on authority levels.

- **Usability:**
  Description: The middleware system should prioritize a user-friendly interface, accommodating users with varying levels of technical expertise.

- **Availability:**

  Description: Factors affecting system availability, such as maintenance and updates, should be carefully managed to meet the specified availability level.

4.2 Platform Constraints:

The development must be done using Python & C# . The system should be designed to integrate seamlessly with ROS2 (Robot Operating System 2).

The middleware system must be compatible with existing middleware technologies, ensuring smooth communication with ROS services and actions.

The system must be deployable on Linux (Ubuntu 20)-based operating systems to align with ROS2 requirements.

The middleware system should utilize a database system compatible with Python & SQL, MongoDB, for efficient data storage and retrieval.

4.2.1 SE Project Constraints:

The inputs for the system during the demo will be provided based on the information and metadata received from users or developers. The specifics of the inputs and their sources will be coordinated and managed by our project monitor, who will facilitate the necessary data to showcase the system's capabilities effectively.

4.3 Special Restrictions & Limitations:

- **Assumptions:**
  Description:The design assumes access to standard development tools and hardware. Simulations may be required for specific hardware conditions. The middleware system assumes users have basic computer literacy.

Chapter 5

Risk Assessment & Plan for the Proof of Concept

Risk Assessment and Feasibility Study Plan

- **Objective:**
  To define the prototype's scope, identify potential risks, and justify the selection of specific features for the Proof of Concept (PoC).

- **Prototype Scope:**
  The prototype will serve as a demonstration of fundamental functionalities of the middleware system. It will focus on core features essential for autonomous robot development within the AOS framework.

- **Justification for Starting Features:**
  1. Basic System Development:
     - ★ Rationale: Starting with a basic version of the system allows us to establish a foundation for further development and testing.
  2. MongoDB Integration:
     - ★ Rationale: MongoDB integration provides efficient data storage and management capabilities, crucial for handling metadata and system configurations.
  3. RESTful User Interface Communication:
     - ★ Rationale: Enabling RESTful communication enhances system accessibility by providing a standardized interface for interaction.
  4. Robot Capability Demonstration:
     - ★ Rationale: Demonstrating the robot's capability to execute specific actions based on commands showcases the system's core functionality and integration with robotic infrastructure.
  5. Graphical Representation Module:
     - ★ Rationale: Developing a module for mapping the robot's state to graphical representation lays the groundwork for visualizing system data and status, aiding in monitoring and troubleshooting.

- **Risk Assessment:**
    1. Integration of New Technologies:
        - ★ Mitigation: Adopt a phased integration approach, conduct thorough compatibility and performance testing at each stage to minimize risks.

    2. GUI Framework Selection:
        - ★ Mitigation: Perform a comprehensive analysis of potential GUI frameworks, considering factors such as scalability, ease of use, and compatibility with project requirements.
    3. Listening to Internal Robot Communications:
        - ★ Mitigation: Develop clear communication protocols and compatibility layers to ensure seamless integration with internal robot communications.
    4. RESTful User Interface:
        - ★ Mitigation: Prioritize effective API design, implement robust security measures, and focus on efficient user interaction management to mitigate risks associated with RESTful interface implementation.

Plan for Proof of Concept:

1. Develop a basic version of our system to showcase fundamental functionalities.

2. Implement MongoDB integration to efficiently store and manage essential data.

3. Enable RESTful user interface communication to enhance system accessibility.

4. Demonstrate the robot's capability to execute specific actions based on received commands.

5. Develop a prototype of the system using ROS2:
   Explain the decision to use ROS2 as the framework for prototyping the system.
   Outline the specific ROS2 components and functionalities that will be
   implemented in the prototype.
   Discuss how the ROS2 prototype will interact with other components of the system , such as MongoDB integration and RESTful communication.
   Detail the expected benefits of using ROS2 for prototyping, such as its robustness in managing robot behavior and its compatibility with various hardware platforms.

- **Feasibility Study:**
  The feasibility study will assess the technical, economic, and operational viability of the proposed middleware system. It will include an analysis of existing technologies, market trends, and user requirements to determine the project's potential for success.