# Chapter III: State of the Art

## Introduction

## I. Presenting Orchestration of Automation Processes

In today's digital era, the automation of repetitive tasks has become crucial for enhancing productivity and efficiency. Orchestration of automation processes refers to the coordination and management of automated tasks, ensuring that they run smoothly and efficiently. This chapter explores various methods of implementing automation, the role of AI in advancing automation scripts, and the rationale behind the chosen approach for this project.

## II. Methods of Implementing Automation

### 1. Scripting and Manual Automation

#### a. Shell Scripts

Shell scripts are a traditional method for automating tasks on Unix-based systems. They are simple to write and can automate command-line tasks, making them suitable for system administration and simple automation needs. However, shell scripts are limited in terms of flexibility and power compared to more modern scripting languages.

#### b. Python Scripts

Python has emerged as a popular language for automation due to its readability, versatility, and extensive library support. Python scripts can automate a wide range of tasks, from simple file manipulation to complex data processing and web scraping. Libraries such as smtplib for sending emails, requests for web interactions, and os for file operations enhance Python's capabilities in automation.

### 2. Automation Frameworks and Tools

#### a. Ansible

Ansible is an open-source automation tool that enables configuration management, application deployment, and task automation. It uses YAML for configuration and can manage systems over SSH without requiring a client on the remote systems. Ansible's simplicity and agentless architecture make it a popular choice for IT automation.

#### b. Jenkins

Jenkins is a popular automation server used for continuous integration and continuous delivery (CI/CD). It automates the building, testing, and deployment of applications, ensuring that code changes are tested and integrated continuously. Jenkins supports numerous plugins, making it highly customizable for various automation needs.

#### c. Docker

Docker is a platform that enables developers to create, deploy, and run applications in containers. Containers encapsulate an application and its dependencies, ensuring consistency across different environments. Docker simplifies the process of building and deploying applications, making it a key tool in modern DevOps practices.

#### d. Kubernetes

Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. It helps in managing complex microservices architectures, providing features like automated rollouts and rollbacks, service discovery, and load balancing.

#### e. UiPath

UiPath is a leading Robotic Process Automation (RPA) tool that allows businesses to automate repetitive tasks across various applications and systems. UiPath provides a user-friendly interface for designing automation workflows and integrates with numerous applications, making it a versatile tool for enterprise automation.

### 3. Robotic Process Automation (RPA)

RPA involves the use of software robots or "bots" to automate high-volume, repetitive tasks that traditionally require human intervention. RPA tools like UiPath, Blue Prism, and Automation Anywhere can mimic human actions within digital systems, interacting with applications, manipulating data, and communicating with other systems. RPA reduces errors, increases efficiency, and allows employees to focus on more strategic tasks.

#### a. Core Concepts of RPA

* **Bot Creation**: RPA bots are created to perform specific tasks. These bots can be programmed using visual interfaces or code to interact with various software systems, execute processes, and make decisions based on predefined rules.
* **Integration with Existing Systems**: RPA tools can integrate with existing enterprise applications, databases, and systems without requiring extensive changes to the underlying infrastructure. This makes RPA an attractive solution for automating legacy systems.
* **Non-Intrusive Nature**: RPA operates at the user interface level, interacting with applications just like a human would. This non-intrusive approach ensures that RPA can be implemented quickly without disrupting existing processes.

#### b. Use Cases of RPA

* **Finance and Accounting**: Automating invoice processing, reconciliations, and financial reporting.
* **Customer Service**: Automating responses to common customer inquiries, processing service requests, and managing customer accounts.
* **Human Resources**: Automating employee onboarding, payroll processing, and benefits administration.
* **Supply Chain Management**: Automating order processing, inventory management, and logistics coordination.

#### c. Benefits of RPA

* **Increased Efficiency**: RPA bots can perform tasks faster and more accurately than humans, leading to significant time savings and productivity gains.
* **Cost Savings**: By automating repetitive tasks, organizations can reduce labor costs and redirect human resources to more value-added activities.
* **Scalability**: RPA solutions can be scaled easily to handle increased workloads without requiring proportional increases in human resources.
* **Improved Accuracy**: RPA reduces the risk of human error, ensuring that tasks are completed consistently and accurately.

#### d. Challenges and Considerations

* **Implementation Costs**: Initial setup and configuration of RPA solutions can be costly and time-consuming.
* **Change Management**: Employees may resist changes brought about by automation. Effective change management strategies are essential to ensure a smooth transition.
* **Maintenance and Monitoring**: RPA bots require ongoing maintenance and monitoring to ensure they continue to function correctly and adapt to changes in underlying systems.

## III. Role of AI in Automation

Artificial Intelligence (AI) has revolutionized the field of automation by enabling more sophisticated, adaptable, and intelligent automation solutions. Here, we delve deeper into the role of AI in automation, focusing on machine learning, AI-powered bots, and recent advances in AI technologies.

### 1. Machine Learning and Automation

Machine learning (ML) is a subset of AI that focuses on developing algorithms that allow computers to learn from and make decisions based on data. This capability has significantly enhanced automation by enabling systems to perform tasks that are too complex for traditional rule-based automation.

#### a. Predictive Analytics

One of the key applications of ML in automation is predictive analytics. By analyzing historical data, ML models can predict future events and trends. For example, in predictive maintenance, ML models analyze data from sensors on machinery to predict equipment failures before they happen. This proactive approach helps prevent downtime and reduces maintenance costs.

#### b. Anomaly Detection

ML is also used for anomaly detection in automation. ML algorithms can be trained to recognize patterns in data and identify deviations from these patterns. This is useful in applications such as fraud detection, network security, and quality control in manufacturing.

#### c. Optimization

ML algorithms can optimize processes by analyzing data and identifying the most efficient ways to perform tasks. For instance, in supply chain management, ML can optimize inventory levels, delivery routes, and order processing to improve efficiency and reduce costs.

### 2. AI-Powered Bots

AI-powered bots, also known as intelligent agents, leverage AI technologies such as natural language processing (NLP) and machine learning to perform complex tasks that require understanding and generating human-like text.

#### a. Natural Language Processing (NLP)

NLP enables bots to understand and process human language. This allows bots to interact with users in a more natural and intuitive way. Applications of NLP in automation include:

* **Chatbots**: AI-powered chatbots can handle customer inquiries, provide support, and engage with users in real-time. They can understand user inputs, retrieve relevant information, and respond appropriately.
* **Text Analysis**: Bots can analyze large volumes of text data to extract meaningful information. For example, sentiment analysis bots can determine the sentiment of customer reviews, helping businesses understand customer opinions.

#### b. Machine Learning Integration

AI-powered bots can use ML algorithms to improve their performance over time. By learning from interactions and feedback, these bots can become more accurate and effective. For example, customer service bots can learn from previous conversations to provide better responses in future interactions.

### 3. Advances in AI for Automation

Recent advances in AI have further expanded the capabilities and applications of automation. Key advancements include:

#### a. Deep Learning

Deep learning, a subset of ML, involves neural networks with many layers that can model complex patterns in data. Deep learning has led to significant breakthroughs in image and speech recognition, making it possible to automate tasks that were previously difficult or impossible.

* **Image Recognition**: AI systems can now recognize and classify images with high accuracy. This capability is used in automated quality inspection in manufacturing, medical imaging analysis, and autonomous vehicles.
* **Speech Recognition**: AI-powered speech recognition systems can transcribe spoken language into text and understand commands. This technology is used in virtual assistants like Siri and Alexa, as well as in automated customer service systems.

#### b. Reinforcement Learning

Reinforcement learning (RL) is an area of ML where agents learn to make decisions by interacting with their environment and receiving feedback in the form of rewards or penalties. RL has been used to develop AI systems that can play games, control robotic systems, and optimize complex processes.

* **Robotics**: RL enables robots to learn and improve their performance in tasks such as assembly, picking and placing items, and navigating environments.
* **Process Optimization**: RL can optimize dynamic processes in areas such as energy management, financial trading, and supply chain logistics.

#### c. Transfer Learning

Transfer learning involves using pre-trained models for new tasks, significantly reducing the amount of data and time required to train AI systems. This approach has accelerated the development of AI applications in various domains.

* **Natural Language Processing**: Pre-trained language models like GPT-4 can be fine-tuned for specific tasks such as summarization, translation, and question answering.
* **Computer Vision**: Pre-trained vision models can be adapted for applications like object detection, image segmentation, and facial recognition.

### 4. AI in Specific Automation Scenarios

#### a. Intelligent Process Automation (IPA)

IPA combines RPA with AI technologies such as ML and NLP to create more intelligent and adaptive automation solutions. IPA can handle unstructured data, make decisions based on data analysis, and continuously improve processes.

* **Document Processing**: IPA can automate the extraction of information from unstructured documents, such as invoices and contracts, using NLP and computer vision.
* **Customer Service**: IPA can enhance customer service automation by combining chatbots with sentiment analysis and personalized recommendations.

#### b. Cognitive Automation

Cognitive automation refers to the use of AI to simulate human thought processes in automation. This involves the use of cognitive technologies such as ML, NLP, and computer vision to understand, reason, and learn from data.

* **Decision Support**: Cognitive automation systems can analyze data and provide insights and recommendations to support decision-making in areas like healthcare, finance, and manufacturing.
* **Knowledge Management**: Cognitive automation can manage and utilize large volumes of information, helping organizations retain and apply knowledge effectively.

## IV. Development of the Project's Bots

### 1. Mail Sender Bot

The Mail Sender bot automates the process of sending emails. It takes the subject, body of the email, and recipient addresses from a configuration file and uses the smtplib library to send emails. This bot was developed to streamline email communications and reduce manual effort. The script ensures secure connections using the ssl module and handles potential errors such as incorrect email addresses or connection issues.

### 2. Chat Bot

The Chat Bot uses an API (<https://simple-chatgpt-api.p.rapidapi.com/ask>) to send questions from the configuration file and returns answers. This bot leverages AI to provide intelligent responses, making it suitable for customer support and information retrieval. The development involved integrating the API and handling JSON responses to extract meaningful answers. Error handling mechanisms were also implemented to manage API rate limits and potential downtime.

### 3. Log Analyzer

The Log Analyzer bot processes log files and counts the number of errors and warnings. It helps in monitoring and maintaining system health by providing insights into the frequency and types of issues occurring in the logs. The script reads log files, parses them, and uses regular expressions to identify and count error and warning messages. Additionally, it generates summary reports that can be used for further analysis.

### 4. Resume Reviewer

The Resume Reviewer bot uses the OpenAI API to evaluate resumes based on job descriptions and return a score out of 10. This bot assists HR departments in screening resumes more efficiently and objectively. The script sends resume and job description texts to the API and processes the response to generate a score. It also provides feedback on areas of improvement for the candidates, helping them enhance their resumes.

### 5. Text Summarizer

The Text Summarizer bot utilizes the OpenAI API to summarize text provided in the configuration file. This bot aids in quickly understanding large volumes of text by providing concise summaries. The development involved sending the text to the API, receiving the summary, and formatting it for presentation. This bot is particularly useful for processing lengthy documents and extracting key information.

## IV. Development of the Project's Bots

The development of the project’s bots involved leveraging Python’s extensive library ecosystem, integrating with various APIs, and implementing robust error handling and data processing techniques. Each bot was designed to perform specific automation tasks, streamlining processes, and improving efficiency.

### 1. Mail Sender Bot

The Mail Sender bot automates the process of sending emails, simplifying communication tasks. Here’s a detailed breakdown of its development:

#### a. Functionality

* **Input**: The bot receives a configuration file containing the subject, body of the email, and recipient addresses.
* **Process**: It uses the smtplib library to connect to an SMTP server and send emails.
* **Output**: Emails are sent to the specified recipients.

#### b. Implementation Details

* **Reading Configuration**: The bot reads a JSON configuration file using the json library. This file includes the email subject, body, sender credentials, and recipient list.
* **SMTP Setup**: Using smtplib, the bot establishes a secure connection to the SMTP server. It uses the ssl module to wrap the connection in a secure socket layer.
* **Sending Emails**: The bot iterates through the recipient list, sending personalized emails. It handles exceptions such as connection errors or incorrect email formats using try-except blocks.

### 2. Chat Bot

The Chat Bot leverages an external API to provide intelligent responses based on user queries. Here’s an in-depth look at its development:

#### a. Functionality

* **Input**: A configuration file containing a list of questions.
* **Process**: The bot sends these questions to the simple ChatGPT API and retrieves responses.
* **Output**: The bot returns the responses for display or further processing.

#### b. Implementation Details

* **API Integration**: The bot uses the requests library to interact with the Simple ChatGPT API. It sends HTTP requests with the questions and handles the responses.
* **Configuration Parsing**: The bot reads the configuration file to extract questions, using json for parsing.
* **Response Handling**: After receiving responses from the API, the bot processes the JSON data to extract the answers.

### 3. Log Analyzer

The Log Analyzer bot automates the process of analyzing log files, identifying errors and warnings. Here’s a detailed breakdown:

#### a. Functionality

* **Input**: Log files to be analyzed.
* **Process**: The bot reads and parses these log files, counting occurrences of errors and warnings.
* **Output**: A summary report detailing the number of errors and warnings.

#### b. Implementation Details

* **Reading the Log File:** The bot opens the specified log file and goes through its content line by line.
* **Counting Errors and Warnings:** As it reads each line, the bot looks for specific words, 'Error' and 'Warning'. Each time it finds 'Error', it increases the error count by one. Similarly, each time it finds 'Warning', it increases the warning count by one.
* **Generating the Report:** After it has read all the lines in the log file, the bot prepares a summary report. This report includes the total number of errors and warnings found.
* **Command-Line Interface:** The bot can be run from the command line. When running the bot, you need to provide the path to the log file you want to analyze. If you don't provide a log file path, the bot will show a message explaining how to use it. When a log file is provided, the bot analyzes it and then prints out the summary report in a structured format.

### 4. Resume Reviewer

The Resume Reviewer bot uses the OpenAI API to evaluate resumes based on job descriptions, providing a score and feedback. Here’s an in-depth look:

#### a. Functionality

* **Input**: A configuration file containing the resume.
* **Process**: The bot sends the resume and the job description to the OpenAI API for analysis.
* **Output**: A score out of 10 and detailed feedback on the resume.

#### b. Implementation Details

* **API Integration**: The bot uses the requests library to interact with the OpenAI API. It sends the resume and job description texts and processes the response.
* **Configuration Parsing**: The bot reads the input texts from a configuration file using the json library.
* **Response Processing**: The bot extracts the score and feedback from the API response and formats it for presentation.

#### c. Error Handling and Customization

* **Error Handling**: The bot includes mechanisms to handle API errors, network issues, and unexpected responses.
* **Customization**: Parameters like scoring criteria and feedback verbosity can be customized to meet specific requirements.

### 5. Text Summarizer

The Text Summarizer bot leverages the OpenAI API to generate concise summaries of input texts. Here’s a detailed breakdown:

#### a. Functionality

* **Input**: A configuration file containing the text to be summarized.
* **Process**: The bot sends the text to the OpenAI API and retrieves the summary.
* **Output**: A concise summary of the input text.

#### b. Implementation Details

* **API Integration**: The bot uses the requests library to send the input text to the OpenAI API and retrieve the summary.
* **Configuration Parsing**: The bot reads the text from a configuration file using the json library.
* **Response Handling**: The bot processes the API response to extract and format the summary for display or further use.

#### c. Enhancements and Extensions

* **Custom Summary Length**: The bot allows customization of summary length based on user requirements.
* **Multi-Document Summarization**: Future enhancements could include the ability to summarize multiple documents and compile a comprehensive summary.

## V. Rationale Behind the Chosen Approach

### 1. Simplicity and Flexibility

The chosen approach of using standalone Python scripts for automation was driven by the need for simplicity and flexibility. This section delves into why this approach was suitable for the project and how it meets various requirements.

#### a. Ease of Use

Python is renowned for its readability and simplicity, making it an ideal choice for rapid development and iteration. Its syntax is clear and concise, which reduces the learning curve for developers and allows for faster implementation of automation tasks.

* **Rapid Prototyping**: Python’s simplicity enables quick prototyping of automation scripts. Developers can quickly write, test, and refine scripts without getting bogged down by complex syntax or verbose code.
* **Minimal Overhead**: Using standalone Python scripts avoids the overhead associated with setting up and maintaining complex frameworks or environments. This allows the team to focus on the core functionality of the automation tasks.

#### b. Versatility

Python’s extensive standard library and third-party packages provide tools for almost any task, from web scraping to machine learning. This versatility is crucial for developing a wide range of bots.

* **Comprehensive Libraries**: Libraries such as requests for HTTP requests, smtplib for email handling, and re for regular expressions cover most automation needs without requiring additional tools.
* **Integration Capabilities**: Python’s ability to integrate with various APIs and external services makes it easy to extend the functionality of the bots. For instance, the integration with OpenAI API for NLP tasks and the Simple ChatGPT API for conversational bots.

### 2. Team Manager's Requirements

The project was also guided by specific requirements from the team manager, emphasizing a straightforward and practical solution.

#### a. Simplicity Over Complexity

The team manager prioritized a simple, easily maintainable solution over a more complex system. This decision was based on the need to deliver a functional product within a limited timeframe and ensure that it could be easily managed by the team.

* **Straightforward Implementation**: By choosing a straightforward approach, the team avoided the complexities and potential pitfalls of more sophisticated frameworks or tools that could have introduced unnecessary complexity.
* **Ease of Handoff**: Simpler solutions are easier to hand off to new team members or maintainers, ensuring continuity and ease of future modifications or extensions.

#### b. Resource Constraints

Considering the constraints on time, budget, and available resources, the chosen approach aligned well with the project’s scope.

* **Cost-Effective**: Utilizing Python scripts avoided the need for expensive software licenses or extensive infrastructure. Open-source libraries and tools further minimized costs.
* **Time Efficiency**: The team could focus on developing core functionalities without spending excessive time on configuring and learning new tools or frameworks.

### 3. Scalability and Maintainability

Using standalone Python scripts also offers advantages in terms of scalability and maintainability, which are crucial for the long-term success of the project.

#### a. Modular Design

The modular nature of Python scripts allows for easy addition, modification, and extension of functionality.

* **Adding New Bots**: New bots can be developed and added to the system without significant changes to the existing architecture. Each bot operates independently, reducing the risk of introducing bugs or conflicts.
* **Modifying Existing Bots**: Updates or changes to individual bots can be made with minimal impact on the overall system. This modular approach ensures that enhancements or fixes can be deployed quickly and efficiently.

#### b. Maintenance and Debugging

Python’s straightforward syntax and comprehensive error handling make maintenance and debugging more manageable.

* **Readable Code**: The readability of Python code simplifies the process of understanding and modifying scripts. Clear, well-documented code is easier to maintain and debug.
* **Robust Error Handling**: Python’s exception handling mechanisms allow for the graceful handling of errors, ensuring that the system remains robust and reliable.

#### c. Community and Support

Python has a large and active community, providing extensive resources for troubleshooting, learning, and extending capabilities.

* **Community Support**: The vast Python community offers forums, tutorials, and documentation that can assist in resolving issues and improving the bots.
* **Library Ecosystem**: The availability of numerous libraries and frameworks developed by the community can be leveraged to enhance the functionality of the bots without reinventing the wheel.

### 4. Alignment with Project Goals

The chosen approach aligns well with the overarching goals of the project, ensuring that it meets the requirements effectively and efficiently.

#### a. Functional Completeness

The approach ensures that all required functionalities are implemented and work as expected. Each bot performs its designated task effectively, contributing to the overall goal of automating various processes.

#### b. Timely Delivery

By adopting a straightforward and manageable approach, the team ensured that the project was completed within the allocated timeframe. This was crucial for meeting deadlines and delivering a functional product on schedule.

#### c. User-Friendly Interface

The simplicity of the backend implementation allowed the team to focus on creating a user-friendly interface. This ensures that both administrators and users can interact with the system easily, without requiring technical expertise.

## Conclusion

This chapter has explored the various methods of implementing automation, the impact of AI on automation processes, and the specific development and functionality of the bots in this project. The chosen approach of using Python scripts aligns with the project's goals of simplicity, flexibility, and meeting the team manager's requirements. By leveraging Python and AI, the project effectively orchestrates automation processes, enhancing productivity and efficiency.

**Sprint 2**

### I. Introduction

In Sprint 2 of our project, we focused on implementing features for managing bots and user activities. This included developing an admin interface for CRUD (Create, Read, Update, Delete) operations on bots and creating detailed activity logs for both admins and users.

* Global use case

A diagram of a diagram

Description automatically generated

### II. Sprint Backlog

For Sprint 2, the backlog consists of the following user stories and tasks:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | Actor | User Story | Task Description | Priority |
| Manage Bots | Admin | |  | | --- | |  |  |  | | --- | | As an admin, I want to create new bots | | - Design bot creation form  - Implement backend API for bot creation  - Validate bot attributes  - Integrate form with backend | High |
| As an admin, I want to update existing bots | - Design bot update form  - Implement backend API for bot update  - Validate updates  - Integrate form with backend | High |
| As an admin, I want to delete bots | |  | | --- | | - Implement delete functionality in backend API  - Add delete option in admin interface  - Confirm deletion | | Medium |
| As an admin, I want to view a list of all bots | --Design list view for bots with pagination , search bar and a Search By option  - Implement backend API to fetch bots  - Display bots in admin interface | High |
| User | As a user, I want to view a list of all available bots based on my group | - Implement backend API to fetch bots based on user groups  - Display bots list in user interface with pagination , a search bar and a search by option  - Ensure correct group-based access | High |
| Manage Activities | Admin | As an admin, I want to view logs of all user activities, including bot starts, configurations used, and outcomes | - Implement backend API to fetch users activities  - Display activities list  - Add filtering options | Medium |
| User | As a user, I want to view my own activity logs to see when I started bots, the configurations used, and the outcomes | - Implement backend API to fetch the user activities  - Display activities list  - Add filtering options | High |
| As a user, I want to start a bot by uploading a configuration file | - Design bot execution form  - Implement backend API for bot execution  - Handle file uploads  - Validate and process configuration  -Start bot | High |
| As a user, I want to schedule a bot to start at a specific time | - Implement scheduling functionality in backend  - Design scheduling interface  - Validate scheduling input  - Integrate with backend | Medium |

### III. Conception

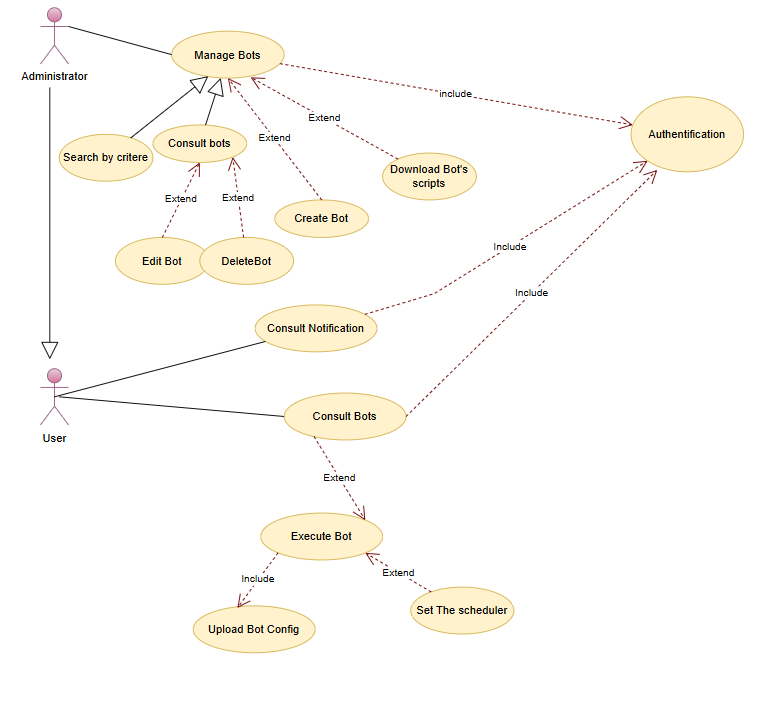
#### 1. Use Case Diagram

### A-Manage Bots

Refinement of the use case “Manage bots”

Top of Form

Bottom of Form



### Use Case Table: Common Actions (Admin and User)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Use Case** |  |  | | --- | |  | | | **Consult Bots** | | --- |  |  | | --- | |  | |
| |  | | --- | | **Actors** |  |  | | --- | |  | | |  | | --- | | Administrator, User |  |  | | --- | |  | |
| |  | | --- | | **Pre-Condition** |  |  | | --- | |  | | |  | | --- | | Actors must be authenticated. |  |  | | --- | |  | |
| |  | | --- | | **Post-Condition** |  |  | | --- | |  | | |  | | --- | | List of bots is displayed. |  |  | | --- | |  | |
| **Primary Scenario Description** | 1. The actor authenticates.  2. The actor navigates to the bots section.  3. The system retrieves and displays the list of available bots based on the actor's group. |
| **Exception Scenario** | 1. Authentication fails: The system prompts for re-authentication.  2. No bots available: The system displays a message indicating no bots are available for the user's group. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Use Case** |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | | **Execute Bot** | | --- |  |  | | --- | |  | |  |  | | --- | |  | |
| |  | | --- | | **Actors** |  |  | | --- | |  | | |  | | --- | | Administrator, User |  |  | | --- | |  | |
| |  | | --- | | **Pre-Condition** |  |  | | --- | |  | | |  | | --- | | Actors must be authenticated. The bot configuration must be uploaded. |  |  | | --- | |  | |
| |  | | --- | | **Post-Condition** |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Bot is executed and the result is logged. |  |  | | --- | |  | |  |  | | --- | |  | |
| **Primary Scenario Description** | 1. The actor authenticates  2. The actor consults the list of bots  3. The actor selects a bot to execute.  4. The actor uploads a bot configuration  5. The system executes the bot with the provided configuration  6. The system logs the execution result and displays it to the actor. |
| **Exception Scenario** | 1. Authentication fails: The system prompts for re-authentication.  2. Invalid configuration: The system shows an error message and prompts for correct configuration upload. |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Use Case** |  |  | | --- | |  | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | | **Set the Scheduler** | | --- |  |  | | --- | |  | |  |  | | --- | |  | |  |  | | --- | |  | |
| |  | | --- | | **Actors** |  |  | | --- | |  | | |  | | --- | | Administrator, User |  |  | | --- | |  | |
| |  | | --- | | **Pre-Condition** |  |  | | --- | |  | | |  | | --- | | Actors must be authenticated. |  |  | | --- | |  | |
| |  | | --- | | **Post-Condition** |  |  | | --- | |  | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | |  | | --- | | Bot is scheduled for execution at a specified time. |  |  | | --- | |  | |  |  | | --- | |  | |  |  | | --- | |  | |
| **Primary Scenario Description** | 1. The actor authenticates.  2. The actor consults the list of bots.  3. The actor selects a bot to schedule.  4. The actor sets the scheduling time.  5. The system saves the schedule and confirms the scheduling. |
| **Exception Scenario** | 1. Authentication fails: The system prompts for re-authentication.  2. Invalid scheduling time: The system shows an error message and prompts for correct scheduling time. |

### Use Case Table: Administrator Actions

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Use Case** |  |  | | --- | |  | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | | **Set the Scheduler** | | --- |  |  | | --- | |  | |  |  | | --- | |  | |  |  | | --- | |  | |
| |  | | --- | | **Actors** |  |  | | --- | |  | | |  | | --- | | Administrator |  |  | | --- | |  | |
| |  | | --- | | **Pre-Condition** |  |  | | --- | |  | | |  | | --- | | Administrator must be authenticated. |  |  | | --- | |  | |
| |  | | --- | | **Post-Condition** |  |  | | --- | |  | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | |  | | --- | | Bot is scheduled for execution at a specified time. |  |  | | --- | |  | |  |  | | --- | |  | |  |  | | --- | |  | |
| **Primary Scenario Description** | 1. The actor authenticates.  2. The actor consults the list of bots.  3. The actor selects a bot to schedule.  4. The actor sets the scheduling time.  5. The system saves the schedule and confirms the scheduling. |
| **Exception Scenario** | 1. Authentication fails: The system prompts for re-authentication.  2. Invalid scheduling time: The system shows an error message and prompts for correct scheduling time. |

B-Manage Activities

Refinement of the use case “Manage Activites”

