**Assignment 2**

My project, Discord Bot Automation Assistant, is designed to monitor process but also will have additional features. Other than tracking prices of products online, it can also list and filter products we are searching. The bot can respond or send simple, straightforward notifications like "Price Dropped" to more complex actions such as creating Excel sheets. Additionally, it will offer the flexibility to log in with different credentials and run unattended at designated times.

**1 - Actors***:* Actors are the users or entities interacting with the system. In my project priceTracker, when introduction considered, actors can be list like following:

* *User*: The person who would like to use the features of this project.
* *Bot*: Web scraping, temporary data storage, generates responses for specific messages, initializing discord bot with discord library of python.
* *External Helpers (Websites/APIs):* Represents the external websites or APIs from which the bot retrieves product information.

**2 - Use Cases:**

Use cases represent the various interactions and functionalities that priceTracker bot will support.

* *NavigateToWebsite*
* *LoginToWebApplication*
* *ChatWithBot*
* *CheckPriceOfTheProduct*
* *CheckAvailableDates*
* *ExtractDataToExcel*
* *ReceiveNotificationsOrEmails*

**3 - Relationship between actors and use cases, and documentation:**

1. *NavigateToWebsite*

**Actor**: User  
**Actor Description**: Initiates the " *NavigateToWebsite*" use case by interacting with the Price Tracker bot.  
**Communicates** **With:** Bot, External Helpers (Websites/APIs)

**Flow of Events**:

* The user initiates the " *NavigateToWebsite* " function of the Price Tracker bot.
* The Price Tracker bot responds by launching the Chrome browser.
* The bot navigates to the specified web application or URL.
* The bot communicates with external helpers (websites/APIs) to retrieve necessary information or perform actions within the Chrome browser.

**Entry Condition:** The user interacts with the Price Tracker bot.

**Exit Condition:** The Chrome browser is successfully launched and navigated to URL, and the bot is ready to perform further actions.

**Quality Requirements:**

* The Chrome browser should be launched and navigated within a reasonable time frame, ensuring a smooth user experience.

1. *LoginToWebApplication*

**Actor**: User  
**Actor Description**: Initiates the "LoginToWebApplication" use case by interacting with the Price Tracker bot.  
**Communicates** **With:** Bot, External Helpers (Websites/APIs)

**Flow of Events**:

* The user initiates the "LoginToWebApplication" function of the Price Tracker bot.
* The Price Tracker bot presents a login form or prompts the user to enter their credentials.
* The user enters their login credentials (e.g., username and password).
* The bot submits the login form or sends the user's credentials to the specified web application.
* The bot interacts with the web application's authentication mechanism to verify the user's credentials.
* Upon successful authentication, the bot navigates the user to the authenticated area of the web application.

**Entry Condition:** The user interacts with the Price Tracker bot.

**Exit Condition:** The user is successfully logged in to the specified web application, and the bot is ready to perform further actions.

**Quality Requirements:**

* The login process should be secure and encrypted to protect the user's credentials.
* The user should be redirected to the appropriate page or dashboard within the web application upon successful login.

1. *ChatWithBot*

**Actor**: User  
**Actor Description**: Initiates the "ChatWithBot" by interacting with the bot.  
**Communicates** **With:** Bot

**Flow of Events**:

* The user initiates the "ChatWithBot" function by sending a message or command to the Price Tracker bot.
* The Price Tracker bot receives the user's message and processes it to determine the user's intent.
* The bot analyzes the user's message to identify the appropriate action or response.
  + If the message corresponds to a recognized command or query:
    - The bot executes the corresponding action, such as checking prices or retrieving information.
    - If necessary, the bot interacts with external helpers, such as websites or APIs, to fulfill the user's request.
    - The bot generates a response based on the results of the action or query.
  + If the message does not correspond to a recognized command or query:
    - The bot responds with a message indicating that the command is not understood or supported.
* The bot sends the response back to the user, typically in the form of a text message or notification within the chat interface.
* The user continues the conversation with the bot by sending additional messages or commands as needed.

**Entry Condition:** The user interacts with the Price Tracker bot by sending a message or command.

**Exit Condition:** The user receives a response from the bot, and the conversation may continue or conclude based on the user's needs.

***Commands/Queries bot recognizes:***

Hello, CheckPriceOfTheProduct, CheckAvailableDates. Also recognizes: Yes, No, “InputURL” if the asks you questions.

The rest of the test cases are sub UseCases of ChatWithBot. And “Actor”, “Actor Description”, Communicates with same/common in all of them:

**Actor**: Bot  
**Actor Description**: Initiates the "UseCase" by interacting with the Price Tracker bot.  
**Communicates** **With:** User, External Helpers (Websites/APIs)

* 1. *ExtractDataToExcel*

**Flow of Events**:

* The user initiates the "ExtractDataToExcel" function by expressing the desire to export data to an Excel file.
* The Price Tracker bot receives the user's request and asks if the user wants to extract the data to an Excel file.
* The bot waits for the user's response:
  + If the user responds affirmatively, the bot proceeds to generate an Excel file containing the requested data.
  + If the user responds negatively, the bot acknowledges the decision and ends the interaction.
* The bot saves the extracted data into an Excel file located at the predefined path (possibly specified in a configuration file).
* Once the Excel file is created, the bot notifies the user of the successful extraction and provides the path to the file.
* The user receives the notification and can access the Excel file for further analysis or manipulation.

**Entry Condition:** The user expresses the desire to extract data to an Excel file by interacting with the Price Tracker bot.

**Exit Condition:** If the user confirms the extraction, the bot successfully creates the Excel file containing the extracted data and notifies the user. If the user declines the extraction, the bot acknowledges the decision, and the interaction concludes.

**Quality Requirements:**

* The bot should retrieve and extract data accurately from the specified sources.
* The Excel file generated by the bot should be well-formatted and organized, facilitating easy analysis and manipulation by the user.
  1. *ReceiveNotificationsOrEmails*

**Flow of Events**:

* The user configures the Price Tracker bot to send notifications or emails for specific events or updates.
* The bot acknowledges the user's configuration and prepares to send notifications or emails as requested.
* When a relevant event occurs, such as a price change or appointment availability, the bot generates a notification or email.
* The bot sends the notification or email to the user's specified contact information, such as email address or messaging platform.
* The user receives the notification or email and takes appropriate action based on the information provided.

**Entry Condition:** The user configures the Price Tracker bot to send notifications or emails for specific events or updates.

**Exit Condition:** The bot successfully generates and sends notifications or emails for relevant events or updates. The user receives the notification or email from the bot, and the interaction concludes.

**Quality Requirements:**

* The bot should reliably generate and send notifications or emails for relevant events or updates.
* The user should be able to easily configure and modify the notification settings through a user-friendly interface.
* The notification system should be robust and scalable, capable of handling a large volume of notifications without delays or errors.
  1. *CheckPriceOfTheProduct*

**Flow of Events**:

* The user provides the command "CheckPriceOfTheProduct" to the Price Tracker bot.
* The bot responds by asking the user to provide the URL of the product.
* The user provides the URL of the product as input to the bot.
* The bot interacts with external helpers, such as websites or APIs, to retrieve the current price of the specified product using the provided URL.
* Upon successfully retrieving the price information:
  + The bot initiates “ExtractDataToExcel” use case based on the user's selection.
* The bot then asks the user how they want to receive notifications, either via email or Discord message.
  + The bot initiates the "ReceiveNotificationsOrEmails" use case based on the user's selection.
* If the product is found and the price retrieval process is successful:
  + The bot informs the user of the current price of the product.
  + The bot prompts the user to specify the frequency for checking price changes (in minutes).
  + The user provides the desired frequency as input to the bot.
  + The bot sets up automated price checks at the specified frequency and notifies the user of any price changes accordingly.
* If the product is not found or the price retrieval process encounters an error:
  + The bot informs the user that the product could not be found or that an error occurred during the retrieval process.

**Entry Condition:** The user provides the URL of the product to be checked.

**Exit Condition:** The bot sends the current price of the specified product to the user if found; however, if the product is not found or an error occurs during the retrieval process, the bot informs the user of the issue, and the interaction concludes with an error state.

**Quality Requirements:**

* The bot should retrieve the current price of the product accurately and efficiently.
* The user should be able to provide the product details in a clear and understandable format.
* The bot's response should include the current price of the product, along with any relevant details or information.  
  1. *CheckAvailableDates*

**Flow of Events**:

* The user provides the command "CheckAvailableDates" to the Price Tracker bot.
* The bot responds by asking the user to provide the URL of the product.
* The user provides the URL of the product as input to the bot.
* The bot interacts with external helpers, such as websites or APIs, to retrieve the current price of the specified product using the provided URL.
* Upon successfully retrieving the available dates:
  + The bot initiates “ExtractDataToExcel” use case based on the user's selection.
* The bot then asks the user how they want to receive notifications, either via email or Discord message.
  + The bot initiates the "ReceiveNotificationsOrEmails" use case based on the user's selection.
* If the information for available dates received successfully:
  + The bot informs the user of about the available dates.
  + The bot prompts the user to specify the frequency for checking dates.
  + The user provides the desired frequency as input to the bot.
  + The bot sets up automated checks at the specified frequency and notifies the user of any changes accordingly.
* If the available dates are not found or successfully taken, process encounters an error:
  + The bot informs the user that the inforation could not be found or that an error occurred during the retrieval process.

**Entry Condition:** The user interacts with the Price Tracker bot by initiating the "CheckAvailableDates" function.

**Exit Condition:** The user receives information about available dates for the specified appointment or reservation. The interaction with the bot concludes.

**Quality Requirements:**

* The bot should retrieve and present available dates accurately and efficiently.
* The user should be able to specify preferences or criteria for the appointment or reservation, such as location or date range.

Oguz Kaan Yildirim  
Student ID = 307637