MODEL INSTITUTE OF ENGINEERING AND TECHNOLOGY

ACADEMIA ASSIGN PLAGA-BOT

Submitted By:

Name: Palak

Roll No: 2022A6R031

Computer Science Engineering – AI/ML

Semester: 2ND

Course In-charge:

Ms Vani Malagar

Abstract

The Academia Assign Plaga-Bot is a comprehensive solution designed to streamline assignment management processes and promote academic integrity. This report provides an overview of the project, focusing on its four main parts: drive folder creation, submission verification, plagiarism detection, and email notifications. Each part contributes to the overall functionality of the bot, offering a seamless workflow from assignment creation to submission verification, plagiarism checks, and email notifications. The report also highlights the objectives, methodology, and future enhancements of the bot. With its systematic approach and automation capabilities, the bot simplifies administrative tasks for instructors and ensures a fair and efficient assignment process for students, ultimately enhancing the academic workflow.

Table of Contents

Figures

1. Introduction

The Academia Assign Plaga-Bot is a comprehensive solution designed to streamline assignment management processes and promote academic integrity. This report provides an overview of the bot, focusing on its four main parts, each serving a specific purpose and contributing to the overall functionality of the bot.

- Part 1: Drive Folder Creation
- Part 2: Submission Verification
- Part 3: Plagiarism Detection
- Part 4: Email Notification

Part 1 of the bot involves the creation of drive folders and an Excel sheet. This step establishes a structured framework for organizing and managing student assignments. By creating dedicated drive folders for each assignment, instructors can ensure that submissions are properly organized. Additionally, an Excel sheet is generated to store student information and assignment status in a centralized manner.

Once the assignment folders are set up, Part 1 also includes sharing the drive folder link with students. This simplifies the submission process, allowing students to easily access the designated folder and submit their assignments. The seamless collaboration between instructors and students is facilitated through this step.

Part 2 of the bot focuses on verifying assignment submissions. Leveraging roll numbers as unique identifiers, Academia Assign Plaga-Bot performs checks to ensure that each student has submitted their assignment. By comparing the submitted assignment filenames (roll numbers) with the corresponding roll numbers in the Excel sheet, the bot accurately identifies missing or incomplete submissions. This verification process enables instructors to track student progress and take necessary actions.

In Part 3, Academia Assign Plaga-Bot incorporates the Plagiarism Checker X application to address academic integrity concerns. This segment involves conducting plagiarism checks on submitted assignments. The bot analyzes the content of each assignment and generates plagiarism reports, highlighting potential instances of copied or unoriginal work. Students who have not submitted their assignments are promptly notified, urging them to adhere to academic integrity standards and submit their work.

Part 4 of the bot focuses on communication with students through email notifications. Based on the plagiarism check results, Academia Assign Plaga-Bot sends automated emails to students containing their plagiarism reports or certificates. Students receive personalized notifications based on their plagiarism percentage. Those with plagiarism percentages below the defined threshold receive certificates, acknowledging their original work. Students with higher plagiarism percentages receive plagiarism reports, emphasizing the need for improvement.

At the end of the project, Academia Assign Plaga-Bot compiles all the relevant information into a comprehensive Excel sheet. This sheet includes essential details such as roll numbers, names, email addresses, assignment status, plagiarism percentages, checking status, submitted file names, percentage exact values, certificate/report status, and assignment links. This consolidated sheet serves as a valuable resource for instructors to track assignment progress, evaluate student performance, and generate reports as needed.

The Academia Assign Plaga-Bot project encompasses four integral parts that collectively automate and enhance various aspects of assignment management. By providing a seamless workflow from assignment creation to submission verification, plagiarism checks, and email notifications, the bot simplifies administrative tasks for instructors and ensures a fair and efficient assignment process for students. Further exploration of each part in this report will shed light on the technical implementation and the benefits offered by Academia Assign Plaga-Bot .

2. Objectives

- **Streamline assignment submission**: Create a drive folder and Excel sheet for assignments, allowing students to easily submit their work.
- **Ensure assignment completion**: Verify if students have submitted their assignments by checking the roll numbers against the Excel sheet.
- **Promote academic integrity**: Conduct plagiarism checks on submitted assignments using the Plagiarism Checker X application.
- **Enable timely intervention:** Send notifications to students who haven't submitted their assignments to ensure prompt submission.
- **Facilitate communication:** Send email notifications to students with their plagiarism reports or certificates based on their plagiarism percentage.

- **Maintain accurate records**: Update the Excel sheet with assignment status, plagiarism percentage, checking status, submitted file name, and other relevant details.
- **Enhance collaboration and organization:** Share the link to the assignment folder with students for seamless sharing and collaboration.
- **Improve efficiency:** Automate administrative tasks, such as data collection and email notifications, to save time and effort.
- **Provide actionable insights:** Generate detailed reports with assignment details, plagiarism percentages, and certificate/report statuses.
- **Enhance the student experience**: Create a user-friendly and efficient workflow for students, ensuring a positive user experience.

3. Problem Statement

"Streamlining assignment submission and plagiarism detection with **Academia Assign Plaga-Bot** : an intelligent bot for efficient academic workflow."

4. Methodology

- **Requirement Gathering:** Gather specific requirements for the Academia Assign Plaga-Bot, including assignment folder creation, file sharing, submission checks, plagiarism detection, email notifications, and certificate generation.

- **Setting up UiPath Studio:** Install and configure UiPath Studio, the development environment for building automation workflows. Set up the necessary dependencies and libraries required for the Academia Assign Plaga-Bot development.

- **Design and Architecture**: Design the system architecture considering the interaction between different components, such as the Academia Assign Plaga-Bot, Plagiarism Checker X desktop application, Google Drive Explorer extension, GSuite Activities, and email applications. Determine the data flow and integration points between these components.

- **Bot Implementation**: Develop the Academia Assign Plaga-Bot using UiPath Studio, utilizing Python code scripts to display message templates for notifications. Implement functions to interact with the Plagiarism Checker X desktop application and handle the logic for assignment creation, submission checks, plagiarism detection, email notifications, and certificate generation.

- **Integration and APIs:** Integrate the bot with external applications and APIs. Utilize the Drive Explorer extension and GSuite Activities to retrieve file details from Google Sheets and compare them with submission details from the email application. Configure the necessary API keys for GSuite integration.

- **Testing and Validation:** Conduct thorough testing to ensure the functionality of the bot. Test different scenarios, including successful assignment creation, accurate submission checks, plagiarism detection, and email notifications. Validate the integration between the bot and external applications.

- **Deployment and User Interface:** Deploy the Academia Assign Plaga-Bot on your local machine. Consider the user interface requirements, such as displaying relevant information and capturing user inputs. Utilize UiPath Studio features to create an intuitive user interface for interacting with the bot.

- **User Feedback and Iterative Development**: Collect feedback from users and stakeholders, incorporating their suggestions for improving the bot's performance, usability, and integration with external applications. Iterate and update the bot based on user feedback and emerging requirements.

- **Documentation**: Prepare comprehensive documentation outlining the system architecture, implementation details, and instructions for installation, configuration, and operation of the Academia Assign Plaga-Bot. Include details about the integration with Plagiarism Checker X, Drive Explorer extension, GSuite Activities, and email applications. Document the setup and configuration steps for UiPath Studio.

- **Evaluation and Performance Analysis**: Evaluate the bot's performance based on predefined metrics, such as assignment submission accuracy, plagiarism detection

success rate, email delivery success, and user satisfaction. Analyze the collected data to identify areas for improvement and future enhancements.

4.1. UiPath Packages and the Activities Used

The project utilizes various UiPath packages, each serving a specific purpose. The packages and their corresponding activities are listed below:

- **Excel Package**:
  - Read Cell: Reads the value of a specific cell in an Excel worksheet. (Interact with Excel cells)
  - Read Range: Reads a range of cells from an Excel worksheet and stores the data in a DataTable variable. (Read data from Excel)
  - Excel Application Scope: Represents a scope for Excel automation where Excel activities can be executed. (Interact with Excel files)
  - Excel Read Range: Reads a range of cells from an Excel worksheet and stores the data in a DataTable variable. (Read data from Excel)
  - Excel Read Cell: Reads the value of a specific cell in an Excel worksheet. (Interact with Excel cells)
  - Excel Write Cell: Writes a value to a cell in an Excel worksheet. (Modify Excel data)

- **GSuite Package**:
  - GSuite Application Scope: Represents a scope for G Suite automation where G Suite activities can be executed. (Interact with Google Suite applications)

- **UiPath.Core.Activities Package**:
  - Assign: Assigns a value to a variable. (Variable assignment)
  - If: Performs a conditional check and executes different branches based on the condition. (Conditional branching)
  - Write Line: Writes a line of text to the output console. (Output debugging information)
  - Sequence: Groups activities together in a sequence. (Organize activities in a sequence)
  - For Each Row: Iterates over each row in a DataTable variable. (Iterate through rows)
  - Open Application: Opens an application window. (Launch applications)

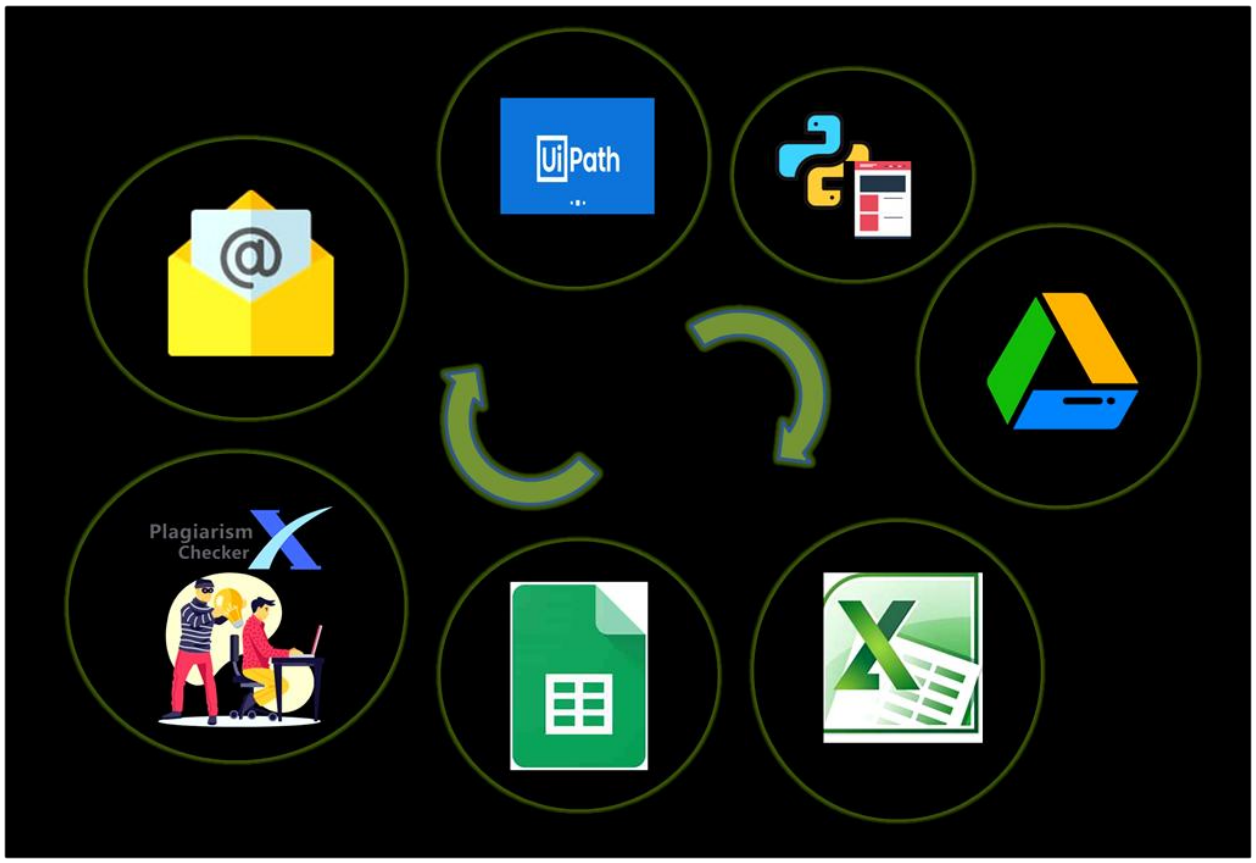
- Switch: Evaluates multiple conditions and executes different branches based on the condition met. (Conditional branching)
- Click: Simulates a click action on a specific element in the target application. (UI automation)
- SendHotkey: Sends a specific hotkey to the target application. (UI automation)
- TypeInto: Simulates typing text into a specific UI element. (UI automation)
- WindowScope: Represents a scope for activities performed within a specific window. (UI automation)
- GetFullText: Retrieves the full text from a target element. (UI automation)

- **System.Activities.Statements Package**:
  - Break: Terminates the nearest enclosing loop or switch statement. (Control flow activity)
  - Try Catch: Defines a try-catch block for handling exceptions. (Exception handling activity)

- **UiPath.Python.Activities Package**:
  - Invoke Python Code: Invokes custom Python code within a workflow. (Integrating Python code with UiPath)
  - Run Python Code: Executes Python code and provides the output. (Running standalone Python scripts)
  - Load Python Code: Loads a Python script from a file. (Loading external Python scripts)
  - Get Python Object: Retrieves the output or value returned by a Python script. (Accessing Python script outputs)

- **UiPath.Mail.Activities Package**:
  - Send SMTP Mail Message: Sends an email message using the SMTP protocol. (Sending email notifications)

5. Workflow



6. Results

Fig 1: Welcome Template

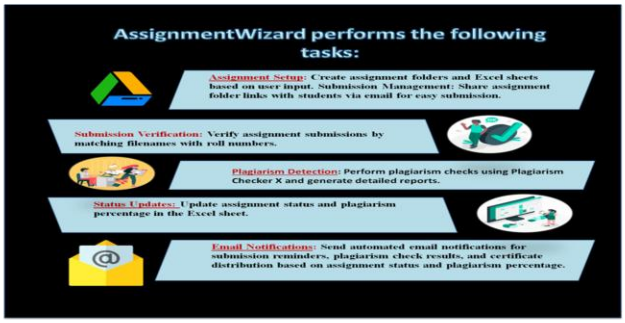


Fig 2 :Information Template
Displaying functionality of the bot

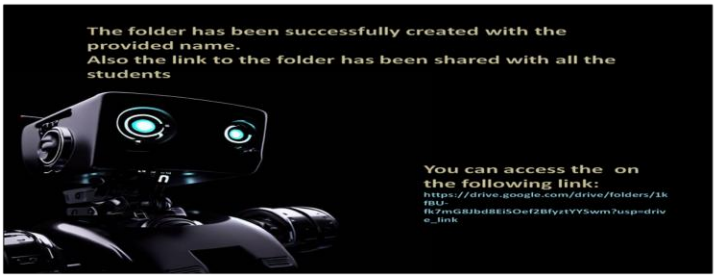


Fig 3: Message Template 1
Informing that the drive folder is created successfully

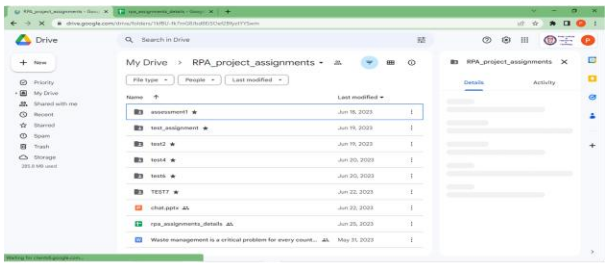


FIG 4: GOOGLE DRIVE FOLDER

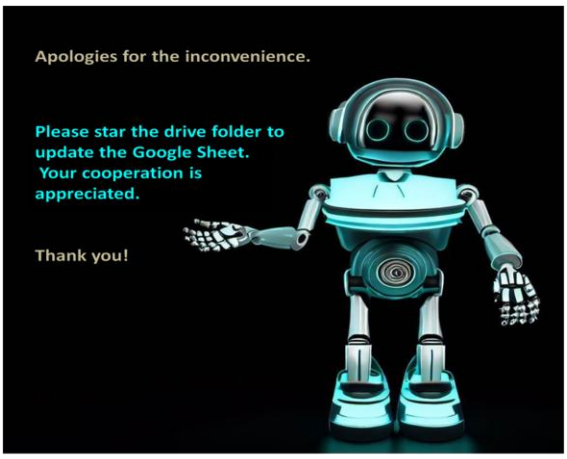


Fig 5: Message Template 2
Requesting user to star the drive folder

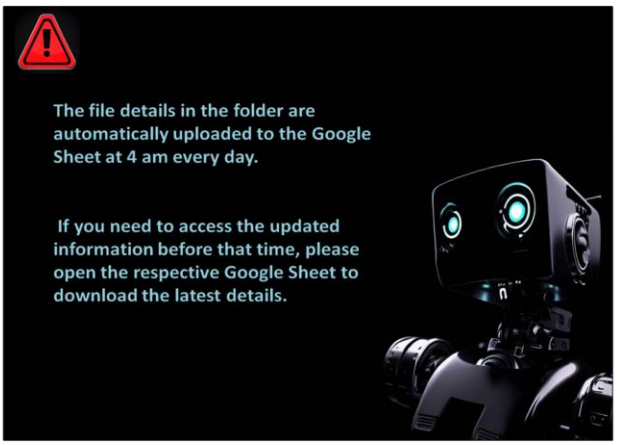


Fig 6: Message Template 3
Giving details about the updation of Google sheet

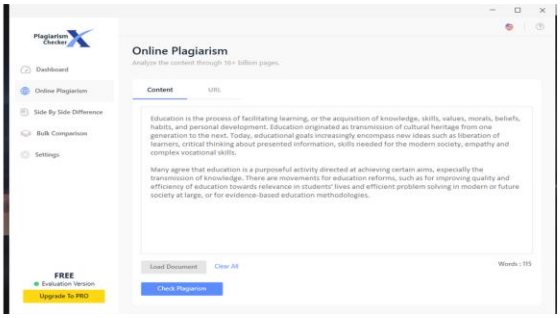


Fig 7: PLAGIARISM CHECKER X
The application used to check plagiarism

Fig 8:THANK YOU TEMPLATE

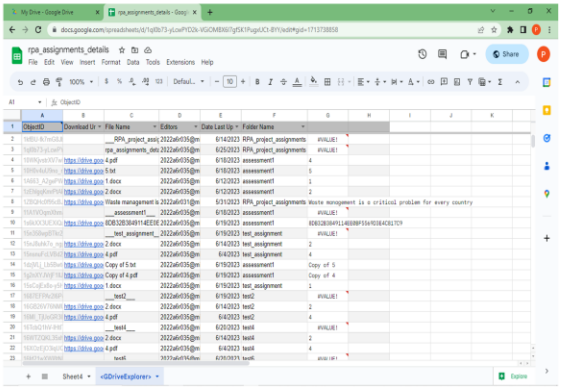


Fig 9: Google Sheet
Contains details about the submitted assignment

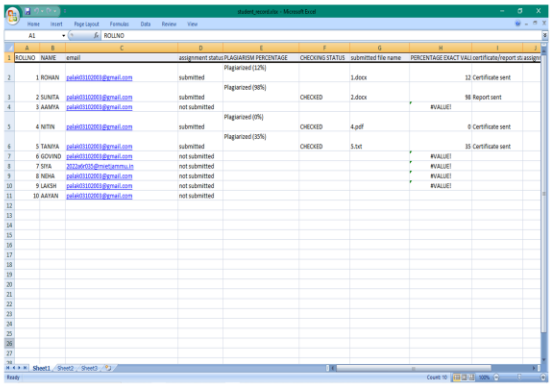


Fig 10: Excel Sheet
Contains all the details about the students and their assignment

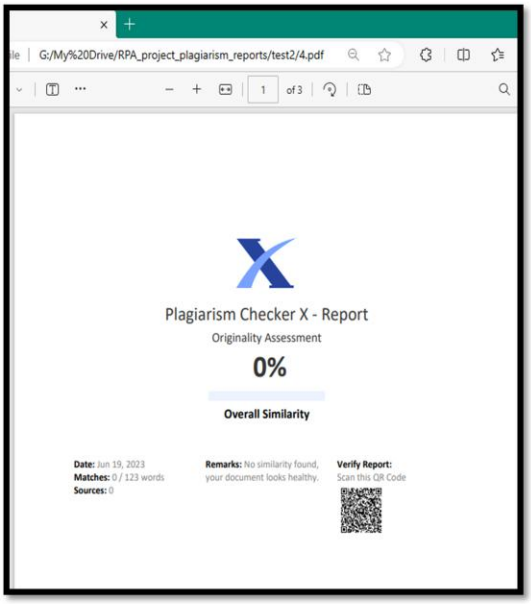


Fig 11: Plagiarism Report 1
Showing results for non plagiarised content

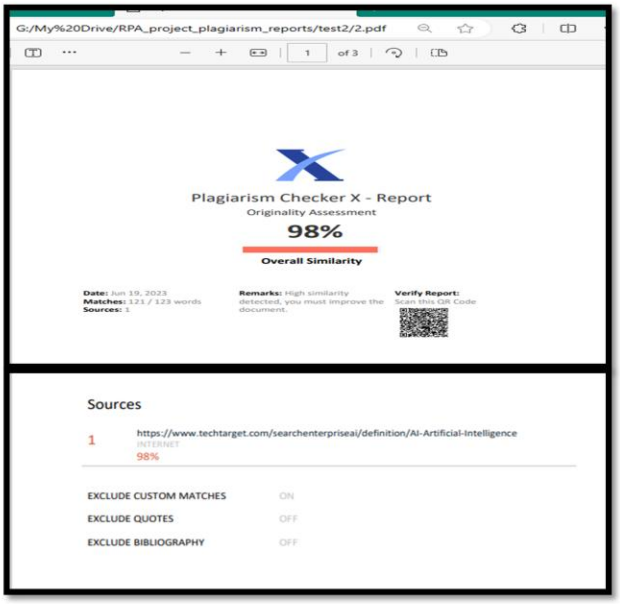
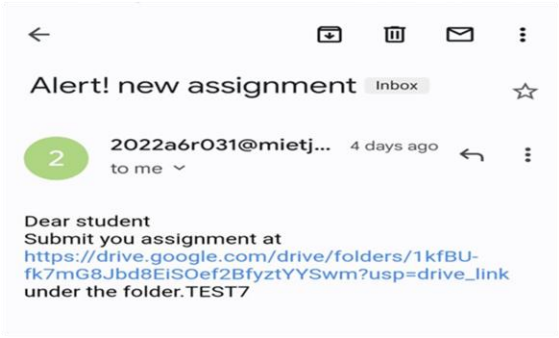
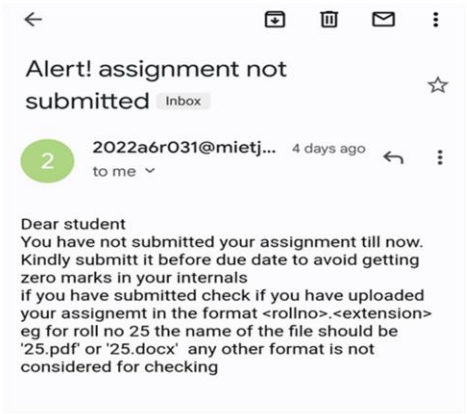


Fig 12: Plagiarism Report 2
Showing results for plagiarised content along with the sources used





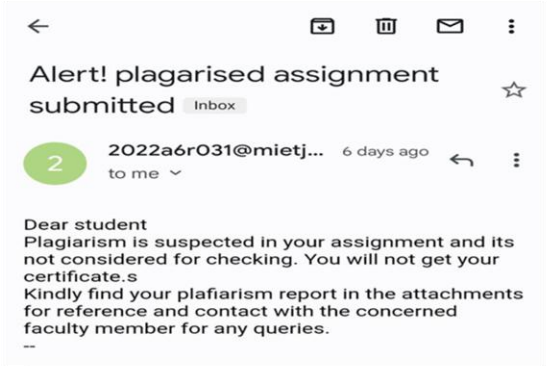


Fig 13: Emails
Different emails sent to students

8. Conclusion

In conclusion, the Academia Assign Plaga-Bot project offers an effective solution for streamlining assignment management and promoting academic integrity. However, it has certain limitations such as the manual selection of drive folders for submissions and the absence of topic validation for assignments, which could be improved to enhance user convenience and ensure valid topic submissions. With its current features and systematic methodology, the bot provides a valuable tool for instructors and students in optimizing assignment processes and maintaining academic integrity.

9. Future Enhancements:

To further improve the project, the following enhancements can be considered:

- **Topic Validation:** Enhance the bot's functionality by incorporating a topic validation mechanism for the submitted assignments. Implement a module that checks whether the topic of the submitted assignment is valid or not, by integrating with external resources such as online databases, reference materials, or APIs that provide information on valid assignment topics.

- **Automated Drive Folder Association**: Implement a feature that automatically associates the user's Drive folder with the star list, eliminating the need for manual folder selection.

- **Real-time Data Updates**: Enhance the bot to fetch and update data from the Google Sheet at regular intervals throughout the day, ensuring that the information is up-to-date and accessible whenever needed..

- **Interactive Feedback and Grading:** Enhance the bot to provide detailed and interactive feedback to users on their submitted assignments, including suggestions for improvement, highlighting strong points, and even automated grading based on predefined criteria.

By incorporating these enhancements, the bot can offer an even more functionalities.

References

- **UiPath Documentation - UiPath Packages**: Documentation on various UiPath packages used in the Academia Assign Plaga-Bot project, covering activities related to Excel, GSuite, Core, Python, and more. [Link](https://docs.uipath.com/references/activities/packages-activities-index)

- **UiPath Documentation - Excel Package**: Documentation specifically focused on the Excel package within UiPath, explaining activities for interacting with Excel files, reading and writing data, and modifying cells. [Link](https://docs.uipath.com/activities/docs/about-the-excel-activities-pack)

- **UiPath Documentation - GSuite Package**: Documentation on the GSuite package in UiPath, demonstrating automation of tasks in G Suite applications like Google Sheets, Google Drive, and Gmail. [Link](https://docs.uipath.com/activities/docs/about-the-gsuite-activities-pack)

- **UiPath Documentation - Python Activities**: Documentation on Python activities in UiPath, providing guidelines for integrating Python code with UiPath workflows. [Link](https://docs.uipath.com/activities/docs/about-the-python-activities-pack)

- **Python Documentation - tkinter**: Documentation on the tkinter library for creating GUI applications in Python, covering modules, classes, methods, and widgets. [Link](https://docs.python.org/3/library/tkinter.html)

- **UiPath Documentation - Installation**: Comprehensive instructions for installing UiPath Studio, the development environment for building automation workflows. [Link](https://docs.uipath.com/installation-and-upgrade/docs/the-installer-requirements)

- **Google Drive Documentation**: Documentation on Google Drive, covering its features, file management, sharing, and collaboration. [Link](https://support.google.com/drive/)

- **Google Sheets Documentation**: Detailed documentation for Google Sheets, covering spreadsheet creation, formatting, formulas, collaboration, and data analysis. [Link](https://support.google.com/docs/answer/6000292)

- **Microsoft Excel Documentation**: Official documentation for Microsoft Excel, including features, formulas, functions, data manipulation, and automation capabilities. [Link](https://support.microsoft.com/en-us/excel)

- **Drive Explorer Chrome Extension:** Extension for Google Sheets to browse, open, import, and export files directly from Google Drive within the Sheets interface. [Link](https://workspace.google.com/marketplace/app/drive_explorer/346145326219)