Sravya Kuchipudi
Professor Kaisler, Hasanov
CSCI 6917
21 July 2023

Project Report: Building a GUI for Unix Commands Execution on MacOS

The focus of this week's work was to have more progress on the error handling for the project. The additions to the code primarily focus on improving the user experience and providing better feedback to the user when interacting with the GUI application. This includes enhancing the run_command function and the run_button_click function to handle error messages and display them in the GUI output window or as pop up messages. These changes were made to provide a more comprehensive and user-friendly experience for the user running Unix commands through the graphical interface.

The initial addition was input validation for the commands and their arguments. It ensures that the user cannot run a command without selecting a valid command from the dropdown menu and, if required, providing the necessary arguments. The validate_arguments function checks for these conditions and returns a boolean value indicating whether the inputs are valid or not. If the inputs are invalid, an error message is displayed using the dialog. This way, users receive immediate feedback about what they need to correct, improving the user experience by guiding them through the correct input process.

The run_command function was modified to capture both the standard output and standard error streams of the executed command. Previously, only standard output was being captured. By adding the capture_output=True argument to the subprocess.run function and extracting completed_process.stderr, any error messages produced by the executed command can now be retrieved. These changes were made to display error messages in the GUI output window and pop up messages when a command encounters issues, making the application more informative and easier to debug. This is still in the process of being worked on however as the output is not being displayed as desired for more command specififc cases.

The run_button_click function was improved to handle the new error messages obtained from the run_command function. It still validates the user's input and displays error messages, when necessary, but it now also handles potential errors returned by the run_command function. If there is an error message, it was first appended to the output in the GUI output window and tagged with an error tag. The tag ensured that the error message is visually distinguished by displaying it in red text, making it easier for users to identify and understand the problem.

While originally, errors were being inserted directly into the output_text widget using tags for formatting, even with the distinct color, this approach might not have been immediately noticeable to users, especially if the output was lengthy. So instead, the alert was changed to using pop-up error messages, so the errors will be displayed in a separate window, ensuring they catch the user's attention. The tkinter.messagebox module is used to display an error message box. If there's an error during command validation (such as an invalid command or missing arguments), the messagebox.showerror presents the error message in a pop-up window, making it clear to the user that something went wrong and explaining the specific issue. Also, after displaying the error message using messagebox.showerror, there is a return statement to exit the run_button_click function early. This is important because to prevent the function from continuing with the command execution or output display if there's an error, and therefore, the error message is the focus of the user's attention, and they can take appropriate action to correct the input.

From here there is some troubleshooting to be done to improve different aspects of the error handling and consolidate the viewing experience. There are also some minor improvements to be made to

other parts of the project as it is being wrapped up such as a full review and testing for any potential issues.