1. A drop-down menu that has a list of names of items. This list of things is searched from a commands.csv file; the commands.csv file's first column is for the drop-down list and the second column has its relevant commands. If a particular item name from the drop-down is selected the executable command against it in the second column of commands.csv would also get and pasted into a new text box labeled "my command".
2. This terminal window. This terminal window shows all the behavior dynamically. The terminal should also have a scroll bar. This terminal window executes the command in the text box labeled "my command " when clicked on a button called "Analyze".
3. A button "Analyze" when clicked executes the selected command on the terminal window and a dynamic timer starts to run while the terminal is executing. This timer keeps running until the terminal outputs the results.
4. A button that allows clearing everything and makes the GUI ready for another execution.
5. A save results button. This button allows saving the results in a .csv file. The .csv file structure would be the first column named "Smart Contract Name" which stores the relevant name from the drop-down menu, and the second column named "Command", which stores the relevant executed command (get-in step. 1), and the third column is named "Results", which stores the complete terminal output in this.
6. The terminal is clickable, which means adding a feature that allows a user to interact with the terminal by clicking on it using a mouse or other pointing device. When the user clicks on the terminal after a command has been executed, it should open a Python file. The name of the Python file that must be opened will depend on the selection from a drop-down menu. For instance, if the user selects "ShowDate" from the drop-down menu, clicking on the terminal will search for a file named "ShowDate.py" in the project directory and open it.