**Final Project Report**

1. Project – Text Processing Tool
2. Reasoning:
   1. I selected this project because as a beginner level project, this seemed like a great way to get into programming in which it requires skill but not a lot of advanced working.
   2. In this project, I will learn how to build a program that processes text files. This includes stuff like word count, line count, and pattern matching through regular expressions. In this I will integrate Unix tools like grep, awk, and sed.
   3. Processing large amounts of text data efficiently and extracting meaningful insights is a significant challenge. Manual analysis is time-consuming and struggles with volume, hindering productivity and the ability to leverage valuable textual information effectively
   4. A Text Processing Tool will offer automated functionalities like text extraction, sentiment analysis, and topic modeling. Its user-friendly interface and customizable options will enable efficient analysis and extraction of key information from diverse text sources.

**Update:**

**Week 1**: The project focused on interpreting basic user commands. The action involved automating the parsing of input text to identify commands and arguments. The result was a system capable of taking actions based on text commands, such as changing directories.

**Week 2**: The project addressed the automated handling of text stream redirection and command linking. The action was to automate the redirection of input/output streams using symbols like ">", "<". This resulted in the system's ability to manipulate how text data flows between automated processes.

**Week 3**: The focus was on managing and controlling text-based processes automatically. Actions included implementing job control commands and enabling background process execution. The system can now list running jobs, move backgrounded processes to the foreground, and manage process states based on text input.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Week | Situation | Task | Action | Result |
| Week 1 | Need to interpret and execute basic user commands entered as text. | Implementing a command interpreter to process alphanumeric character sequences (like "fort", "pwd", "cd"). | Automating the parsing of the input text to identify the command and any arguments. This involves analyzing the character encoding and structure of the command string to trigger corresponding system actions without manual intervention. | The system can automatically take actions (e.g., change directory, print working directory) based on the text commands entered, demonstrating automated manipulation of the "abstraction layer" of the text representing the user's intent. |
| Week 2 | Require automated handling of text stream redirection and connecting commands. | Implementing features to manipulate the input and output of commands using symbols like " | ", "<", and ">". | Automating the process of redirecting the standard input/output streams of processes based on the presence and interpretation of these text symbols. This involves manipulating how text data flows between different automated processes. |
| Week 3 | Need to manage and control text-based processes automatically. | Implementing job control commands (like "jobs", "fg", "bg") and the ability to run processes in the background using "&". | Automating the interpretation of text commands related to process lifecycle (listing, foregrounding, backgrounding). This requires processing the text input to manage the state and execution of other automated processes represented and controlled through text commands. | The system can automatically list running jobs based on the "jobs" text command, move backgrounded processes to the foreground based on "fg", etc., demonstrating automated manipulation of process states triggered by specific text input. |