Non-functional Requirements

I. Product Requirements
   A. Usability Requirements
      The system should be easy to use and may be run through the command line.
   B. Efficiency Requirements
      1. Performance Requirements
         The system should run quickly (no more than 1 minute per run).
      2. Space Requirements
         The system should have minimal impact on memory when run.
         Clever use of memory management will be fairly easy.
   C. Dependability Requirements
      The system should perform error checking for user input/config file parameters and shouldn't crash/error out when passed assignment submissions.
   D. Security Requirements
      We do not expect security to be an issue since the system will only be run locally.

II. Organizational Requirements
   A. Environmental/Operational Requirements
      The system should be able to run on any modern machine (Windows,Mac,Linux)
   B. Development Requirements
      The system should be complete by the end of the semester. If issues arise, the team will consult with Dr. Gurung in order to adjust requirements.

III. External requirements
   A. Ethical Requirements
      The system should produce unbiased results, irrespective of which student's assignment is being graded.
   B. Regulatory Requirements
      N/A
   C. Legislative Requirements
      N/A
   D. Accounting Requirements
      N/A
   E. Safety/Security Requirements
      N/A

Functional Requirements

Terminal

- Request user input for assignment folder path
- Request user input for output path
- Request user input for text file with customized grading criteria
- Runs the program
- Notifies user upon successful/failed completion
- Java logic
    - Parsing zipped file folder (ZFF)
        - Given path to ZFF
        - Loops through ZFF and:
            - Program unzips a zipped folder
            - Hands content to a grader function
            - Receives some confirmation from grader function?
            - Repeat
    - Criteria
        - Comments - Check for comments and headings
        - Function declarations/identifier names
        - Fork bomb checking
        - Compilation success
        - # of lines of code
        - Redundancies (i.e. if x == True)
        - Correctness
            - Complete assignments provided
            - Write different possible test cases
            - Check if output from the assignments graded matches the test cases.