04834580 Software Engineering (Honor Track) 2024-25 Assessment Tasks & Marking Criteria

Sergey Mechtaev

Assessment Scheme

50% project (group)

- **30%** marked based on automated tests
- **20%** marked based on peer-review

20% 2-page report (individual)

marked by teaching staff

30% 2-page peer review (individual)

marked by teaching staff

Deadlines:

24 February 2025, 4pm	Group selection
13 May 2025, 4pm	Code submission
3 June 2025, 4pm	Code review & report submission

Group Project

Given a legacy UNIX shell that provides the following (incomplete) functionality:

Shell Features: calling applications, quoting, semicolon operator, globbing

Supported Applications: `cd`, `pwd`, `ls`, `cat`, `echo`, `head`, `tail`, `grep`

the goal of this project is to refactor the implementation to incorporate software design principles studied in this course, preserve the existing functionality (while fixing potential bugs), and implement the following new features:

Enhanced Shell Features: pipe operator, redirection, command substitution

Additional Applications: `find`, `sort`, `uniq`, `cut`, `wc`, unsafe versions of all applications

Detailed specification of the above features is given in "README.md".

You are expected to refactor the codebase to improve maintainability, readability, and adherence to software design principles, write comprehensive unit tests to ensure the correctness of the implementation (95% branch coverage), analyze the project using a static analyzer to identify and resolve potential issues (no high-severity violations), maintain consistent code style throughout the project, add necessary inline comments and docstrings.

The code will be marked according to the following criteria:

Component	Weight	Marking Method
Functionality	High	System test execution (automatic)
Code coverage	Medium	Unit test execution (automatic)
Static analysis violations	Low	Static analysis execution (automatic)
Design	High	Via peer review
Code quality	Medium	Via peer review
Error handling	Low	Via peer review

Only peer reviews that meet or exceed a specific quality threshold will be used for marking.

The submission must contain the complete source code of the shell. The source code must be anonymized for peer review, meaning it should not reveal your identity. The submitted package will be tested by executing Docker commands specified in "README.md".

Individual Report

Your individual report must contain two pages:

- One page containing one or two mermaid diagrams explaining your design, without or with minimal text annotation.
- One page discussing your individual extra contributions (if no, leave empty)

An individual extra contribution refers to a coursework effort made independently (or with minimal assistance, provided you retain over 50% ownership), which extends beyond the taught materials and specified coursework requirements. Examples of such contributions are:

- Parser generators (ANTRL, lark) or parser combinators for parsing shell commands.
- Extra shell features such as variables, syntax highlighting, or auto-complete.
- Advanced testing methods such as fuzzing/test generation (e.g. EvoSuite), symbolic execution (e.g. CrossHair), mutation testing (PIT), property-based testing (Hypothesis).
- Advanced collaborative or DevOps tools.
- Advanced design patterns.

An individual extra contribution will be recognized only if its use is sufficiently explained and logically motivated.

The report will be marked according to the following criteria:

Component	Weight	Marking Method
Individual extra contributions	High	Subjective manual assessment
Design clarity	Low	Subjective manual assessment

The report must be submitted in PDF format, using a standard font like Times New Roman or Arial in size 11 or 12 for readability. Margins should be set to 1 inch on all sides to ensure adequate white space, and single-spacing is acceptable as long as paragraphs are clearly separated by a blank line or indentation for clarity and structure.

Peer Review

You will be given two shell implementations submitted by other students. Your individual peer review must contain two pages, exactly one per review. Each review must contain three sections:

Design: score + discussion

Code Quality: score + discussion

Error Handling: score + discussion

where a score is a natural number from 1 (low) to 5 (high).

The peer review will be marked according to the following criteria:

Component	Weight	Marking Method
Argumentation quality	High	Subjective manual assessment
Alignment with studied materials	Low	Subjective manual assessment
Thoroughness	Low	Subjective manual assessment

The formatting requirements for peer review are the same as for the individual report.

General Guidance

- The outcomes, findings, and discussions related to your team project must remain strictly confidential. Sharing or discussing these with members of other teams is prohibited.
- Both the individual report and peer review must be completed independently. Collaboration, discussion, or sharing of these materials with other students——including your teammates—is not permitted.
- If you encounter obstacles that hinder your progress in the project——such as team conflicts or uncooperative teammates——it is essential to notify the teaching staff promptly.
- If you believe there has been an error in the assessment of your work, you may request a re-evaluation of your marks by the teaching staff. However, please note that this process does not extend to disputing the marker's academic judgment or the grading criteria applied.
- The use of generative AI tools like DeepSeek is permitted and encouraged. However, directly using unmodified outputs from large language models (LLMs) is unlikely to produce high-quality results.
- Late submissions, incomplete work, or deviations from the specified format will incur penalties.