
 UMS UNIVERSITI MALAYSIA SABAH	 Faculty of Computing & Informatics
COURSE CODE: KK45103	PROGRAMME: SOFTWARE ENGINEERING
COURSE NAME: SOFTWARE EVOLUTION MANAGEMENT	ASSESSMENT TYPE: GROUP PROJECT (4 person per group)
SEMESTER: 1, SESSION 2024/2025	MARKS: 30%
LECTURER: Ts DR NORAZLINA KHAMIS	CLO3: To formulate the maintenance framework by applying appropriate maintenance techniques and tools (A4, PLO5)

1. Objective:

The objective of this group assignment is to apply the concepts of software evolution and maintenance to a legacy software system. The assignment will involve analyzing, reverse engineering, refactoring for reusability, improving maintainability, and employing maintenance tools to evaluate the quality of the system.

2. Instructions:

Your group will be assigned a legacy source code (or you can choose an open-source project with approval). The project will consist of tasks related to reverse engineering, reusability, configuration management, and tools for assessing code quality and maintainability. The tasks below are to be completed and submitted as a group:

Task 1: Reverse Engineering (6 Marks)

Goal: Gain an understanding of the legacy source code and produce high-level documentation to assist in program comprehension.

Instructions:

- Analyze the source code and extract a high-level architecture diagram of the system (2 Marks).
- Provide a brief explanation of key components and their interactions (2 Marks).
- Document the data flow or control flow of the system using diagrams (2 Marks).

Deliverables:

- High-level architecture diagram (UML or similar)
- Data/control flow diagram
- Explanation of the system components

Task 2: Refactoring for Reusability (6 Marks)

Goal: Improve the source code by refactoring to enhance reusability and modularity.

Instructions:

- Identify code smells or redundant code (2 Marks).
- Refactor key parts of the code to promote reusability (4 Marks).
- This may involve breaking large functions into smaller, reusable functions, eliminating code duplication, or modularizing the code.

Deliverables:

- List of identified code smells and issues
- Refactored code with a brief explanation of changes

Task 3: Configuration Management and Version Control (6 Marks)

Goal: Set up version control for the project using Git and demonstrate effective configuration management practices.

Instructions:

- Set up a Git repository and implement branching strategies for your team (2 Marks).
- Demonstrate the use of version control features such as branching, merging, and pull requests during your project (4 Marks).
- Show how you manage code changes, resolve conflicts, and track the history of your changes.

Deliverables:

- GitHub (or similar) repository link
- Description of the branching strategy
- Screenshot or log of pull requests and merges

Task 4: Measuring and Improving Maintainability (6 Marks)

Goal: Evaluate the maintainability of the system and refactor it to improve maintainability.

Instructions:

- Use static analysis tools (e.g., SonarQube, ESLint) to measure the maintainability of the system (2 Marks).
- Refactor code to address maintainability issues and improve code readability and structure (4 Marks).
- This might involve improving documentation, adding comments, or simplifying complex code structures.

Deliverables:

- Tool output/report showing maintainability metrics before and after refactoring
- Explanation of changes made to improve maintainability

Task 5: Testing and Automation (6 Marks)

Goal: Implement a set of regression tests and integrate them into a CI/CD pipeline.

Instructions:

- Write and automate tests for key components of the system (3 Marks).
- Set up a basic CI/CD pipeline (e.g., using GitHub Actions, Jenkins) to automate the testing process (3 Marks).

Deliverables:

- Test scripts (unit or integration tests)
- CI/CD pipeline setup and description
- Output showing successful automated testing runs

Submission Guidelines:

1. Submit all code via a shared GitHub repository (or another version control platform) and provide the link to access
2. Submit a report (HARDCOPY) summarizing the work done for each task, including:
 - a. Diagrams, screenshots, and code excerpts where necessary
 - b. Links to the repository and CI/CD results

Each report must in proper format (COVER PAGE (names in alphabetical order), assignment submission declaration form. Use appropriate formatting, headings, and references where necessary.

Submission WEEK 12.

3. Each group will also present their findings and demonstrate their results in class.

Important Notes:

1. Collaboration among group members is essential; everyone should contribute to each task.
2. Clearly indicate each group member's contribution in the final report.
3. Late submissions will incur a penalty unless prior permission is granted due to exceptional circumstances.

Assignment Grading Rubric:

Task	Marks	Criteria
Task 1: Reverse Engineering	6 Marks	<ul style="list-style-type: none">• Accurate architecture diagram (2)• Explanation of components (2)• Data/control flow diagram (2)
Task 2: Refactoring for Reusability	6 Marks	<ul style="list-style-type: none">• Identification of code smells (2)• Quality of refactoring for reusability (4)
Task 3: Configuration Management	6 Marks	<ul style="list-style-type: none">• Git repo setup and branching strategy (2)• Proper use of version control (4)
Task 4: Maintainability	6 Marks	<ul style="list-style-type: none">• Tool output and analysis (2)• Improvements to maintainability through refactoring (4)
Task 5: Testing & Automation	6 Marks	<ul style="list-style-type: none">• Quality of test scripts (3)• CI/CD pipeline implementation (3)

