DUY TÂN UNIVERSITY
International School

GROUP PROJECT 3

Version 1.0

Date: 23/4/2025

SOFTWARE CONFIGURATION MANAGEMENT

Created by GROUP_2 CLASS_CMU-CS 246 PIS

Mentor: Nguyen Dang Quang Huy

Team Member:

Anh, Nguyen Thi Thu
29209051377

Hao, Pham Tan
29219020823

Thang, Van Vo
28210248363

Hau, Mai Phuc
29211160520

Trung, Tam Nguyen
29212736855

Quyet, Dang Thanh
29211143495

Table of Contents

1. Project Introduction

Project Overview

The Caloulator project aims to develop a versatile calculator application with both basic and advanced functionalities to meet diverse user needs. The primary goal is to provide accurate calculations while enhancing user experience through new features. Recently added features include trigonometric functions (sin, cos, tan, cot), the ability to change the calculator's color theme for better personalization, and other advanced mathematical operations such as logarithmic, exponential, and factorial calculations.

Technologies Used

The project leverages the following technologies:

Java: The core programming language used for developing the calculator's logic and user interface.

GitHub: Used for version control and collaborative development, enabling team members to manage code repositories effectively.

NetBeans: The primary Integrated Development Environment (IDE) for coding, debugging, and testing the application.

TortoiseSVN: A Subversion client used for additional version control, facilitating code integration and backup.

2. Team Workflow

Work Organization

Our team organized the project through a combination of in-person study group sessions and online collaboration via Zalo. We conducted regular face-to-face meetings to discuss progress, brainstorm solutions, and ensure alignment, while Zalo was used for daily communication, task updates, and quick issue resolution among team members.

Task Assignment

Tasks were assigned to team members based on their skills and roles:

Nguyễn Thị Thu Ánh: Redesigned the user interface, focusing on improving the visual layout and user experience.

Phạm Tấn Hào: Developed new features, including advanced mathematical functions and other enhancements.

Mai Phúc Hậu: Worked on writing the project report, documenting the team’s progress and outcomes.

Đặng Thanh Quyết: Coded additional new features alongside Phạm Tấn Hào, contributing to the calculator's functionality.

Võ Văn Thắng: Served as the tester, ensuring the application worked as intended and identifying bugs.

Nguyễn Tam Trung: Collaborated with Mai Phúc Hậu on writing the project report, ensuring all details were covered.

Completed Work and Sub-Teams

The team successfully completed key features, including the color-changing interface, advanced mathematical functions (such as sin, cos, tan, cot, logarithmic, and exponential calculations), and the integration of a calculation history feature. We divided the team into sub-teams based on functionality: the UI team (led by Nguyễn Thị Thu Ánh), the logic team (Phạm Tấn Hào and Đặng Thanh Quyết), and the integration/reporting team (Mai Phúc Hậu and Nguyễn Tam Trung, with testing support from Võ Văn Thắng). Each sub-team worked on separate branches in GitHub to avoid conflicts, regularly merging their contributions into the main branch after testing.

3. SCM Operations

SCM Operations Performed

Our team utilized GitHub and TortoiseSVN for source code management (SCM). The primary SCM operations included:

Checkout: Team members checked out the latest version of the project repository from GitHub to their local machines using Git commands (git clone for the initial setup and git pull for updates). TortoiseSVN was also used to check out specific project files for backup purposes.

Commit: Each member committed their changes frequently to their respective branches on GitHub using git commit, ensuring that updates were saved with descriptive messages (e.g., "Added sin/cos functions" or "Updated UI color theme").

Push: After committing, changes were pushed to the remote repository on GitHub using git push, making them available for collaboration. TortoiseSVN was used to push backup copies to a secondary SVN repository.

Merge: Once features were completed and tested, branches were merged into the main branch using git merge on GitHub. Pull requests were created to review changes before merging, ensuring code quality.

Evidence of Commits and Pushes

To demonstrate the contributions of each team member, we captured GitHub commit logs and TortoiseSVN history. For example:

Nguyễn Thị Thu Ánh’s commits show updates to the UI files (e.g., "Redesigned color theme - 10 commits").

Phạm Tấn Hào and Đặng Thanh Quyết’s logs include additions to the logic for trigonometric functions (e.g., "Implemented sin/cos/tan - 15 commits").

Screenshots of the GitHub commit history and TortoiseSVN logs were taken, showing timestamps, commit messages, and the respective branches each member worked on.

Conflict Prevention and Resolution

To avoid conflicts while working simultaneously:

Each sub-team worked on separate branches (e.g., ui-branch, logic-branch) on GitHub, ensuring that changes did not overlap.

We communicated regularly via Zalo to coordinate tasks and inform others about ongoing changes, reducing the risk of overwriting code.

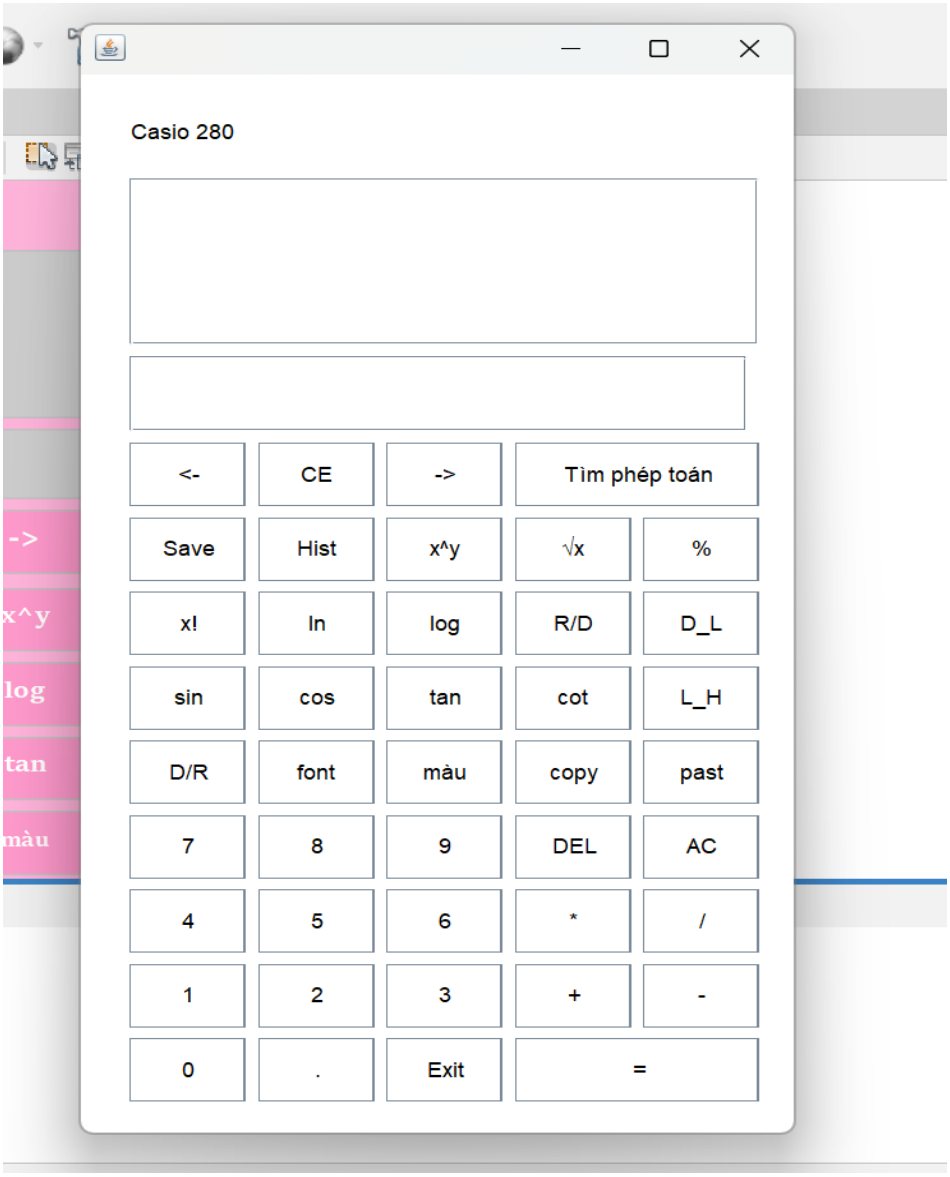
When conflicts occurred, such as when two members modified the same file:

We used GitHub’s conflict resolution tools to manually review and resolve differences. For instance, a conflict in the main calculation file was resolved by merging changes to ensure both the UI updates and new functions worked seamlessly.

TortoiseSVN’s diff tool was also used to compare changes and resolve conflicts in backup files, ensuring consistency across repositories.

4. Document Management

Design documents



Describe

Any changes in the document are archived, making it easy to compare, rollback, or view the edit history.

Documents can be managed in parallel with the source code, helping to synchronize information.

Documents can be linked to specific issues for easy tracking of requests.

Documents are stored publicly or internally in a repository, easily accessible to the entire project team.

Examples of how documents work

Situation:

A software development team needs to update the system installation guide. This document is stored in the docs/ directory of a repository on GitHub.

Steps to update and store documents:

- Clone the repository to your local machine.
- Create a new branch to update the documentation
- Edit the document.
- Commit the changes.
- Push the branch to the repository.
- Create a Pull Request (PR)
- Code review and merge.
- Store change history.

5. Testing and Results

Additional Features

Scientific Operations:

• Factorial (n!).

• Logarithms ($\log_{10}$ x, ln x).

• Trigonometric functions: sin(x), cos(x), tan(x), cot(x).

• Angular unit conversion: Radian ↔.

Calculation History:

• Store the calculation history in a JSON or text file (instead of just saving it in the remember).

• Allows users to re-search for old calculations in history.

• Allows users to delete each line of history.

• Allows users to choose the font and interface color.

• Supports keyboard shortcuts (use the keyboard to enter calculations).

Control Functions:

• Added CE (Clear Entry) button to remove a unique number without changing it calculations.

• Added ← (Backspace) and → (Forward) functions to remove one character in two cursor direction.

Other additional features:

• Copy/Paste the result to the clipboard.

• Supports calculation according to complex expressions such as: "5 + (3 * 2) - √9".

Advanced Error Handling:

• Testing input error scenarios such as calculations exceeding limits, or Invalid math (e.g., divide by 0, square root of a negative number).

• Added a detailed error message when entering incorrect data (e.g., "5++2", "√-9").

Test Case Materials

Module	**Common Logarithm FeatureComm**				
Code	Logarit-X				
Check requirement					
Tester	Vo Van Thang				
Status	**TOTAL**	**PASS**	**FAIL**	**Not Implemented**	**SKIPPED**
	4	4	0	0	0

STT	Test case ID	Requirements ID	Test case decription	Pre - Condition	Step	Data	Excepted Result	Test Date	Actual Result	Assign to
1	Logarit-X - 1	Basic 1	Common Logarithm FeatureCommon	Calculator app opened, performed calculations	1. Enter '5' 2. Press 'x!' 3. Press '='	5!=120	Factorial	22-Thg4-25	PASS	Vo Van Thang
2	Logarit-X - 2	Basic 2	Common Logarithm FeatureCommon	Calculator app opened, performed calculations	1. Enter '0' 2. Press 'x!' 3. Press '='	0!=1	Factorial	22-Thg4-25	PASS	Vo Van Thang
3	Logarit-X - 3	Basic 3	Common Logarithm FeatureCommon	Calculator app opened, performed calculations	1. Enter '-5' 2. Press 'x!' 3. Press '='	-5!= -120	Factorial	22-Thg4-25	PASS	Vo Van Thang
4	Logarit-X - 4	Basic 4	Common Logarithm FeatureCommon	Calculator app opened, performed calculations	1. Enter '5' 2. Press '*' 3. Enter '5' 4. Press 'x!' 5. Press '='	5*5!=600	Factorial	22-Thg4-25	PASS	Vo Van Thang

6. Problems encountered and solutions

Problems encountered

Technical Issues

- Unsynchronized environment
- Lack of CI/CD configuration
- Poor dependency management
- Sensitive data pushed to the repo
- Documentation is lacking or outdated

Source code conflicts

Merge conflict

Rebase conflict

Delete untracked files

Do not comply with naming/format conventions

Do not update the branch before working

Difficulties in the process of teamwork

Lack of communication

Lack of consistency in work processes

Delays in reviewing pull requests

Lack of a repository manager

Ineffective use of issue trackers

Solution

Solutions to technical problems

Environment not synchronized Use Docker or .env, requirements.txt, package.json, pyproject.toml files to configure a unified environment.

Lack of CI/CD Set up a CI/CD pipeline (GitHub Actions, GitLab CI) to test, build and deploy automatically every time you push code.

Unstable dependencies Lock library versions with files like package-lock.json, poetry.lock, Pipfile.lock, etc.

Sensitive data leaks Use .gitignore, tools like git-secrets and store environment variables in a secure environment like GitHub Secrets.

Lack of or outdated documentation Assign tasks to update documentation after each new feature. Apply "Documentation as Code" - documentation is written as code, stored in the repo.

Solution to handle source code conflicts

Merge conflict Regularly pull from main and use git merge or git rebase to update before pushing.

Rebase conflict Only rebase when you are sure how to resolve conflicts. Prioritize merge if the team is not familiar with rebase.

Delete the wrong file Use pull requests (PR) so that all changes are thoroughly reviewed before merging.

Inconsistent code formatting Use automatic formatters like Prettier, Black, eslint, and configure pre-commit hooks.

Do not update branches Make a habit of running git pull origin main before starting a new task.

Solution to teamwork difficulties

Lack of communication Use tools like Slack, Discord, or GitHub Discussions to update work status. Hold short meetings every day (daily standup).

Lack of process consistency Build a clear workflow: how to name branches, how to write commit messages, review process. Save to CONTRIBUTING.md.

Slow PR review Assign someone to review. Set a maximum review time (e.g. within 24 hours).

No repository manager Assign someone to manage the repository or rotate roles (scrum master, maintainer).

No issue tracker Use GitHub Issues, GitLab Issues, or Jira, assign tasks to each person, link PR to issue for easy tracking.