**report DOCUMENT**

Contents

[**I. Project introdution** 2](#_Toc195866783)

[1. Purpose 2](#_Toc195866784)

[2. Scope 2](#_Toc195866785)

[3. Intended Audience 2](#_Toc195866786)

[**II. Teamwork process:** 3](#_Toc195866787)

[4. Completed work and how the team collaborates via GitHub 4](#_Toc195866788)

[**a) Completed work** 4](#_Toc195866789)

[**b) Teamwork** 4](#_Toc195866790)

[**III. SCM OPERATIONS** 4](#_Toc195866791)

[1. Detailed description of SCM operations 4](#_Toc195866792)

[2. Commit, push of each member 4](#_Toc195866793)

[3. Conflict resolution 5](#_Toc195866794)

[**IV. Document management:** 5](#_Toc195866795)

[1. Description of relevant documents 5](#_Toc195866796)

[2. Example 6](#_Toc195866797)

# I. Project introdution

## 1. Purpose

- This document describes the software requirements for the Calculator application - a graphical calculator application built in Java language. The purpose of the system is to provide users with an accurate, stable, easy-to-use calculation tool, simulating the interface and functions of popular handheld calculators (such as Casio fx-580VNX). The application supports both basic and advanced calculations, along with the feature of saving the history of calculations during the working session.

- This document is used as a basis for the process of analyzing, designing, implementing and testing the software.

## 2. Scope

- The Calculator application is developed as a desktop software, with a graphical user interface (GUI) using Java Swing. The application supports:

- Perform basic arithmetic operations: addition, subtraction, multiplication, division.

- Advanced operations: percentage, exponent, square root, change of sign.

- Display results in standard decimal format and support rounding, scientific notation if needed.

- Save calculation history in current session (not saved permanently).

- Interface simulates real computer keyboard, easy to get used to by users.

- The project does not support full expression input like expression calculator (e.g., 2 + 3 × (5 - 1)), but processes each calculation according to the user's keystroke flow (step by step, like on Casio calculator).

## 3. Intended Audience

- End users: Students, students, or anyone who needs to use a simple scientific calculator.

- Programmers: People who want to extend features or integrate this application into a larger system.

- Testing team (QA): Use this document to build appropriate test plans and test cases.

- Project manager (PM): Monitors the scope, goals, and progress of the system.

**4. Scope of use**

- The system consists of 3 main components:

- Logic layer: Manages mathematical operations, number conversion, rounding, and result formatting (CalculatorLogicNew.java).

- User interface: Displays the screen, operation buttons, and coordinates the calculation flow (CalculatorPanel.java, MainUI.java).

- Calculation history: Remembers and displays calculations performed in the session (HistoryPanel.java).

The system operates completely offline, does not require the internet or depend on external systems.

# II. Teamwork process:

**1. Work organization**

- The project is implemented by a team of 5 members, organized according to a specialized functional assignment model, each person takes on a specific role in the software development chain. The team uses a flexible working model combined with the GitHub tool for remote coordination.

- Each member takes on tasks that match their personal strengths and is primarily responsible within the assigned scope. Online team meetings are held periodically to update progress, share difficulties and ensure consistency in the development process.

**2. Assign work among members**

|  |  |  |
| --- | --- | --- |
| Member | Roles | Main task |
| Nguyen Trong Khoi | Project management, report writing, Code the functions | Manage progress, assign work, coordinate teams and write final reports. |
| Nguyen Nam Hung | Code the functions, Writing SRS document | Developed the computer interface using Java Swing and handled all the computational logic. |
| Doan Thi Ngoc Han | Writing SRS document | Write technical specifications, add content to SRS and review output documents. |
| Do Van Hieu | Tester | Write test cases, test functionality, detect logic and UI errors. |
| Nguyen Tran Duc Khai | Writing SRS document | Write technical specifications, add content to SRS and review output documents. |

## 4. Completed work and how the team collaborates via GitHub

### a) Completed work

### - Build a computer interface using Java Swing.

- Develop precise calculation logic, handle many types of calculations and format results intelligently.

- Test the entire application through manual test cases.

- Complete SRS documents that are complete, clear, and technically correct.

- Write a comprehensive project report, detailing the process and results of teamwork.

### b) Teamwork

- GitHub is the main platform for sharing and managing source code. The team uses separate branches to develop features, then merges them back to the main branch via pull requests.

- Google Meet and Messenger are used for quick communication and online team meetings.

- Google Docs is used as a platform for writing and editing SRS documents in real time, ensuring consistency and synchronization between Han and Khai.

- GitHub Issues is used to manage bugs discovered during testing, helping the team track and handle errors quickly.

# III. SCM OPERATIONS

## 1. Detailed description of SCM operations

## 2. Commit, push of each member

## 3. Conflict resolution

Each member works on a separate branch:

* Each person is responsible for a Git branch corresponding to the assigned task (e.g. ui-layout, logic-processing, srs-writing, testing), thereby minimizing editing the same file at the same time.

Clearly separate tasks:

* Logic, interface, documentation and testing are clearly separated, avoiding touching the common code area. The project structure is designed from the beginning to ensure high modularity.

Regularly update the main branch (git pull):

* Each member is reminded to perform git pull origin main before starting work each day to synchronize the latest source code, avoiding editing based on the old version.

Clear pull request process:

* Each change is created as a pull request, and only merged into main after at least 1 other member reviews and approves. This helps detect conflicts or logic errors early.

# IV. Document management:

## 1. Description of relevant documents

Software Requirements Specification (SRS) document:

* Detailed description of software requirements, including functional, non-functional, overall design, work assignments, and team coordination process.

Interface design document:

* Including user interface structure diagram, function button layout, and user experience orientation.

Test Case:

* List of manually written test scenarios for each function, with status (Pass/Fail), expected input and output.

## 2. Example

# V. Testing and results:

TEST CASE: [Test Case version 2](../../../Downloads/TEST_CASE_VERSION_2.xlsx)

# vi. Problems encountered and solutions:

### a) Technical issues

Processing high-precision calculations (Decimal):

* Problem: When performing division, square root or calculations with long decimal results, the display and calculation are prone to errors or data overflow.
* Solution: The team uses the BigDecimal class instead of float/double to ensure accuracy, and limits the number of digits displayed in the output using a custom format to keep the interface neat.

Inconsistent interface design using Java Swing:

* Problem: Initially, the interface was not uniform in color, layout and button size, causing confusion.
* Solution: The team clearly divided the interface components into each Panel, then unified the font, background color and grid layout to unify the UI.

### b) Source code conflicts (Merge Conflict)

Problem:

* Conflicts occur when 2 members edit the same Java file (especially CalculatorPanel.java) and push to the repository at the same time.

Solution:

* Each member works on a separate branch, avoiding editing directly on main.
* Regularly update from main using git pull before working.
* When conflicts occur, the person who creates the pull request will use the conflict resolution tool (VSCode/GitHub) to manually handle and retest before merging.

### c) Team communication problem

Problem:

* There are times when the team does not clearly agree on the content of the SRS document, leading to duplicate writing or missing important items.

Solution:

* The team organizes a quick meeting to specifically divide the SRS content for each person (Han and Khai), and at the same time uses shared Google Docs to edit and track versions in real time, avoiding overwriting.

### d) Manual testing problem

Problem:

* Because an automated testing framework has not been used, all test cases are performed manually, so it is easy to miss edge cases.

Solution:

* Tester (Hieu) created an Excel test case table, which clearly listed the input, expected output, status, and notes to ensure comprehensive testing. Important cases were tested repeatedly before the deadline.

# Vvi. Video demo

## 1. Demo

## 2. New features

### a) Scientific calculation

Support new operations:

* Factory: n!
* Logarithm: log(x), ln(x)
* Trigonometric functions: sin(x), cos(x), tan(x), cot(x)
* Angle conversion: Degree ↔ Radian

How to use:

* Users select the corresponding operation icon on the interface or use the keyboard to enter quickly.

### b) Advanced calculation history

New feature:

* Save history to .json file so it is not lost when the application is closed.
* Allow searching for old calculations.
* Allow deleting each line in the history.

How to use:

* Access the "History" tab, there is a search bar and buttons to delete each line.

### c) User-friendly interface

Customize the interface:

* Add Dark Mode / Light Mode.
* Allow changing fonts and colors.

Support shortcut keys:

* You can use the keyboard to enter calculations instead of clicking the mouse.

How to use:

* The settings are in the "Interface Settings" menu, users can select directly.

### d) Utility control functions

CE button: Delete the number being entered without affecting the entire calculation.

← / → button: Delete or move the cursor in the expression being entered.

Copy/Paste: Support copying the result to the clipboard or pasting the expression from another location.

### e) Processing complex expressions

Support entering and processing expressions in the form:

* "5 + (3 \* 2) - √9"

Ability to check and report detailed errors such as:

* Divide by 0

Incorrect syntax input: "5++2", "√-9"

Clear error messages will appear on the screen to help users fix quickly.