Calculation Nation: A Collaborative

Python-Based Project

Jack Gilbert, Vaishnavi Mandage, Hema Deepika Thanigai Arasu, Matthew Thibault

Team Number 05

CS504-Software Engineering, Master of Science in Computer Science, School of Technology and Computing, Seattle City University)

[gilbertjack@cityuniversity.edu](mailto:gilbertjack@cityuniversity.edu)

[mandagevaishnavi@cityuniversity.edu](mailto:mandagevaishnavi@cityuniversity.edu) [thanigaiarasuhemade@cityuniversity.edu](mailto:thanigaiarasuhemade@cityuniversity.edu) [thibaultmatthew@cityuniversity.edu](mailto:thibaultmatthew@cityuniversity.edu)

**Abstract**

Developing a simple calculator application involves designing and implementing the user interface, defining the arithmetic operations, and handling user input and output. The user interface should be intuitive and easy to use, with buttons for each digit and operation. The arithmetic operations should include addition, subtraction, multiplication, and division. User input should be validated to ensure it is a valid numerical expression, and the result of the calculation should be displayed in a clear and readable format. The application should also handle errors and edge cases, such as division by zero and input of non-numerical characters. To implement this application, one can use any programming language with a graphical user interface toolkit or web development framework.

**Keywords:** Calculator application, user interface, arithmetic operations, input validation, error handling, edge cases.

1. **INTRODUCTION**

A calculator refers to a device used in the processing of various kinds of mathematical calculations. There are diverse types of calculators depending on their use with some calculators being designed for use in a specific context while others being designed for basic mathematical calculations. A standard handheld calculator comes with a keypad for entering numbers and operations and a screen that displays entered numbers and results. Modern calculators can execute calculus, statistical, trigonometric, geometric, and algebraic operations (Ipeayeda et al., 2015). These calculators rely on electric power, which can be solar cells, batteries, or standard electric current. Modern calculators come with additional features to those of a standard calculator. While the features of a modern calculator may differ across manufacturers and the models employed, a scientific calculator will often include trigonometric functions, floating point arithmetic, easy access to constants, additional roots to the square root, scientific notation, and exponential functions.

Technological advancements have led to the development of software-based calculators. This kind of calculator was first developed using Visual Studio, which refers to an integrated development environment (IDE) that provides a single interface for different languages (Isizoh et al., 2012). This environment combines the development of software into a single interface. This study's purpose is to develop a software-based calculator that will perform basic arithmetic operations, including addition, subtraction, multiplication, and division. The goal is to design an intuitive and easy to use interface with buttons for every digit and operation.

1. **LITERATURE REVIEW**

Calculators exist in different forms. One of them is the traditional, basic calculator, which is a simple four-function calculating device. A basic calculator executes addition, subtraction, multiplication, and division operations. The second is the scientific calculator, which is capable of writing and storing programs, such as the quadratic formula (Rojas-Sola et al., 2023). Another important calculator type is the graphing calculator, which refers to a handheld, battery-powered device capable of plotting graphs, giving numerical answers to equations, and performing a wide range of operations, such as statistics, matrix, geometry, and algebra (Kandemir & Demirbag Keskin, 2019). It falls under the category of scientific calculators but comes with additional features for producing tables and plots. The graphing calculator is commonly used as an exploratory, confirmatory, graphing, multidimensional, and problem-solving tool (Parrot & Leong, 2018)l. It plays a crucial role in the development of mathematical concepts. Unlike basic calculators used by anyone to solve common equations, scientific calculators are commonly used by scientists and mathematicians.

Technological developments have led to a shift from the traditional handheld calculator to the design of software-based calculators. Unlike a handheld calculator, a software calculator is designed and implemented as a computer program. Krishnan et al. (2018) designed an energy calculator in the form of a mobile application for potential solar measurement. The project was motivated by the need to quickly and effortlessly obtain estimates of the cost saved in the installation of solar. Rahim et al. (2018) designed and developed a mobile application that could be used by students to calculate Inverse Method and Cramer’s Rule. The findings of the study showed that using the mobile-based calculator resulted in an improvement in the performance of students. Sumarni et al. (2020) used Matlab software to create a Matlab-based physics calculator. The results of the study showed that a physics-based Matlab calculator focused on the concept of work and energy would serve as an important physics learning medium. In a different study, Moriwaki et al. (2018) developed a descriptor-calculation software application known as Mordred to enable the calculation of at least 1800 two-and three-dimensional descriptors. Their study led to the conclusion that the good performance and convenience of the software calculator made it a promising choice for the calculation of molecular descriptors.

The two important concepts that should be taken into consideration in the design of a software solution are user interface (UI), and user experience (UX). User interface (UI) is concerned with how a solution appears to the user, along with its visual design (Zhu, et al., 2022). A good user interface design prioritizes the delivery of relevant functionality over the solution’s beauty or elegance. The interface is concerned with the convenience and ease with which a user interacts with the solution (Granić, 2017). A set of principles should inform the design of the user interface. These include visibility, simplicity, structure, and consistency (Ruiza et al., 2021). The important elements that should be used in the design of the user interface include iconography, typography, shapes and placement, color, and animation. On the other hand, user experience (UX) is concerned with how effectively the solution performs its functionality (Sandesara, et al., 2022). The UX elements that should guide the design and development of the solution include goal fulfillment, usability and ergonomics, utility satisfaction and user control, language, and minimalism.

1. **METHODOLOGY**

The development of a software-based calculator that will perform basic arithmetic operations, including addition, subtraction, multiplication, and division involved the use of the experimental research method. Experimentation involves testing the hypotheses of causal relationships between variables (Kandel, 2011). It is the most rigorous approach to understanding relationships between variables. It involves conducting research in an objective and controlled manner to maximize precision to arrive at specific conclusions. Experimentation can be understood as a planned interference with a process while carefully observing how the variables interact. It requires the researcher to deliberately control and manipulate conditions to establish results. Through experimentation, which involves the actual development of a software-based calculator, researchers discover the factors that contribute to the development of an intuitive, easy-to-use software solution. The experimental process used in developing the software-based calculator is described in the next section.

1. **PLANNING AND PREPARATION**

To successfully design and develop a Python calculator app, a 4-person software team should follow a systematic approach. The team begins by gathering the requirements, both functional and non-functional, of the calculator app, considering the target platforms and desired features. Once the requirements are clear, the team proceeds to design the app's architecture, selecting an appropriate pattern such as MVC or MVVM, and defining the main components and their interactions. The user interface design comes next, focusing on creating an intuitive and visually appealing GUI (Graphical User Interface) that adheres to usability principles. If the app requires persistent storage, the team designs the database schema and implements the data access layer.

The actual development phase involves splitting tasks among team members, leveraging their strengths and expertise. Following coding best practices and conventions, the team implements the user interface based on the design, as well as the calculator logic and mathematical operations based on predefined algorithms. Thorough testing is conducted, including unit testing, integration testing, and system testing, to ensure the app's functionality and user experience. Any identified bugs or issues are addressed promptly.

Documentation plays a crucial role in this process. The team documents the design decisions, architectural overview, and key functionalities of the app. User documentation is prepared to guide users on how to use the calculator effectively. Additionally, code documentation with clear comments and function/method descriptions is created. The deployment phase involves packaging the app into an executable or installer, testing the deployment package, and distributing the app through appropriate channels.

Maintenance and updates are essential for long-term success. The team establishes a process for bug fixes, updates, and feature enhancements, while actively seeking and addressing user feedback. Continuous improvement of the codebase is pursued through best practices and code refactoring. Effective collaboration and communication are maintained within the team using project management tools, regular meetings, and code reviews. Regular communication with stakeholders ensures transparency and aligns with their expectations.

By following this comprehensive approach, the software team can design, develop, and deliver a high-quality Python calculator app that meets the requirements of the users and stakeholders.

**5. DEVELOPMENT**

**5a. Source Code**

The deliverables produced for this project include completing all designing and coding required to create the initial prototype. Our calculator is currently deployed by executing the following two code files:

1. calc.py

A screenshot of a computer program

Description automatically generated with medium confidence

A screen shot of a computer

Description automatically generated with medium confidence

A screen shot of a computer

Description automatically generated with medium confidence

A screen shot of a computer

Description automatically generated with medium confidence

2. calc\_math.py

A screen shot of a computer

Description automatically generated with medium confidence

A screen shot of a computer

Description automatically generated with medium confidence

**5b. Source Code Explained**

The first code file contains the implementation of a class called CalcMath, which represents a calculator with various mathematical operations. The second code file implements a graphical user interface (GUI) calculator using the tkinter library and utilizes the CalcMath class for performing the mathematical calculations.

Let us break down the first code file, which defines the CalcMath class:

1. The \_\_init\_\_ method initializes the CalcMath class and sets up a dictionary called operations. This dictionary maps mathematical operators to their corresponding operation names.
2. The calculate method takes an operator, num1, and an optional num2 as input and performs the requested calculation. It first retrieves the operation name from the operations dictionary based on the provided operator. Then, it dynamically evaluates the expression using the eval function by constructing the appropriate function call based on the operation name and the input numbers. If num2 is not provided, it calls the corresponding method with only num1. The method also checks if the result is a float and if it is an integer, it converts it to an integer before returning.
3. The subsequent methods in the class represent specific mathematical operations such as addition, subtraction, multiplication, division, negation, exponentiation, square root, and nth root. Each method performs the corresponding mathematical operation and returns the result.

Moving on to the second code file, which implements the GUI calculator:

1. The Calculator class is defined, and the \_\_init\_\_ method sets up the calculator's initial state. It creates an instance of the CalcMath class (cm) to perform the calculations. It also initializes variables for storing the current number (number) and operator (operator).
2. The class creates a GUI window using tkinter, sets its title and disables resizing.
3. The create\_numeral\_button, create\_single\_operand\_button, create\_two\_operand\_button, create\_equals\_button, and create\_clear\_button methods are responsible for creating the buttons on the calculator's GUI. These buttons are associated with different actions such as entering numerals, performing single operand operations, two operand operations, calculating the result, and clearing the display.
4. The numeral\_helper method is called when a numeral button is pressed. It adds the corresponding digit or decimal point to the calculator's display. If the clear\_input\_on\_next\_numeral flag is set to True, it clears the display before adding the numeral.
5. The single\_operator\_helper method is called when a single operand operator button is pressed. It checks if the current display value can be converted to a float using the is\_float method. If it is a valid float, it sets the operator variable to the corresponding operator and calculates the result using the CalcMath instance (cm) by calling the calculate method. The result is then displayed on the calculator's display.
6. The operator\_helper method is called when a two-operand operator button is pressed. It performs a similar check as in the single\_operator\_helper method to ensure the current display value is a valid float. If so, it stores the value in the number variable and sets the operator variable to the corresponding operator. If the display value is not a valid float, it clears the display and shows an error message. It also sets the clear\_input\_on\_next\_numeral flag to True to clear the display when the next numeral is entered.
7. The is\_float method is a helper function that checks if a given string can be converted to a float by attempting the conversion using a try-except block. It returns True if the conversion.

**6. CONCLUSIONS**

The development of software, such as a software-based calculator, plays a crucial role in finding solutions to the problems and challenges faced by humankind. A software-based calculator is designed and implemented as a computer program. The program should be capable of aiding the solution of mathematical problems. As a software program, its effectiveness is impacted by both its user interface (UI) and user experience (UX). The solution should have all the required functionalities while remaining as usable as possible. This implies that in addition to aiding the determination of answers to arithmetic problems, the user interface should be intuitive and easy to use.

We sought to design and develop a Python calculator app. The design of the application’s architecture was preceded by the collection of functional and non-functional requirements. The development of the solution adhered to coding best practices and conventions. The deployed calculator executes two codes, namely calc.py, which represents a calculator with various mathematical operations, and calc\_math.py, which implements a graphical user interface calculator using the tkinter library, while utilizing the calcMath class in performing mathematical calculations. The resulting calculator can execute arithmetic operations, such as addition, subtraction, multiplication, and division. User input is validated to ensure it is a valid numerical expression, and the result of the calculation are displayed in a clear and readable format.

**7.** **REFERENCES**

Granić, A. (2017). Technology in use: The importance of good interface design. *2017 International Conference on Infocom Technologies and Unmanned Systems (ICTUS'2017)* (pp. 43-49). IEEE.

Ipeayeda, F. W., Binuyo, G. O., & Binuyo, A. O. (2015). Development of a calculator software for determining the dominant eigenvalue and eigenvector of n x n matrices. *Applied Mathematics, 83*, 33271-33274.

Isizoh, A. N., Anazia, A. E., Okide, S. O., & Okwaraoka, C. A. (2012). Software-based scientific calculator using Visual Basic.Net. *International Journal of Engineering Research & Technology (IJERT), 1*(10), 1-5.

Kandemir, M. A., & Demirbag Keskin, P. (2019). Effect of graphing calculator program supported problem solving instruction on mathematical achievement and attitude. *International Journal of Research in Education and Science (IJRES), 5*(1), 203-223.

Krishnan, S. M., Rawat, S., Surender, M., Balakrishna, R., & Anandan, R. (2018). Implementing an energy calculator in a mobile based application for solar potential measurement. *International Journal of Engineering & Technology, 7*(3), 403-406.

Moriwaki, H., Tian, Y.-S., Kawashita, N., & Takagi, T. (2018). Mordred: A molecular descriptor calculator. *Journal of Cheminformatics, 10*(4), 23-45. doi:https://doi.org/10.1186/s13321-018-0258-y

Parrot, M. A., & Leong, K. E. (2018). Impact of using graphing calculator in problem solving. *International Electronic Journal of Mathematics, 13*(3), 139-148.

Rahim, R. H., Baharum, A., & Hijazi, M. H. (2018). Mobile application calculator for inverse method and Cramer’s rule at Polytechnic Kota Kinabalu. *Journal of Fundamental and Applied Sciences, 10*(7), 132-142.

Rojas-Sola, J. I., Río-Cidoncha, G., Ortíz-Marín, R., & Cebolla-Cano, A. (2023). Design and development of a geometric calculator in CATIA. *Symmetry, 15*(547), 1-26. doi:https://doi.org/10.3390/sym15020547

Ruiza, J., Serral, E., & Snoeck, M. (2021). Unifying functional user interface design principles. *International Journal of Human–Computer Interaction , 37*(1), 47-67. doi:https://doi.org/10.1080/10447318.2020.1805876

Sandesara, M., Bodkhe, U., Tanwar, S., Alshehri, M. D., Sharma, R., Neagu, B. C., . . . Raboaca, M. S. (2022). Design and experience of mobile applications: A pilot survey. *Mathematics, 10*(14), 2380-2407. doi:https://doi.org/10.3390/math10142380

Sumarni, R. A., Juliardi, M., Widiyatun, F., Astuti, I. A., Okyranida, I. Y., & Bhakti, Y. B. (2020). MATLAB-based physics calculator: Alternative for learning media for work and energy concept. *International Conference on Mathematics and Science Education (ICMScE) 2020.* 1-7: IOP Publishing. doi:10.1088/1742-6596/1806/1/012022

Zhu, D., Wang, D., Huang, R., Jing, Y., Qiao, L., & Liu, W. (2022). User interface (UI) design and user experience questionnaire (UEQ) evaluation of a to-do list mobile application to support the day-to-day life of older adults. *Healthcare, 10*(10), 2068-2077. doi:https://doi.org/10.3390/healthcare10102068