Project Report: Student Utility Tools (SUT)

1. Project Title

Student Utility Tools (SUT)

2. Introduction

The Student Utility Tools (SUT) project is a terminal-based tool developed using Bash scripting. It aims to provide students with a collection of essential tools that are useful for daily calculations and conversions. The tool is designed to help students manage a variety of tasks, from calculating grades to converting currencies, all within a simple command-line interface. The project is an excellent example of using the power of Bash scripting to solve real-world problems in a minimalistic yet efficient manner.

3. Objective

The primary objective of the Student Utility Tools (SUT) project is to create a consolidated utility tool that includes several functions commonly used by students. These include basic mathematical operations, unit conversions, and specific calculations such as Body Mass Index (BMI), temperature conversion, and currency conversion. The project demonstrates how to combine different tools into a single script with an easy-to-use, interactive interface.

4. Tools and Technologies

Programming Language: Bash
Operating System: Linux or any Unix-based operating system with Bash shell (e.g., macOS, Ubuntu, CentOS, etc.)
Text Editor: Any text editor (VS Code, Nano, Sublime Text, etc.)
Execution Environment: Terminal (Shell or Command Line Interface)

Why Bash?

Bash was chosen for its widespread use in Linux environments and its ability to interact directly with the operating system. It's lightweight, fast, and well-suited for handling command-line user interfaces, making it an ideal choice for simple applications like this one.

5. Features

The Student Utility Tools (SUT) includes multiple utilities designed to assist students in various areas. The following are the available features:

1. Simple Calculator

Performs basic arithmetic operations such as addition, subtraction, multiplication, division, and modulus. Allows the user to input two numbers and select the operation they wish to perform.

2. Grade Calculator

Accepts input for the number of subjects and their corresponding marks. Calculates the average mark and provides a grade based on the following scale:
A+: 80-100
A: 75-79
B+: 70-74
B: 65-69
D+: 60-64
D: 55-59
C+: 45-54
C: 40-44
F: Below 40

3. Temperature Converter

Converts temperatures between different units (Celsius, Fahrenheit, and Kelvin). Provides various options for converting between units, such as Celsius to Fahrenheit or Kelvin to Celsius.

4. BMI Calculator

Calculates the Body Mass Index (BMI) based on the user's weight and height. Provides the corresponding BMI category (underweight, normal weight, overweight) based on the calculated BMI value.

5. Weight to Pound Converter

Converts weight from kilograms to pounds, making it easier for students to switch between metric and imperial systems.

6. Number Converter

Converts numbers between different number systems, including Decimal, Binary, Octal, and Hexadecimal.

7. Factorial Calculator

Computes the factorial of a given number, demonstrating the use of loops in Bash to perform repetitive tasks.

8. Currency Converter

Converts values between various currencies (BDT, USD, EUR, and INR) using fixed conversion rates. Supports multiple currency conversion options (e.g., BDT to EUR, EUR to USD, etc.).

9. VAT Calculator

Calculates the total price after VAT (Value Added Tax) is applied based on the given tax rate and initial price.

10. Discount Calculator

Calculates the discount amount and the amount saved after applying a discount percentage to a given price.

6. Working Mechanism

The script works as follows:
1. Initial Setup: When the user runs the script, they are presented with a welcome message and the main menu.
2. User Interaction: The user is prompted to choose one of the available tools by selecting the corresponding option.
3. Tool Execution: Based on the user's choice, the script executes the relevant logic for the selected utility. This is done using conditional statements (if, elif, case) and loops.
4. Output Generation: The result of the chosen operation is displayed, and the user is then given the option to return to the main menu to perform another task or exit the program.

Flowchart of the Script:

Start -> Display Main Menu -> User Selects Option -> Execute Corresponding Function -> Show Result and Return to Menu -> Exit or Continue

7. Advantages

1. Simplicity: The tool is easy to use and doesn't require installation of additional software.
2. Efficiency: It eliminates the need for students to open multiple applications or websites to perform different tasks.
3. All-in-One Solution: Combines multiple essential tools in a single script, saving time and effort.
4. Cross-Platform: Can run on any system with Bash installed, making it highly portable.
5. Cost-Effective: Since it runs in a terminal, it doesn't require a graphical user interface, making it suitable for older systems or minimal setups.

8. Limitations

1. No Real-Time Currency Conversion: The script uses static conversion rates, which can become outdated. A future version could fetch live rates from an API.
2. No Input Validation: The script doesn't perform input validation, so users could input invalid data (e.g., non-numeric values), which could cause errors.
3. Limited to Basic Functionalities: It only supports basic versions of each utility. For example, the grade calculator doesn't take weighted subjects into account, and the currency

converter only supports a limited set of currencies.
4. No Advanced Error Handling: The script doesn't handle all possible edge cases, such as dividing by zero in the calculator.

9. Conclusion

The Student Utility Tools (SUT) project successfully provides an easy-to-use terminal-based solution for students to perform common calculations and conversions. The script combines several essential utilities into one package, saving users time and effort. It serves as a practical example of Bash scripting, demonstrating how to create interactive tools with basic user interfaces and logic. The project is highly accessible, lightweight, and functional, making it an excellent tool for any student.

10. Future Improvements

1. Input Validation: Implement input checks to ensure that users enter valid data, especially for numeric input.
2. Dynamic Currency Rates: Integrate an API to fetch real-time currency conversion rates for more accurate results.
3. Additional Units and Conversions: Add more units for conversion, such as speed (km/h to mph), distance (km to miles), or time (minutes to seconds).
4. Graphical User Interface (GUI): A GUI could be created using Python and Tkinter or a web-based interface to make the tool more user-friendly.
5. Error Handling: Add proper error handling for edge cases, such as division by zero, invalid inputs, or unexpected results.

11. Acknowledgements

We would like to acknowledge the following:
- Bash Scripting Resources: For providing a comprehensive understanding of Bash scripting and enabling the development of this project.
- Online Forums and Communities: For sharing their knowledge on scripting techniques and best practices.

12. References

1. 'Bash Scripting for Beginners,' tutorials (https://www.tutorialspoint.com/unix/bash_scripting.htm)
2. 'Introduction to Linux,' documentation (https://www.gnu.org/software/bash/manual/bashref.html)