Product Requirements for the Unit Conversion Program:

1. *User-Friendly Interface:*
    - *The program should have a clear and intuitive interface, guiding users through the unit conversion process.*
    - *Display a menu with options for different conversion categories (length, weight, volume, temperature).*
    - *Prompt the user to enter their choice and validate the input to ensure a valid selection.*
2. *Conversion Categories:*
    - *The program should support conversion in multiple categories, including length, weight, volume, and temperature.*
    - *Provide separate conversion functions for each category to handle specific conversion calculations.*
3. *Input Validation:*
    - *Validate user input to ensure only valid choices and values are accepted.*
    - *Handle invalid inputs gracefully by displaying appropriate error messages and prompting the user to re-enter the correct information.*
4. *Length Conversion:*
    - *Support conversions between meters and feet.*
    - *Allow users to input a value in either meters or feet and display the converted value based on the selected conversion.*
5. *Weight Conversion:*
    - *Support conversions between kilograms and pounds.*
    - *Allow users to input a value in either kilograms or pounds and display the converted value based on the selected conversion.*
6. *Volume Conversion:*
    - *Support conversions between liters and gallons.*
    - *Allow users to input a value in either liters or gallons and display the converted value based on the selected conversion.*
7. *Temperature Conversion:*
    - *Support conversions between Celsius and Fahrenheit.*
    - *Allow users to input a value in either Celsius or Fahrenheit and display the converted value based on the selected conversion.*
8. *Error Handling:*
    - *Handle exceptional cases, such as division by zero or out-of-range values, with appropriate error messages.*
    - *Ensure that the program does not crash or produce incorrect results when faced with invalid inputs.*

9. *Error Tolerance:*
    - *Provide a certain level of error tolerance in the converted values to accommodate decimal precision and rounding errors.*
10. *Modularity and Reusability:*
    - *Implement the program using modular functions to enhance code organization and reusability.*
    - *Each conversion category should have its dedicated function to perform the conversion calculations, promoting easy maintenance and future expansion.*
11. *Clear Output Presentation:*
    - *Display the converted values in a clear and well-formatted manner.*
    - *Include appropriate units of measurement with the converted values to provide context.*
12. *Graceful Program Termination:*
    - *Allow the user to exit the program gracefully at any point.*
    - *Provide an option to exit the program from the main menu.*
13. *Documentation:*
    - *Include clear and concise documentation describing the program's functionality, usage instructions, and any additional information that may be useful to the user.*
14. *Portability:*
    - *Ensure the program is compatible with common C compilers and platforms to allow users to run it on different systems.*
15. *Performance:*
    - *Aim for efficient and responsive performance, minimizing delays and unnecessary computations.*