بِسْمِ ٱللَّٰهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

**SOFTWARE CONFIGURATION MANAGEMENT**

**NAME:**

**Umair Ali (2022-ag-8025)**

**M. Zulqarnain Awan (2022-ag-7995)**

**M. Muneeb Hussain (2022-ag-8031)**

**Ahmad Nadeem (2022-ag-7993)**

**Submit To:**

**Mam Anum Khalid**

**Currency Converter Application Documentation**

**Table of Contents**

1. Introduction
2. Objectives
3. Features
4. Technologies Used
5. System Requirements
6. Project Structure
7. Code Explanation
8. Future Enhancements
9. Testing and Debugging
10. Challenges Faced
11. Lessons Learned
12. Conclusion

**1. Introduction**

Currency conversion applications are essential tools for individuals and businesses dealing with international transactions. This project involves creating a Python-based currency converter with two main versions. The first version converts USD to PKR, while the second version extends the functionality to include BTC to PKR conversion. Both versions offer an intuitive and user-friendly interface using the tkinter library. The project demonstrates progressive development through version control, allowing incremental feature additions and improvements.

Currency exchange rates fluctuate constantly, making it imperative to have tools that provide accurate calculations. This project is a step toward creating a functional and scalable application that serves as a learning experience for Python programming, GUI development, and version control.

**2. Objectives**

The primary goals of the project are:

* To provide a simple, reliable tool for currency conversion.
* To implement and enhance programming skills in Python.
* To explore GUI development using tkinter.
* To demonstrate version control by incrementally adding new features.
* To familiarize users with basic financial concepts related to currency conversion.
* To offer a platform for practicing software development lifecycle concepts, including testing and debugging.

**3. Features**

**Version 1: USD to PKR Converter**

* Takes the exchange rate of USD to PKR as input.
* Accepts the amount in USD.
* Displays the equivalent amount in PKR.
* Simple and user-friendly interface.
* Error handling for invalid inputs.

**Version 2: BTC to PKR Converter**

* Adds BTC to PKR conversion functionality.
* Takes BTC to USD and USD to PKR rates as input.
* Accepts the amount in BTC and displays the equivalent amount in PKR.
* Retains simplicity and usability in the GUI.
* Enhanced error handling and clear user prompts.
* Provides feedback on incorrect or missing inputs for better user experience.

**4. Technologies Used**

* **Programming Language**: Python
* **GUI Framework**: tkinter
* **Version Control**: Git and GitHub for tracking changes and managing versions.
* **Development Environment**: Any Python IDE (e.g., PyCharm, Visual Studio Code, or IDLE)

**5. System Requirements**

* **Operating System**: Windows, macOS, or Linux
* **Python Version**: Python 3.8 or above
* **Dependencies**: tkinter (pre-installed with Python)
* **Hardware**: Minimum 2 GB RAM and 1 GHz processor
* **Internet Connection**: Required for potential future integration with real-time API services

**6. Project Structure**

**Version 1 (USD to PKR Converter)**

1. Accepts user input for:
   * USD to PKR exchange rate
   * Amount in USD
2. Calculates and displays the PKR equivalent.
3. GUI includes labels, input fields, a button, and a result display.
4. Implements basic error handling for non-numeric inputs.

**Version 2 (BTC to PKR Converter)**

1. Accepts user input for:
   * BTC to USD exchange rate
   * USD to PKR exchange rate
   * Amount in BTC
2. Calculates and displays the PKR equivalent.
3. GUI includes labels, input fields, buttons, and a result display.
4. Introduces comprehensive error handling and dynamic user feedback.
5. Maintains a clear separation of concerns, ensuring modularity for future extensions.

**7. Code Explanation**

**Version 1: USD to PKR Conversion**

import tkinter as tk

from tkinter import messagebox

def convert\_usd\_to\_pkr():

try:

usd\_amount = float(usd\_entry.get())

usd\_to\_pkr\_rate = float(usd\_to\_pkr\_rate\_entry.get())

pkr\_amount = usd\_amount \* usd\_to\_pkr\_rate

result\_label.config(text=f"{usd\_amount} USD = {pkr\_amount:.2f} PKR")

except ValueError:

messagebox.showerror("Input Error", "Please enter valid numbers for USD amount and exchange rate.")

root = tk.Tk()

root.title("Currency Converter - USD to PKR")

# Input Fields and Labels

tk.Label(root, text="USD to PKR Rate:").grid(row=0, column=0, sticky="e", pady=5)

usd\_to\_pkr\_rate\_entry = tk.Entry(root)

usd\_to\_pkr\_rate\_entry.grid(row=0, column=1, padx=5)

tk.Label(root, text="Amount in USD:").grid(row=1, column=0, sticky="e", pady=5)

usd\_entry = tk.Entry(root)

usd\_entry.grid(row=1, column=1, padx=5)

# Convert Button

tk.Button(root, text="Convert USD to PKR", command=convert\_usd\_to\_pkr).grid(row=2, column=0, columnspan=2, pady=10)

# Result Label

result\_label = tk.Label(root, text="", font=("Arial", 12, "bold"), fg="blue")

result\_label.grid(row=3, column=0, columnspan=2, pady=10)

root.mainloop()

**Version 2: BTC to PKR Conversion**

import tkinter as tk

from tkinter import messagebox

def convert\_btc\_to\_pkr():

try:

btc\_amount = float(btc\_entry.get())

btc\_to\_usd\_rate = float(btc\_to\_usd\_rate\_entry.get())

usd\_to\_pkr\_rate = float(usd\_to\_pkr\_rate\_entry.get())

usd\_value = btc\_amount \* btc\_to\_usd\_rate

pkr\_amount = usd\_value \* usd\_to\_pkr\_rate

result\_label.config(text=f"{btc\_amount} BTC = {pkr\_amount:.2f} PKR")

except ValueError:

messagebox.showerror("Input Error", "Please enter valid numbers for BTC amount and exchange rates.")

root = tk.Tk()

root.title("Currency Converter - BTC to PKR")

# Input Fields and Labels

tk.Label(root, text="BTC to USD Rate:").grid(row=0, column=0, sticky="e", pady=5)

btc\_to\_usd\_rate\_entry = tk.Entry(root)

btc\_to\_usd\_rate\_entry.grid(row=0, column=1, padx=5)

tk.Label(root, text="USD to PKR Rate:").grid(row=1, column=0, sticky="e", pady=5)

usd\_to\_pkr\_rate\_entry = tk.Entry(root)

usd\_to\_pkr\_rate\_entry.grid(row=1, column=1, padx=5)

tk.Label(root, text="Amount in BTC:").grid(row=2, column=0, sticky="e", pady=5)

btc\_entry = tk.Entry(root)

btc\_entry.grid(row=2, column=1, padx=5)

# Convert Button

tk.Button(root, text="Convert BTC to PKR", command=convert\_btc\_to\_pkr).grid(row=3, column=0, columnspan=2, pady=10)

# Result Label

result\_label = tk.Label(root, text="", font=("Arial", 12, "bold"), fg="blue")

result\_label.grid(row=4, column=0, columnspan=2, pady=10)

root.mainloop()

**8. Future Enhancements**

1. **Multi-Currency Support:** Add options to convert other currencies like EUR, GBP, or CAD to PKR.
2. **API Integration:** Fetch real-time exchange rates using APIs like Open Exchange Rates or Fixer.io.
3. **Historical Data:** Allow users to view historical exchange rates.
4. **Mobile-Friendly GUI:** Create a responsive interface or deploy as a mobile application.
5. **Conversion History:** Save conversion history locally or in a database for later reference.
6. **Customization Options:** Enable users to select preferred currency symbols and themes for the GUI.
7. **Multi-Language Support:** Integrate different languages to enhance accessibility for diverse users.

**9. Testing and Debugging**

1. **Input Validation:** Tested with various valid and invalid inputs to ensure error messages are displayed correctly.
2. **Calculation Accuracy:** Verified the accuracy of conversion formulas with real-world data.
3. **GUI Responsiveness:** Ensured all buttons and input fields work seamlessly without delays.
4. **Error Handling:** Checked robustness against missing or incorrect user inputs.
5. **Cross-Platform Testing:** Verified performance across different operating systems, including Windows and macOS.

**10. Challenges Faced**

1. **Dynamic GUI Design:** Balancing simplicity with functionality in the interface design.
2. **Error Handling:** Ensuring comprehensive error messages for all possible input issues.
3. **Scalability:** Designing the code structure to easily accommodate additional features in future versions.
4. **Testing Variability:** Handling diverse input cases during debugging to identify edge cases.
5. **Time Management:** Allocating sufficient time to integrate and test new features between versions.

**11. Lessons Learned**

1. **Version Control:** Incremental development improves project manageability and allows systematic testing.
2. **User-Centric Design:** Simple, intuitive interfaces enhance usability and user experience.
3. **Error Handling:** Thorough validation is crucial for robust applications.
4. **Project Documentation:** Clear documentation aids in understanding and maintaining the codebase.
5. **Adaptability:** Flexibility in design choices ensures easier future enhancements.

**12. Conclusion**

The Currency Converter project demonstrates the fundamentals of Python programming, GUI development, and progressive version control. It provides a practical and user-friendly solution for currency conversion while allowing room for scalability and feature enhancements in the future. With the integration of real-time exchange rates, additional currencies, and multi-platform support, this application has the potential to cater to a broader audience effectively.

**THANK YOU**