CURRENCY CONVERTER-PROJECT REPORT

Submitted by: - PALAK BHATLA

Registration no.: - 25BCY10014

Course: - Introduction to Problem Solving and Programming

Platform: - VITyarthi - Build Your Own Project

1.Introduction: -

Because of market dynamics, currency exchange rates change from time to time in the global economy. Real-Time Currency Converter is a desktop application that was designed to provide users with up-to-the-minute exchange rates. Apart from the static conversion tables found in most text materials, it integrates with the live financial API to fetch real-time data that allows travellers, international students, and traders to instantly and accurately convert specific amounts in varying international currencies.

2. Problem Statement: -

The Problem:

Manual currency conversion with outdated sources, such as newspapers or static websites, often causes financial discrepancies. No lightweight, centralized desktop tool exists that fetches live market data without forcing users to dig through overly complex and advertisement-heavy financial news websites.

The Solution:

A Python-based client-side application that connects a user to a live API, performs automatic and very precise calculations, and keeps a local history of the user's transactions for future use.

3. Functional Requirements: -

The system provides three major functional modules:

1. Live Rate Extraction (API Integration)
2. Conversion Engine & User Interface
3. History & Reporting (Data Persistence)

4.Non-Functional Requirements: -

These define how the system performs its functions:

- **Performance:** For typical network conditions, API responses and calculations must be processed in 2 seconds or less.
- **Reliability (Offline Mode):** In the case of an internet connection failure, the system should not crash. It automatically switches to "Offline Mode" using cached or default fallback rates.
- **Usability:** The design has a modern UI (card layout, clear typography), and the user will be able to perform a conversion in less than 3 clicks.
- **Error Handling:** The system should handle invalid input-namely, text in numeric fields-gracefully, with clear, user-friendly warning messages rather than termination of the system.

5. System Architecture: -

The project follows a Modular Architecture, separating concerns into distinct layers:

- **Presentation Layer (GUI):** Built using Tkinter, this handles all user interactions and displays data.
- **Logic Layer (Calculator):** Contains the pure Python functions for mathematical conversion and input validation.
- **Data Layer (API & Storage):**
  - api_handler.py: Manages HTTP requests to the external API.
  - storage.py: Manages reading/writing history to the local CSV file.

6. Design Diagrams

6.1 Use Case Diagram: -

- Actor: User
- Use Cases:
  - Select "From" Currency
  - Select "To" Currency
  - Input Amount
  - View Conversion Result
  - View Transaction History
  - System: Fetch Live Rates

6.2 Workflow Diagram

Start **->** Launch App **->** Check Internet **->** Fetch API Data **->** User Inputs Amount **->** Validate Input **->** Calculate **->** Display Result **->** Save to History **->** End.

6.3 Sequence Diagram

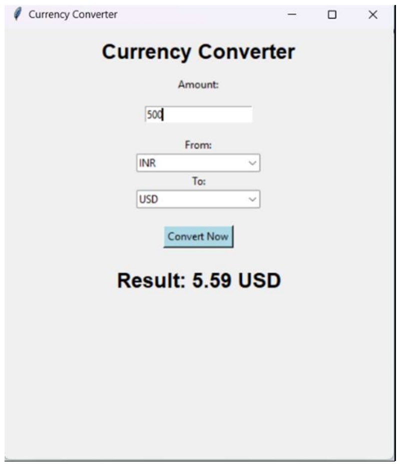
1. User enters amount and clicks "Convert".
2. GUI calls Logic.validate().
3. If valid, GUI calls API.get_rates() (if not already cached).
4. API returns JSON data.
5. GUI calls Logic.convert().
6. Logic returns the result.
7. GUI updates the display and calls Storage.save().

7.Design Decisions & Rationale: -

- **Language:** Python was chosen for its simplicity and powerful libraries (requests for HTTP).
- **GUI Framework:** Tkinter was selected because the requirement was for a lightweight *desktop* application that runs locally without a browser.
- **API:** ExchangeRate-API was selected for its reliable free tier and simple JSON response structure.

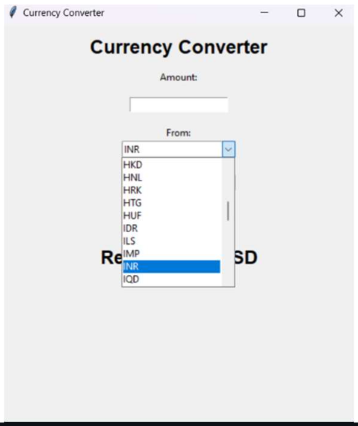
8.Screenshots / Results: -

1.Entering the amount for conversion from INR to USD

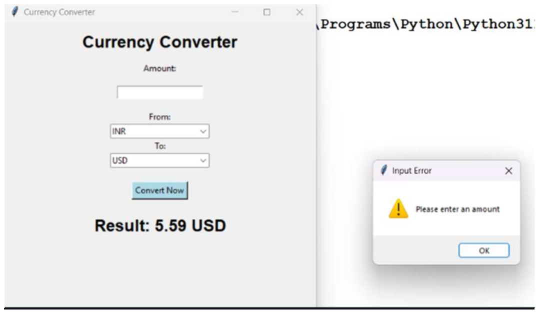


500
INR
USD
Result: 5.59 USD

2.Options available for conversion in form of list in drop down way



3.Error generated when no amount is entered and pressed convert now



Result: 5.59 USD

Please enter an amount

```
\Programs\Python\Python31
```

9. Testing Approach: -

- **Unit Testing:** Mathematical functions were tested with known values (e.g., 1 USD = 1 USD) to ensure accuracy.
- **Boundary Testing:** Tested inputs with 0, negative numbers, and very large numbers to ensure the validation logic holds.
- **Network Testing:** The application was launched with Wi-Fi disabled to verify that the exception handling correctly triggered the "Offline Fallback" mode.

10. Challenges Faced: -

- **Unit Testing:** The mathematical functions are tested with known values, such as 1 USD = 1 USD.
- **Boundary testing:** Testing the input with 0, negative numbers, and very large numbers to see if the validation logic holds.
- **Network Testing:** The application was first launched with Wi-Fi disabled to check whether the "Offline Fallback" mode would be appropriately triggered by the exception handling.

11. Learnings & Key Takeaways: -

- Gained practical experience in consuming APIs and parsing JSON data in Python.
- Learned the importance of Modular Design—separating logic from the UI made debugging significantly easier.
- Understood how to manage Data Persistence using simple file handling techniques.

12. Future Enhancements: -

- **Currency Graphs:** Add a visualization module (using Matplotlib) to show rate trends over the last 30 days.
- **Multi-Language Support:** Localize the UI strings for non-English users.
- **Crypto Support:** Integrate a second API to support Bitcoin and Ethereum conversions.

13. References

1. **Python Documentation**: https://docs.python.org/3/
2. **Tkinter Reference:** https://tkdocs.com/
3. **ExchangeRate-API Documentation:** https://www.exchangerate-api.com/docs