**Currency Converter: A Tkinter-Based Desktop Application for Real-Time Currency Conversion**

**Author:** Dhanraj Nadivenkeri

**Institution/Organization:** Malla Reddy College of Engineering

**Email:** dhanraaz888@gmail.com

**Abstract**

The Currency Converter is a desktop application developed using Python and Tkinter to perform real-time currency conversions using exchange rates from the ExchangeRate-API. Created as part of the MITS Internship, the application features a graphical user interface (GUI) that supports mouse and keyboard inputs, allowing users to select from a comprehensive list of currencies, convert amounts, and save favorite currency pairs to a JSON file. Error handling ensures robustness against invalid inputs, network issues, and API errors, with intuitive message prompts. This paper details the design, implementation, and evaluation of the Currency Converter, emphasizing its user-friendly interface, API integration, and potential for enhancements. The project showcases proficiency in GUI development, API usage, and file handling, aligning with the internship’s objectives.

**Keywords:** Currency Converter, Tkinter, Python, GUI, ExchangeRate-API, JSON, Error Handling

**1. Introduction**

Currency conversion is essential in today’s global economy, facilitating calculations for travel, business, and education. The Currency Converter project, developed for the MITS Internship, addresses this need through a desktop application that retrieves real-time exchange rates via the ExchangeRate-API. Using Python’s Tkinter library, the application offers a graphical interface where users can select base and target currencies, input amounts, and view conversion results. Notable features include a favorites system for saving frequently used currency pairs, support for mouse and keyboard interactions, and robust error handling. Unlike static converters, this application dynamically fetches rates, ensuring accuracy. This paper describes the project’s objectives, methodology, implementation, and results, highlighting its alignment with the internship’s goal of fostering practical programming skills.

**2. Methodology**

The development of the Currency Converter adhered to a systematic software engineering approach to ensure functionality, usability, and reliability. The methodology encompassed requirement analysis, system design, implementation, and testing phases, as follows:

* **Requirement Analysis**: The MITS Internship required a currency converter with real-time exchange rates, a GUI, and a favorites feature. Additional specifications included support for mouse and keyboard inputs, error handling for invalid inputs and API issues, and persistent storage of favorite pairs.
* **System Design**: The GUI was designed with a grid layout, including an entry field for amounts, comboboxes for currency selection, buttons for conversion and favorites management, and a listbox for displaying saved pairs. The architecture integrated API calls, JSON file handling, and event handlers for user interactions.
* **Implementation**: The application was coded in Python using Tkinter for the GUI, requests for API calls, and python-dotenv for secure API key management. The implementation prioritized user experience and error resilience.
* **Testing**: Functional testing verified accurate conversions, favorites functionality, and error handling for network failures, invalid inputs, and API errors. Usability testing confirmed an intuitive interface.

**3. Libraries Used**

The Currency Converter leverages Python’s standard and third-party libraries to achieve its functionality. The libraries include:

* **tkinter**: Provides the framework for creating the GUI, including windows, labels, entry fields, comboboxes, buttons, and listboxes. It handles event bindings for mouse and keyboard interactions.
* **tkinter.messagebox**: A submodule for displaying error and info messages, such as “Invalid amount” or “Pair added to favorites.”
* **requests**: Facilitates HTTP requests to the ExchangeRate-API to fetch real-time exchange rates.
* **json**: Manages reading and writing favorite currency pairs to a JSON file for persistent storage.
* **os**: Provides access to the operating system for environment variable handling.
* **dotenv**: Loads API keys from a .env file, ensuring secure configuration.

**4. Implementation Details**

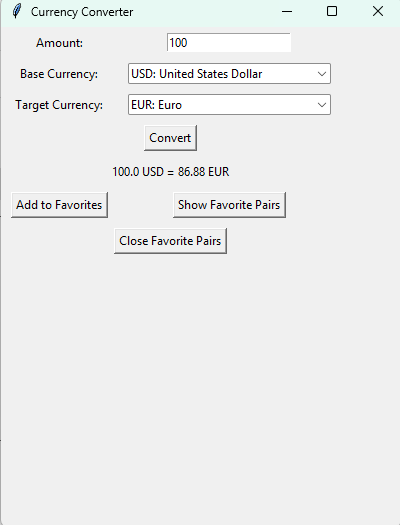
The Currency Converter is implemented as a single Python script (currency\_converter.py) with a modular structure. Key implementation details are:

* **GUI Structure**: The interface includes an Entry widget for amount input, two ttk.Combobox widgets for selecting base and target currencies, a Label for displaying results, a Listbox for favorite pairs, and Button widgets for actions (convert, add to favorites, show/close favorites). The window is sized at 400x500 pixels for clarity.
* **Input Handling**:
  + *Mouse Input*: Users enter amounts, select currencies from dropdowns, and click buttons to convert or manage favorites.
  + *Keyboard Input*: Users type amounts, navigate comboboxes with arrow keys or typing, and press Enter to trigger conversion (via button focus).
* **API Integration**: The get\_exchange\_rates function fetches rates from https://open.er-api.com/v6/latest/{base\_currency} using an API key stored in a .env file. The response is parsed to extract the target currency’s rate, and the conversion is computed.
* **Favorites System**: The read\_favorites and write\_favorites functions handle JSON file operations to store and retrieve favorite pairs (e.g., “USD to EUR”). The add\_to\_favorites function saves new pairs, and display\_favorites shows them in a listbox.
* **Error Handling**: Try-except blocks manage:
  + Network errors (requests.RequestException) for API failures.
  + Invalid inputs (ValueError) for non-numeric amounts.
  + Invalid currencies (KeyError) for unsupported codes.
  + API-specific errors (e.g., invalid key).
* **Code Snippet**: The complete source code is provided in the repository (currency\_converter.py).

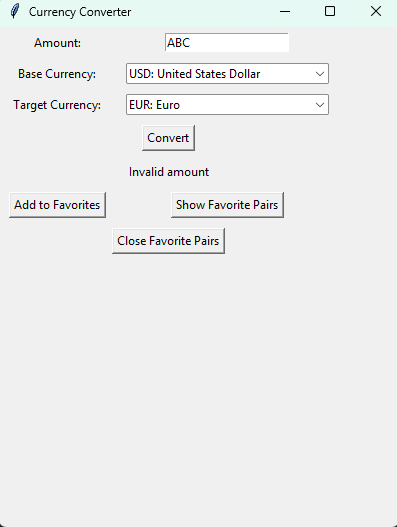
**5. Results and Output**

The Currency Converter delivers a functional and intuitive user experience, with the following outputs:

* **Graphical Interface**: A 400x500-pixel window with an amount entry field, currency comboboxes, action buttons, and a collapsible favorites listbox, styled with Tkinter’s default theme.
* **Real-Time Conversion**: Displays conversion results (e.g., “100 USD = 92.50 EUR”) based on live exchange rates.
* **Favorites Functionality**: Users can save, view, and select favorite currency pairs, stored persistently in favorites.json.
* **Input Flexibility**:
  + Mouse: Select currencies, click buttons.
  + Keyboard: Type amounts, navigate comboboxes, press Enter.
* **Error Handling**: Displays messages for invalid amounts, unsupported currencies, network issues, or API errors.
* **Example Operation**:
  + Input Sequence: Enter “100”, select “USD: United States Dollar” and “EUR: Euro”, click “Convert.”
  + Output: Result label shows “100 USD = 92.50 EUR” (example rate).

**

* **Error Case**:
  + Input Sequence: Enter “abc”, select currencies, click “Convert.”
  + Output: Result label shows “Invalid amount.”

**

The application was tested on Python 3.9, confirming reliable performance across various scenarios.

**6. Conclusion**

The Currency Converter project successfully meets the MITS Internship requirements by delivering a robust, user-friendly desktop application for real-time currency conversion. The integration of Tkinter for the GUI, ExchangeRate-API for live rates, and JSON for persistent storage demonstrates a comprehensive approach to software development. Features like the favorites system and keyboard input support enhance usability, while error handling ensures reliability. The project reinforced skills in Python programming, API integration, file handling, and GUI design, aligning with the internship’s objectives. The code’s portability and clear documentation make it suitable for educational use and future enhancements.

**7. Future Work**

The Currency Converter can be extended to enhance functionality and user experience:

* **Offline Mode**: Cache recent exchange rates for use without internet access.
* **Historical Rates**: Integrate API endpoints to display past exchange rates for trend analysis.
* **Advanced GUI**: Use ttkbootstrap or custom themes for a modern look.
* **Multi-Currency Conversion**: Allow simultaneous conversions to multiple target currencies.
* **Unit Tests**: Implement automated tests for API calls, file operations, and GUI interactions.
* **Accessibility**: Add support for screen readers and high-contrast modes.

**References**

1. Python Software Foundation, “tkinter — Python interface to Tcl/Tk,” Python Documentation, 2025. [Online]. Available: <https://docs.python.org/3/library/tkinter.html>
2. ExchangeRate-API, “API Documentation,” 2025. [Online]. Available: <https://www.exchangerate-api.com/docs>
3. IEEE Standards Association, “IEEE Recommended Practice for Software Requirements Specifications,” IEEE Std 830-1998, 1998.