PROJECT REPORT — CURRENCY CONVERTER APPLICATION

Program: Currency Converter with Real-Time Exchange Rate Visualization

Course: Introduction to problem solving & programming using python

Submitted by: Rehab Hussain

(25BSA10162)

College: VIT Bhopal

Introduction

Currency conversion is an essential requirement for students, travelers, and professionals workers with global transactions. This project implements a **Graphical User Interface (GUI) based Currency Converter** using **Python's Tkinter** library. The application fetches **real-time exchange rates** from the internet, converts values between currencies, and also visualizes exchange rate trends using **Matplotlib**.

The system provides a clean interface, automated calculations, and an integrated plotting module, making it a useful tool for educational and practical purposes.

Problem Statement

Users often require quick and accurate currency conversion. Manual conversions can lead to errors, and online tools may not always be accessible or customizable.
Thus, the problem is:

To develop a desktop-based application that retrieves real-time exchange rates, converts currency values accurately, and displays historical/extracted trends using a graphical interface.

Functional Requirements

A. Three Major Functional Modules

1. **Currency Data Fetching Module**
    - Retrieves exchange rates from an online source using requests and BeautifulSoup.
2. **Currency Conversion Module**
    - Converts any user-entered amount between selected source and target currencies.
3. **Visualization Module**
    - Uses Matplotlib to dynamically plot exchange rate trends within the GUI.

B. Input / Output Structure

Input	Output
Amount to convert	Converted amount
Source currency	Updated conversion value
Target currency	Graph of exchange rate trend
Trigger actions through buttons	Results displayed in GUI

C. Logical Workflow

1. User launches the application.
2. User selects:
    - Source currency
    - Target currency

- Amount to convert

3. App fetches real-time exchange rates from the internet.
4. Conversion result is displayed instantly.
5. User may click the "Graph" button to view exchange rate trends.
6. Application updates the graphical plot inside the Tkinter window.

Non-Functional Requirements

Performance

- Fast access and parsing of online exchange rate API/HTML.

- Graph rendering done efficiently using Matplotlib.

Security

- No sensitive data stored.
- Only reads publicly available currency data.

Usability

- Simple Tkinter-based GUI.
- Drop-down selectors improve ease of use.

Reliability

- Handles missing data or internet connectivity issues.
- Prompts the user via message boxes for invalid operations.

Scalability

- New currencies can be added easily.
- Visualization module supports extended datasets.

Maintainability

- Modular code structure.
- Uses Python’s standard libraries for compatibility.

Error Handling Strategy

- Try/except blocks for:
  - Network request failures
  - Invalid numeric input
  - Graph rendering issues

Logging / Monitoring

- Basic internal messages shown on GUI or console for debugging.

System Architecture

Layers:

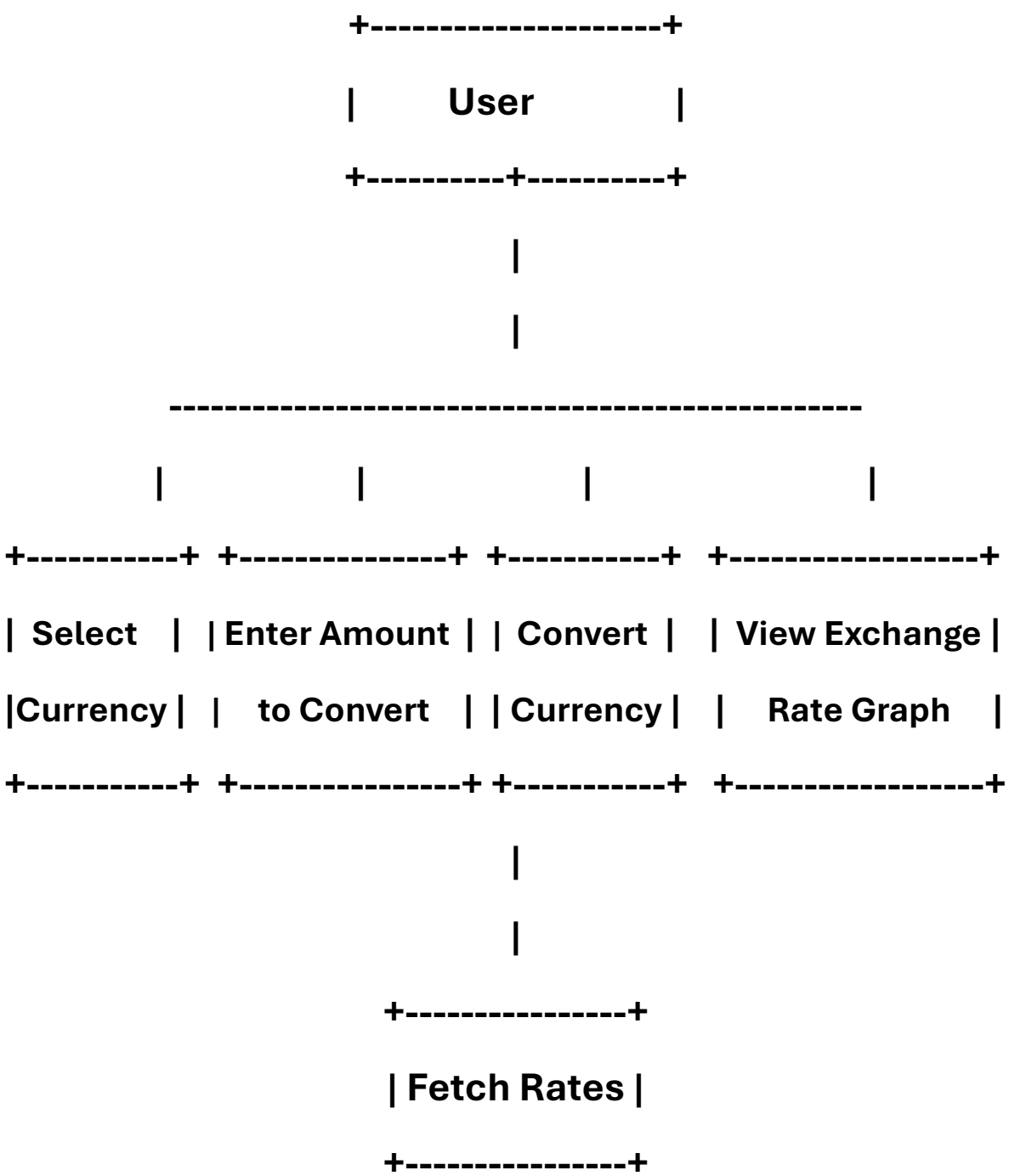
1. **Presentation Layer:** Tkinter GUI
2. **Logic Layer:** Conversion algorithms
3. **Data Layer:** Exchange rate scraping using Requests + BeautifulSoup
4. **Visualization Layer:** Matplotlib graphs embedded in Tkinter

Architecture Style:

→ Modular, event-driven GUI application.

Design Diagrams

A. Use Case Diagram



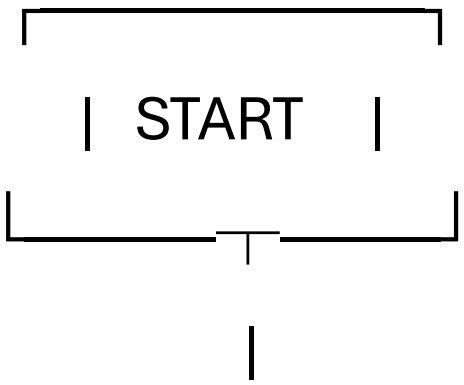
Actors:

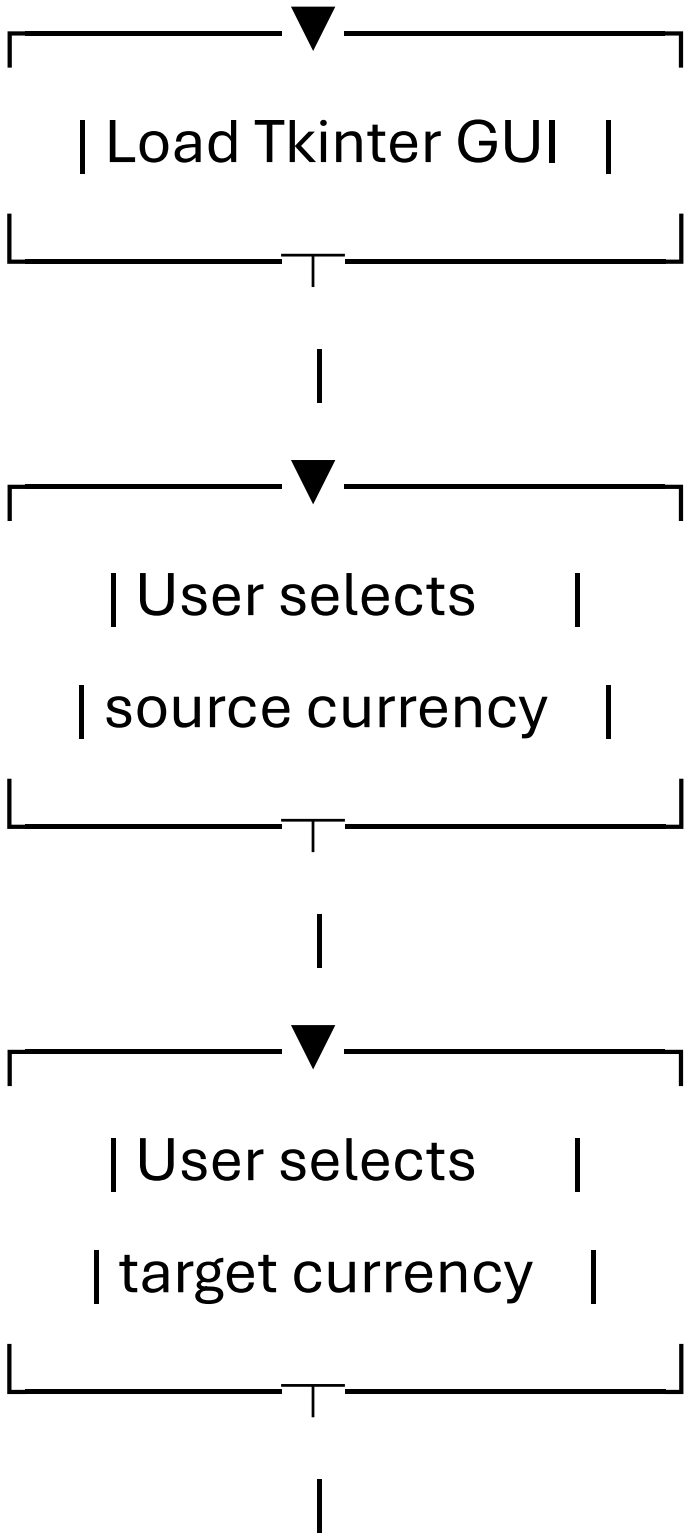
- User

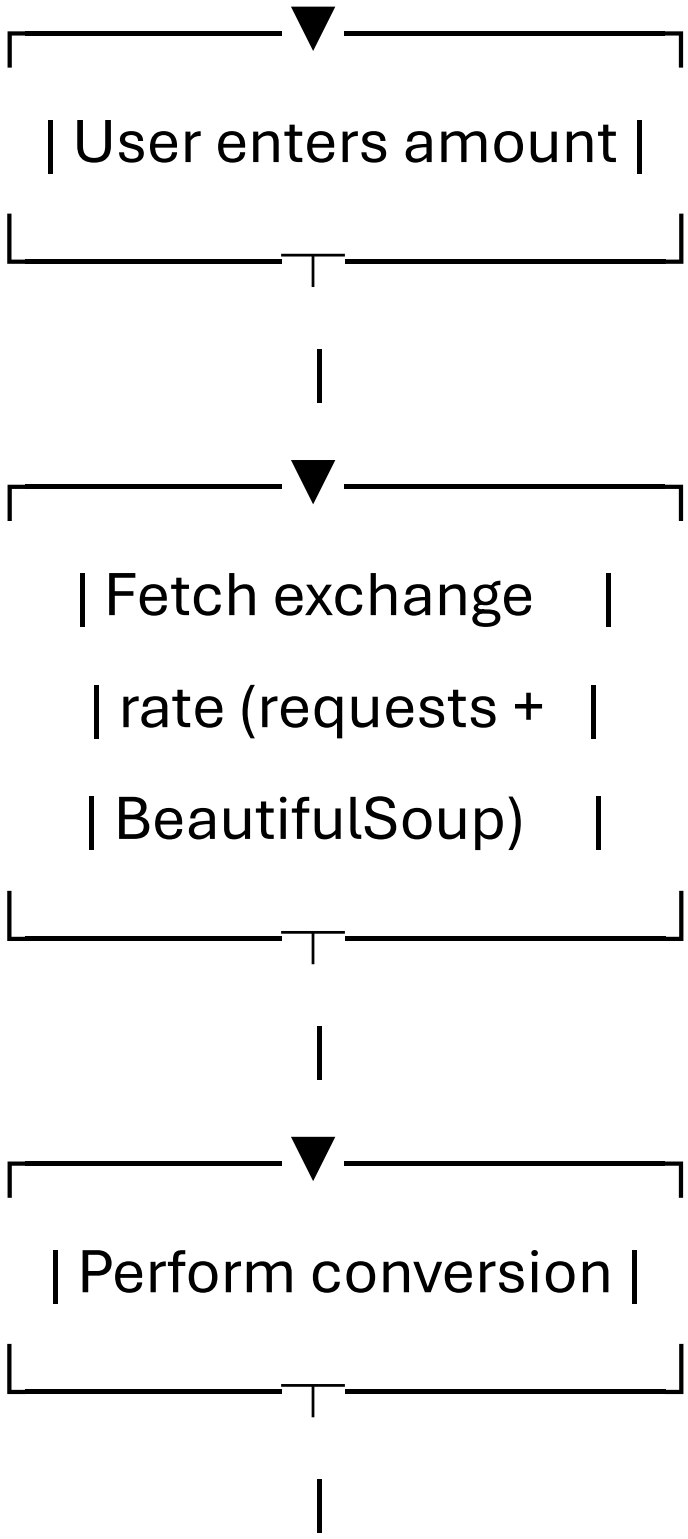
Use Cases:

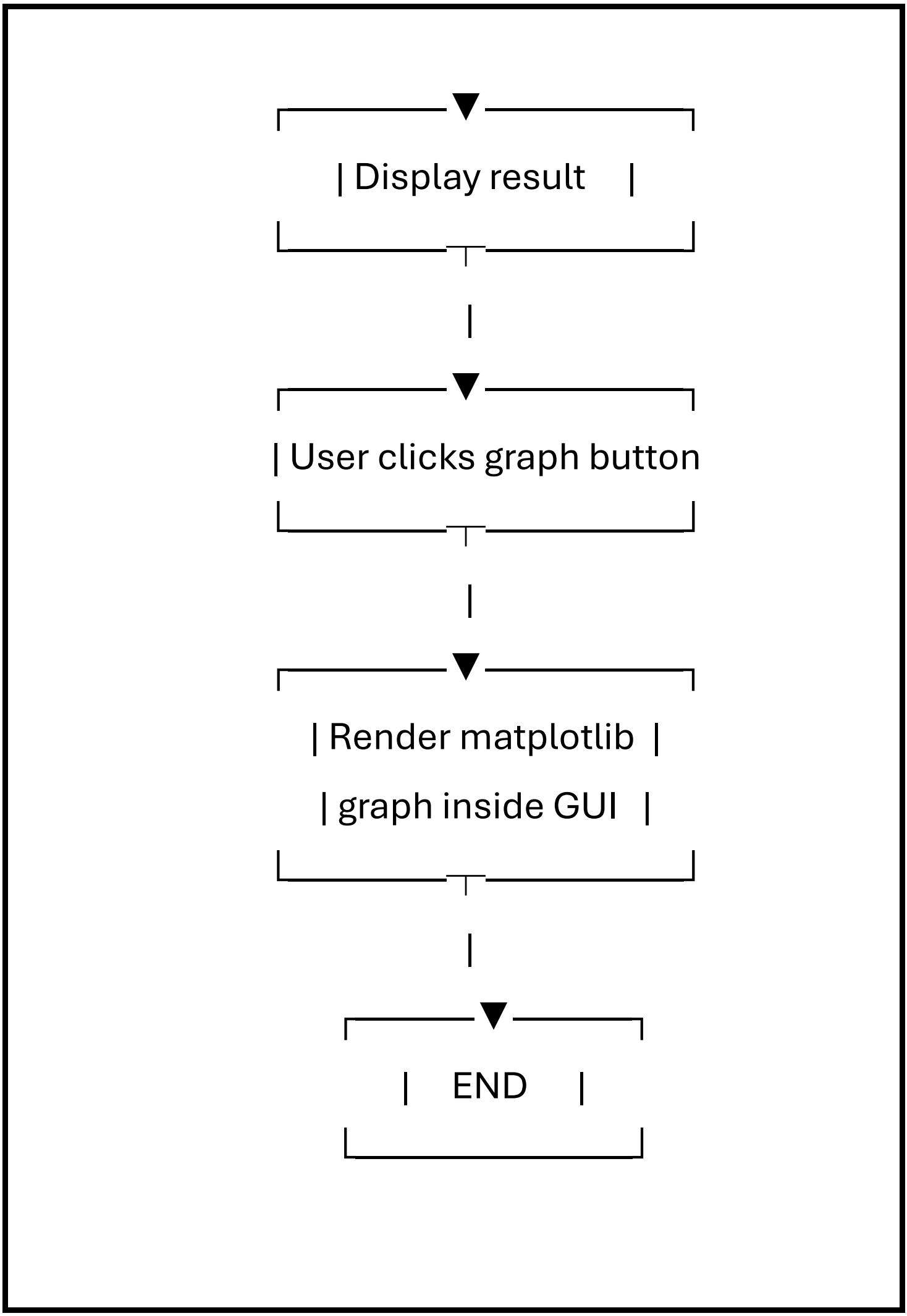
- Select currency
- Enter amount
- Convert currency
- Fetch real-time rates
- View trend graph

B. Workflow Diagram

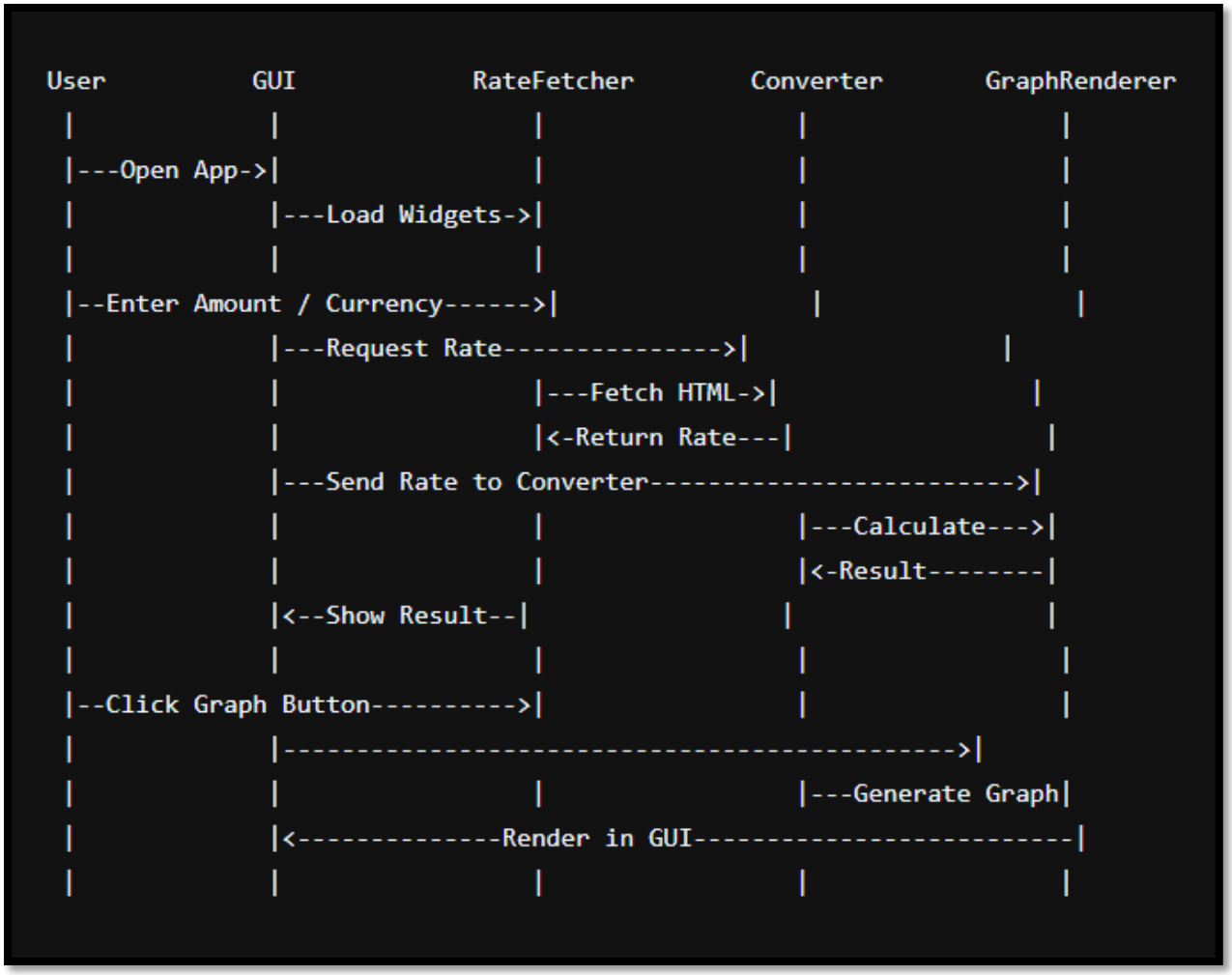



| Load Tkinter GUI |
| User selects |
| source currency |
| User selects |
| target currency |

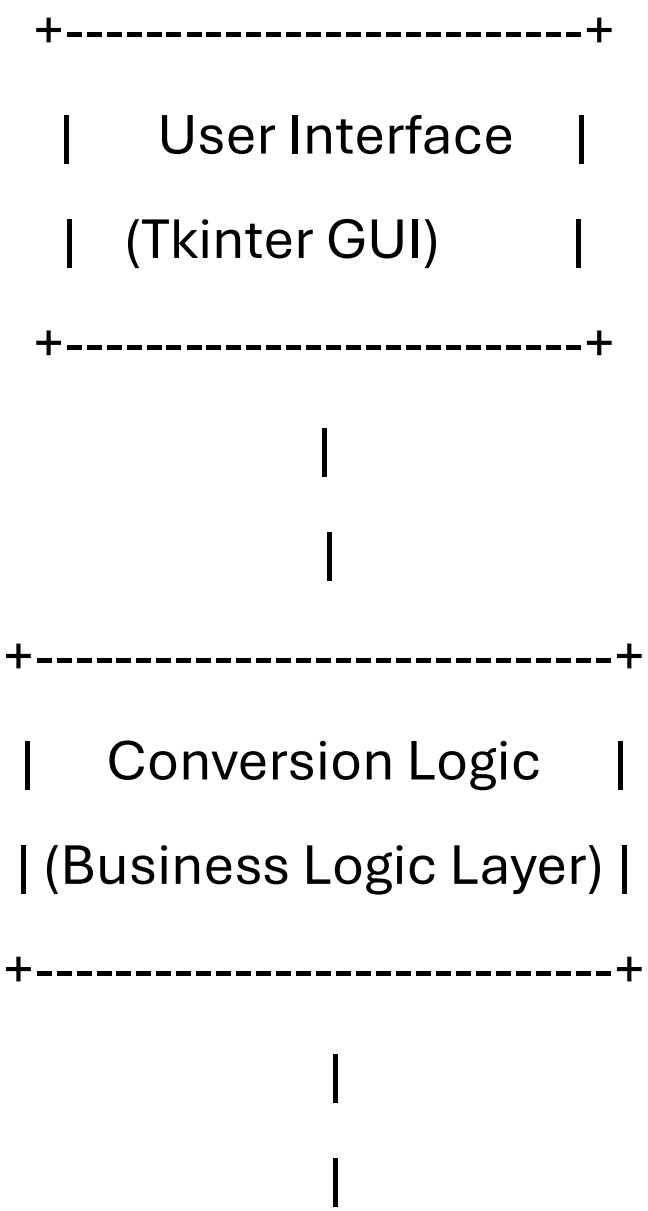

| User enters amount |
|
| Fetch exchange |
| rate (requests + |
| BeautifulSoup) |
|
| Perform conversion |
|

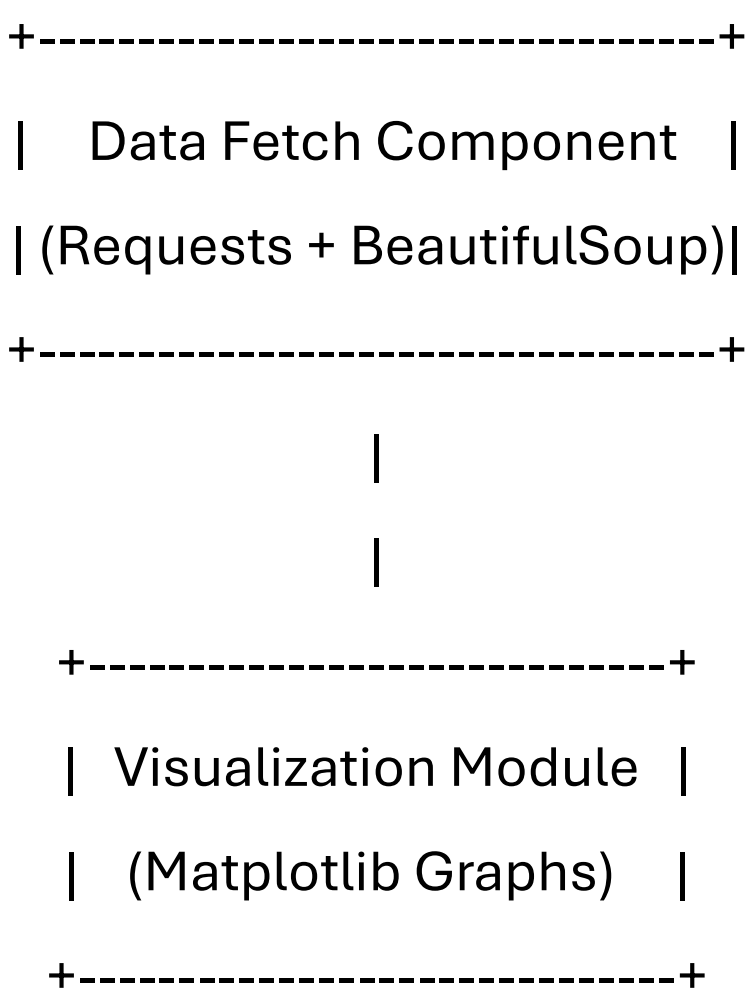

| Display result |
|
| User clicks graph button
|
| Render matplotlib |
| graph inside GUI |
|
| END |

C. Sequence Diagram



D. Component Diagram




Data Fetch Component
(Requests + BeautifulSoup)
Visualization Module
(Matplotlib Graphs)

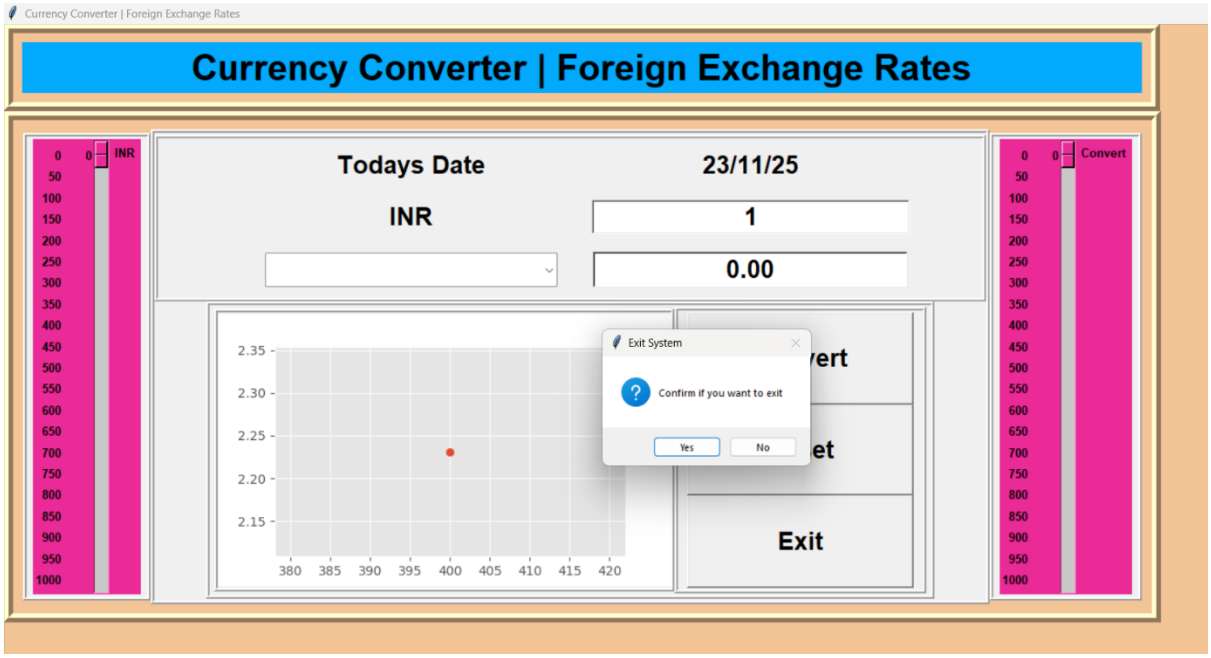
Design Decisions & Rationale

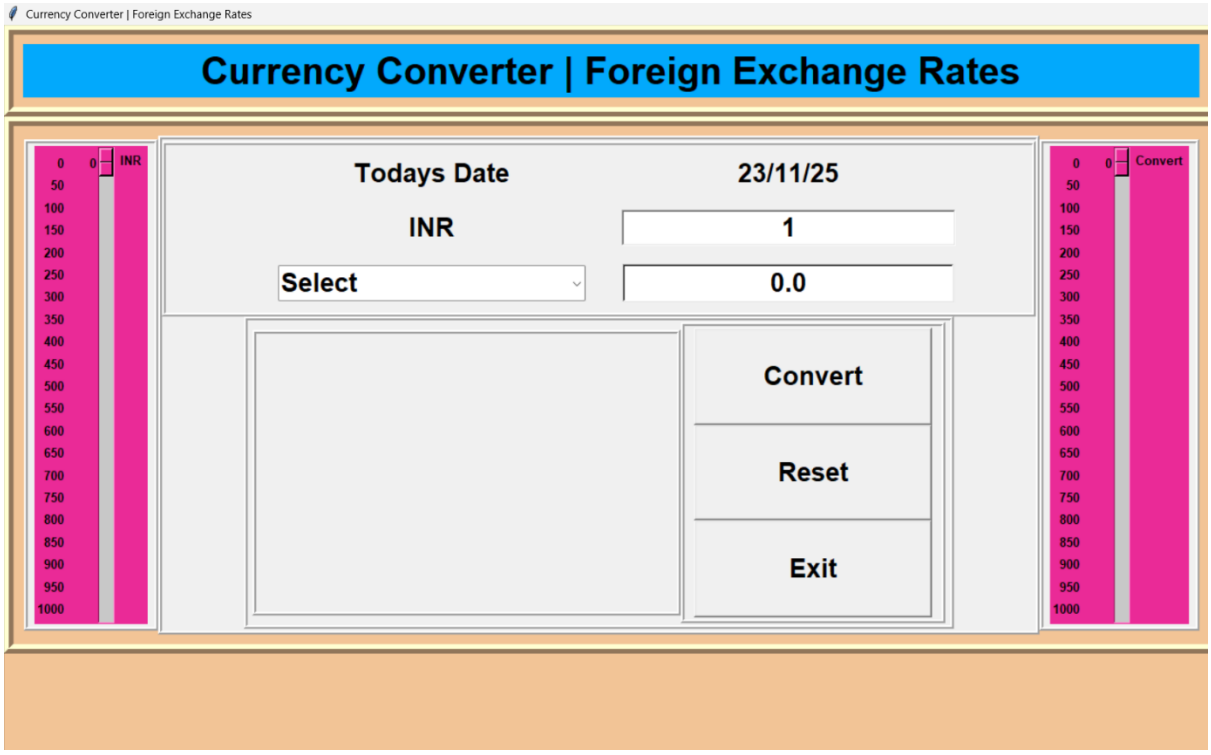
Decision	Rationale
Tkinter chosen for GUI	Lightweight, built-in, ideal for student projects
Requests + BeautifulSoup	Easy for web scraping and rate extraction
Matplotlib	Allows embedded, dynamic graph plotting
Modular code	Easier maintenance and extension

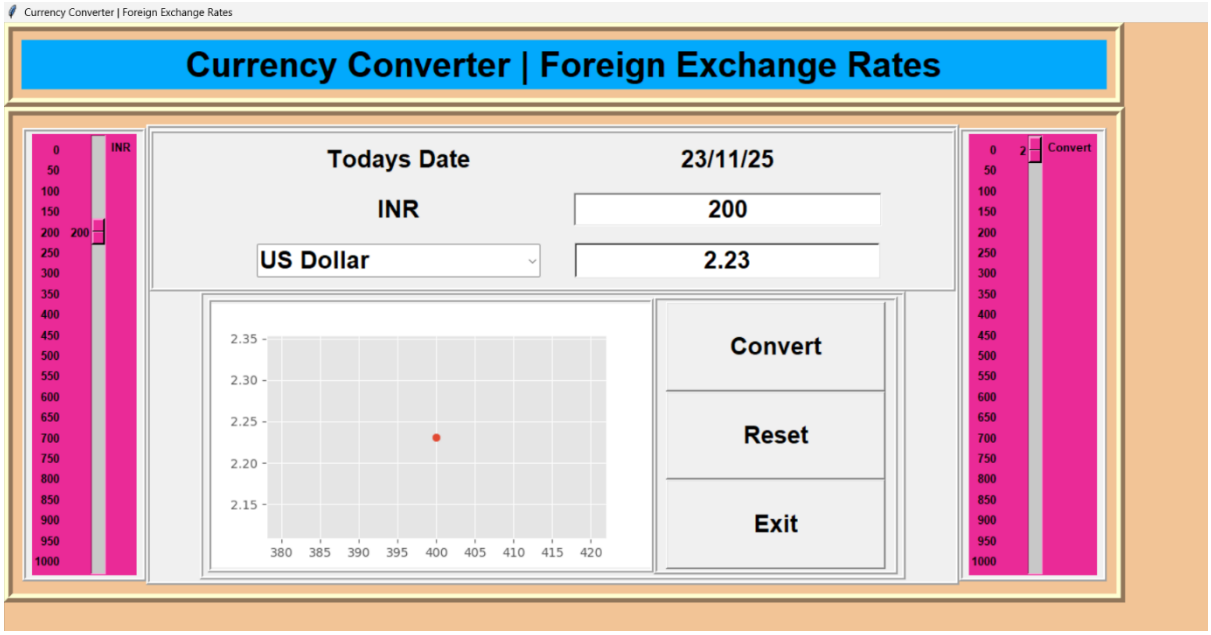
Implementation Details

- **Language:** Python
- **Libraries Used:**
  - tkinter (GUI)
  - requests (Web requests)
  - BeautifulSoup (HTML parsing)
  - matplotlib (Graphs)
- **Process:**
  1. Fetch exchange rate → parse value
  2. Convert using formula
  3. Display in GUI
  4. Render graph using canvas embedding

Screenshots / Results







Testing Approach

Testing Types Performed

- **Unit Testing:**
  - Input validation
  - Conversion accuracy
- **Integration Testing:**
  - GUI + Conversion logic
  - Graph embedding

- **System Testing:**
  - Full workflow test
- **Error Testing:**
  - No internet
  - Invalid user input

Sample Test Cases

Test Case	Input	Expected Output
Amount input	=text"abc"	Error popup
Missing currency selection	empty	Warning message
Valid values	100 USD → INR	Accurate conversion

Challenges Faced

- Handling inconsistent web-scraped data formats.
- Embedding Matplotlib graphs inside Tkinter.
- Managing internet failure scenarios.
- Ensuring GUI responsiveness during web requests.

Learnings & Key Takeaways

- Improved understanding of Tkinter GUI development.
- Learned to integrate **web scraping**, **data processing**, and **real-time plotting**.

- Understood modular programming and exception handling.
- Gained experience building a complete Python desktop application.

Future Enhancements

- Add support for more currencies.
- Introduce dark mode / modern UI themes.
- Use an official currency API instead of web scraping.
- Add offline caching of exchange rates.
- Export results to PDF or Excel.

References

- Python Tkinter Documentation
- BeautifulSoup Documentation
- Matplotlib Official Docs
- Real exchange rate websites used for scraping
- StackOverflow community examples