Project Report

Title: Currency Converter in Python using API and Flask

---

1. Introduction

A currency converter is a tool that converts the value of one currency into another based on real-time exchange rates.
In this project, we develop a **Python web application** using Flask that connects to a live exchange rate API to fetch current rates and perform currency conversion.
The project demonstrates how to:

- Interact with web APIs
- Handle JSON data
- Build a simple web interface with HTML
- Process user input through Flask routes

---

2. Objectives

- To build a Python program that converts amounts from one currency to another using live exchange rates.
- To use the **requests** module for making API calls.
- To practice extracting and processing JSON data from API responses.
- To implement a **Flask-based web application** with an HTML form for user interaction.

---

3. Tools and Technologies

- **Programming Language:** Python 3.x
- **Framework:** Flask (for web application development)

- **Modules Used:**
  - `requests` – for making HTTP requests to the API
  - `flask` – for creating web routes and rendering templates
  - `json` – for handling JSON data (optional since `requests` handles JSON directly)
- **API Used:** Exchange Rate API (`https://api.exchangerate-api.com`)
- **Frontend:** HTML form embedded in Flask templates
- **IDE/Text Editor:** Visual Studio Code / PyCharm / IDLE

---

4. Methodology

Step 1: Select a free exchange rate API and obtain the API endpoint.
Step 2: Use the requests module to fetch the latest exchange rates in JSON format.
Step 3: Build a **Flask web app** with a route (/) that handles both GET and POST requests.
Step 4: Create an HTML form to take input for amount, source currency, and target currency.
Step 5: Extract the relevant rate from the API response.
Step 6: Multiply the user-entered amount by the exchange rate.
Step 7: Display the converted amount dynamically on the same web page.

7. Advantages

- Fetches live exchange rates directly from the internet.
- Works for multiple currency pairs.
- Provides a **user-friendly web interface**.
- Simple and lightweight implementation.

---

8. Limitations

- Requires an internet connection.

- Dependent on the availability of the chosen API service.
- Free APIs may have usage limits.
- No authentication or caching implemented yet.

---

9. Future Enhancements

- Create a standalone GUI version using Tkinter or PyQt.
- Add caching to reduce API calls and improve performance.
- Support offline mode with stored rates.
- Implement better error handling for API failures.
- Add CSS and JavaScript for an improved user interface.
- Deploy the application on a web server (e.g., Heroku, PythonAnywhere).

---

10. Conclusion

This project successfully demonstrates how Python can interact with live web APIs to perform real-world tasks such as currency conversion.
By integrating Flask, the project moves beyond a console-based program into a **fully functional web application**. It is an excellent example of combining networking, data parsing, and web development in Python, and can be extended into a full-fledged currency converter application with advanced features.