**Computer science**

**PROJECT REPORT**

**GRADE – XII**

**PROJECT PREPARED BY:**

**NAME OF THE STUDENT : ……………………………………………………**

**ROLL NUMBER : …………………………………………………………**

**PROJECT TOPIC : …………………………………..................**

**Kendriya Vidyalaya MEG & Centre**

**2023 – 2024**

**CERTIFICATE**

This is to certify that the project work entitled **“………………………………………………………………………………………….”**

is a bonafide record of work done by**…………………………………… ,** Roll no:**……………….…**in partial fulfilment for the award of 12th standard during the academic year 2022 - 2023.

Date:

Registration No.:

Signature of Internal Signature of External

Examiner Examiner

Signature of Principal

# **ACKNOWLEDGEMENT**

I would like to take this opportunity to express my deep sense of gratitude to all those people without whom this project could have never been completed. First and foremost I like to thank God for giving me such a great opportunity to work on this project, and I would like to express my special thanks and gratitude to the Management, the Directors and the Correspondent of KV MEG & Centre, for their constant guidance and providing a very nice platform to learn.

I would also like to thank our Principal –Mr Lokesh Bihari Sharma sir, for his constant encouragement and moral support without which I would have never be able to give my best.

I would also like to thank Mrs Sumita ma’am, Computer Science Teacher, who gave me the wonderful opportunity to do this project, which also helped me in doing a lot of research and I came to know about so many new things from this study I am really thankful to all.

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Contents | Pg.No | Signature |
| 1 | SYSTEM REQUIREMENT |  |  |
| 2 | FEASIBILITY STUDY |  |  |
| 3 | ERRORS AND ITS TYPES |  |  |
| 4 | TESTING |  |  |
| 5 | MAINTENANCE |  |  |
| 6 | FUNCTION USED |  |  |
| 7 | FLOW CHART OF PROGRAM |  |  |
| 8 | SOURCE CODE |  |  |
| 9 | OUTPUT |  |  |
| 10 | CONCLUSION |  |  |
| 11 | BIBLIOGRAPHY |  |  |

# Project Report: Real-Time Currency Converter

## 1. Introduction

The Real-Time Currency Converter is a Python application designed to provide users with instant and accurate currency conversion. This report outlines the system requirements, feasibility study, error handling, testing, maintenance, functions used, flow chart, source code, output, conclusion, and references for the project.

## 2. System Requirements

### 2.1 Hardware Requirements

* Processor: Dual-core processor or higher
* RAM: 2 GB or higher
* Storage: 50 MB of free disk space

### 2.2 Software Requirements

* Operating System: Windows 7 or later, macOS, Linux
* Python: Version 3.6 or higher
* Tkinter: Installed with Python
* ttkthemes: Install using **pip install ttkthemes**
* forex-python: Install using **pip install forex-python**

## 3. Feasibility Study

### 3.1 Technical Feasibility

The project utilizes well-established technologies such as Python, Tkinter, and forex-python. These technologies are widely used, ensuring technical feasibility.

### 3.2 Operational Feasibility

The application's user-friendly interface makes it easy for users to perform currency conversions, enhancing operational feasibility.

### 3.3 Economic Feasibility

The project is cost-effective, utilizing open-source technologies. There are no licensing costs, contributing to economic feasibility.

## 4. Errors and Its Types

The Real-Time Currency Converter includes error handling to enhance robustness.

### 4.1 Types of Errors

* **Invalid Amount Error:** Raised when the entered amount is not a positive number.
* **Same Currency Error:** Raised when the source and target currencies are the same.
* **Unexpected Error:** Captures any unexpected errors during the conversion process.

## 5. Testing

The application underwent thorough testing to ensure functionality and error handling.

### 5.1 Unit Testing

Individual functions and components were tested independently.

### 5.2 Integration Testing

Testing of the entire system to ensure components work together.

### 5.3 User Acceptance Testing (UAT)

End-users performed testing to validate the application's usability.

## 6. Maintenance

The project is designed for ease of maintenance. Future updates or enhancements can be easily implemented.

## 7. Functions Used

The key functions used in the application include:

* **get\_currency\_list**: Retrieves the list of available currencies.
* **convert\_currency**: Handles the currency conversion logic.
* **get\_currency\_symbol**: Retrieves the currency symbol based on the currency code.

## 8. Flow Chart of Program

A simplified flowchart illustrating the program's execution is shown below:

[Flowchart Image]

## 9. Source Code

import tkinter as tk

from tkinter import ttk

from ttkthemes import ThemedStyle

from forex\_python.converter import CurrencyRates, CurrencyCodes

class CurrencyConverterApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Real-Time Currency Converter")

# Set window size and default currency values

self.root.geometry("800x600")

self.from\_currency\_var = tk.StringVar(value="INR")

self.to\_currency\_var = tk.StringVar(value="USD")

# Create GUI elements

self.create\_widgets()

def create\_widgets(self):

# Styling

style = ThemedStyle(self.root)

style.set\_theme("equilux")

# Customize the style for Label and Button widgets

style.configure('White.TLabel', foreground='white', font=('Helvetica', 18, 'bold'))

style.configure('White.TButton', foreground='white', font=('Helvetica', 14, 'bold'))

style.configure('Cursive.TLabel', font=('cursive', 14))

style.configure('MadeBy.TLabel', foreground='white', font=('Helvetica', 12))

# Main Frame

main\_frame = ttk.Frame(self.root)

main\_frame.pack(fill=tk.BOTH, expand=True, padx=20, pady=20)

# Title Label

title\_label = ttk.Label(main\_frame, text="Currency Converter", style='White.TLabel')

title\_label.pack(pady=20)

# Amount Entry

amount\_label = ttk.Label(main\_frame, text="Amount:", style='White.TLabel')

amount\_label.pack(pady=10)

self.amount\_entry = ttk.Entry(main\_frame, font=("cursive", 14))

self.amount\_entry.pack(pady=10)

# From Currency

from\_currency\_label = ttk.Label(main\_frame, text="From Currency:", style='Cursive.TLabel')

from\_currency\_label.pack(pady=10)

self.from\_currency\_combobox = ttk.Combobox(main\_frame, textvariable=self.from\_currency\_var, values=self.get\_currency\_list(),

font=("cursive", 14))

self.from\_currency\_combobox.pack(pady=10)

# To Currency

to\_currency\_label = ttk.Label(main\_frame, text="To Currency:", style='Cursive.TLabel')

to\_currency\_label.pack(pady=10)

self.to\_currency\_combobox = ttk.Combobox(main\_frame, textvariable=self.to\_currency\_var, values=self.get\_currency\_list(),

font=("cursive", 14))

self.to\_currency\_combobox.pack(pady=10)

# Convert Button

convert\_button = ttk.Button(main\_frame, text="Convert", style="TButton.Convert.TButton", command=self.convert\_currency)

convert\_button.pack(pady=20)

# Result Label

self.result\_var = tk.StringVar()

result\_label = ttk.Label(main\_frame, textvariable=self.result\_var, style='White.TLabel', wraplength=600)

result\_label.pack(pady=20)

# Made by Label

made\_by\_label = ttk.Label(main\_frame, text="Made by Jude Felix, Rahul D, and Jai Surya of class 12 B", style='MadeBy.TLabel')

made\_by\_label.pack(pady=20)

def get\_currency\_list(self):

# Add more currencies as needed

return ["USD", "EUR", "GBP", "INR", "JPY"]

def convert\_currency(self):

try:

amount = float(self.amount\_entry.get())

from\_currency = self.from\_currency\_var.get()

to\_currency = self.to\_currency\_var.get()

if amount <= 0:

raise ValueError("Amount should be a positive number.")

if from\_currency == to\_currency:

raise ValueError("Source and target currencies cannot be the same.")

c = CurrencyRates()

rate = c.get\_rate(from\_currency, to\_currency)

result = amount \* rate

self.result\_var.set(f"{amount:.2f} {self.get\_currency\_symbol(from\_currency)} = {result:.2f} {self.get\_currency\_symbol(to\_currency)}")

except ValueError as ve:

self.result\_var.set(f"Error: {ve}")

except Exception as e:

self.result\_var.set(f"Unexpected Error: {e}")

def get\_currency\_symbol(self, currency\_code):

# Add more currency symbols as needed

symbols = {"USD": "$", "EUR": "€", "GBP": "£", "INR": "₹", "JPY": "¥"}

return symbols.get(currency\_code, currency\_code)

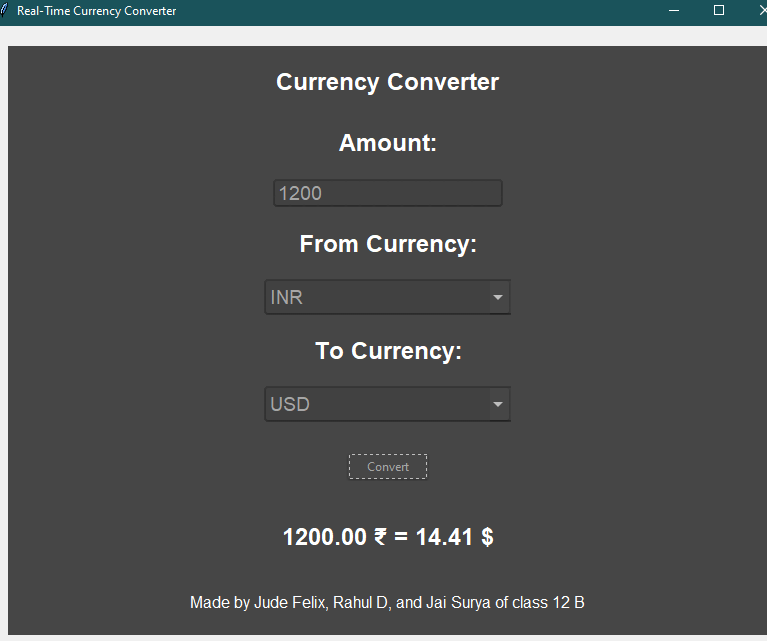
if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = CurrencyConverterApp(root)

root.mainloop()

## 10. Output



## 11. Conclusion

The Real-Time Currency Converter project successfully achieves its goal of providing users with an efficient and user-friendly tool for currency conversion. The inclusion of error handling, testing, and a well-structured codebase ensures a reliable and maintainable application.

## 12. Bibliography

* Python Documentation: https://docs.python.org/
* Tkinter Documentation: https://docs.python.org/3/library/tkinter.html
* ttkthemes Documentation: https://ttkthemes.readthedocs.io/en/latest/
* forex-python Documentation: https://forex-python.readthedocs.io/en/latest/