# Exchange Rates API Integration for Historical Analysis

Table of Contents

[**1.** **Overview** 2](#_Toc171414157)

[**2.** **Project Structure** 2](#_Toc171414158)

[**3.** **Approach** 2](#_Toc171414159)

[**4.** **Architecture** 3](#_Toc171414160)

[**5.** **Best Practices** 3](#_Toc171414161)

[**6.** **Source Files** 3](#_Toc171414162)

[**7.** **Additional Notes** 4](#_Toc171414163)

[**8.** **Output Result** 4](#_Toc171414164)

[**9.** **Test Result** 4](#_Toc171414165)

[**10.** **Code Snippets** 4](#_Toc171414166)

[i. Utilities 4](#_Toc171414167)

[ii. Project Level Configurations 6](#_Toc171414168)

[iii. Module Level Configurations 6](#_Toc171414169)

[iv. Main Scripts 7](#_Toc171414170)

# **Overview**

The project involves connecting to an exchange rates API to retrieve exchange rate data from Australia to New Zealand for the past 30 days in JSON format. The data is then preprocessed to handle potential issues like missing dates or values. A detailed analysis is conducted to identify the best (lowest) and worst (highest) exchange rates over the specified period, and to calculate the average exchange rate for the month. All project code, documentation, and relevant files are available in a public [GitHub](https://github.com/AsifSyedLive/cs_exchange_rate) repository.

# **Project Structure**

cs\_exchange\_rate-master/

├── config/

│ ├── config\_common.json

│ ├── config\_exchange\_rate\_analyze.json

│ ├── config\_exchange\_rate\_fetcher.json

│ └── config\_exchange\_rate\_preprocess.json

├── docs/

│ ├── Results/

│ ├── ProjectDoc\_ExchangeRates.docx

│ ├── TestDoc\_ExchangeRates.docx

├── exchange\_rate/

│ ├── \_\_init\_\_.py

│ ├── exchange\_rate\_analyze.py

│ ├── exchange\_rate\_fetcher.py

│ └── exchange\_rate\_preprocess.py

├── test/

│ ├── test\_exchange\_rate\_analyze.py

│ ├── test\_exchange\_rate\_fetcher.py

│ └── test\_exchange\_rate\_preprocess.py

├── utils/

│ ├── config\_loader.py

│ └── logger.py

├── .gitignore

├── README.md

├── main.py

├── requirements.txt

└── .env

# **Approach**

**Configuration Management:** The project utilizes JSON files to manage configurations for various components, including fetching, preprocessing, and analyzing data.

**Modularity:** The project is organized into modules, each dedicated to a specific task, enhancing reusability and maintainability.

**Logging:** Custom logging has been implemented to monitor progress and facilitate debugging.

**Standards Followed:** The project adheres to PEP 8 guidelines for Python coding, ensuring readability and consistency throughout the codebase.

# **Architecture**

**Configuration Management:** Responsible for loading and managing project configurations sourced from JSON files, ensuring flexibility and ease of configuration across different components such as data fetching, preprocessing, and analysis.

**Fetcher Module:** Designed to fetch exchange rate data reliably from external sources, ensuring the data is current and accurate for subsequent processing steps.

**Preprocess Module:** Handles the cleaning and initial preparation of fetched data, ensuring it is formatted correctly and ready for detailed analysis.

**Analyze Module:** Conducts comprehensive analysis on the preprocessed data, extracting meaningful insights and trends that contribute to the project's objectives.

**Logger:** Implements a custom logging mechanism to track project progress, monitor operations, and facilitate effective debugging when necessary, ensuring transparency and efficiency in development and deployment phases.

# **Best Practices**

**Modular Design:** The project employs a modular architecture to segregate functionalities, thereby promoting maintainability and clarity.

**Configuration Files:** Utilizes JSON-based configuration files to eliminate hardcoded values, ensuring flexibility and ease of configuration adjustments.

**Error Handling:** Incorporates robust error-handling mechanisms alongside logging capabilities to streamline debugging processes effectively.

**PEP 8 Compliance:** Ensures adherence to Python's PEP 8 style guide, fostering code cleanliness and readability throughout the project.

# **Source Files**

**main.py:** Entry point of the application.

**config\_loader.py:** Utility for loading configuration files.

**logger.py:** Custom logger implementation.

**exchange\_rate\_fetcher.py:** Module to fetch exchange rate data.

**exchange\_rate\_preprocess.py:** Module to preprocess exchange rate data.

**exchange\_rate\_analyze.py:** Module to analyze exchange rate data.

# **Additional Notes**

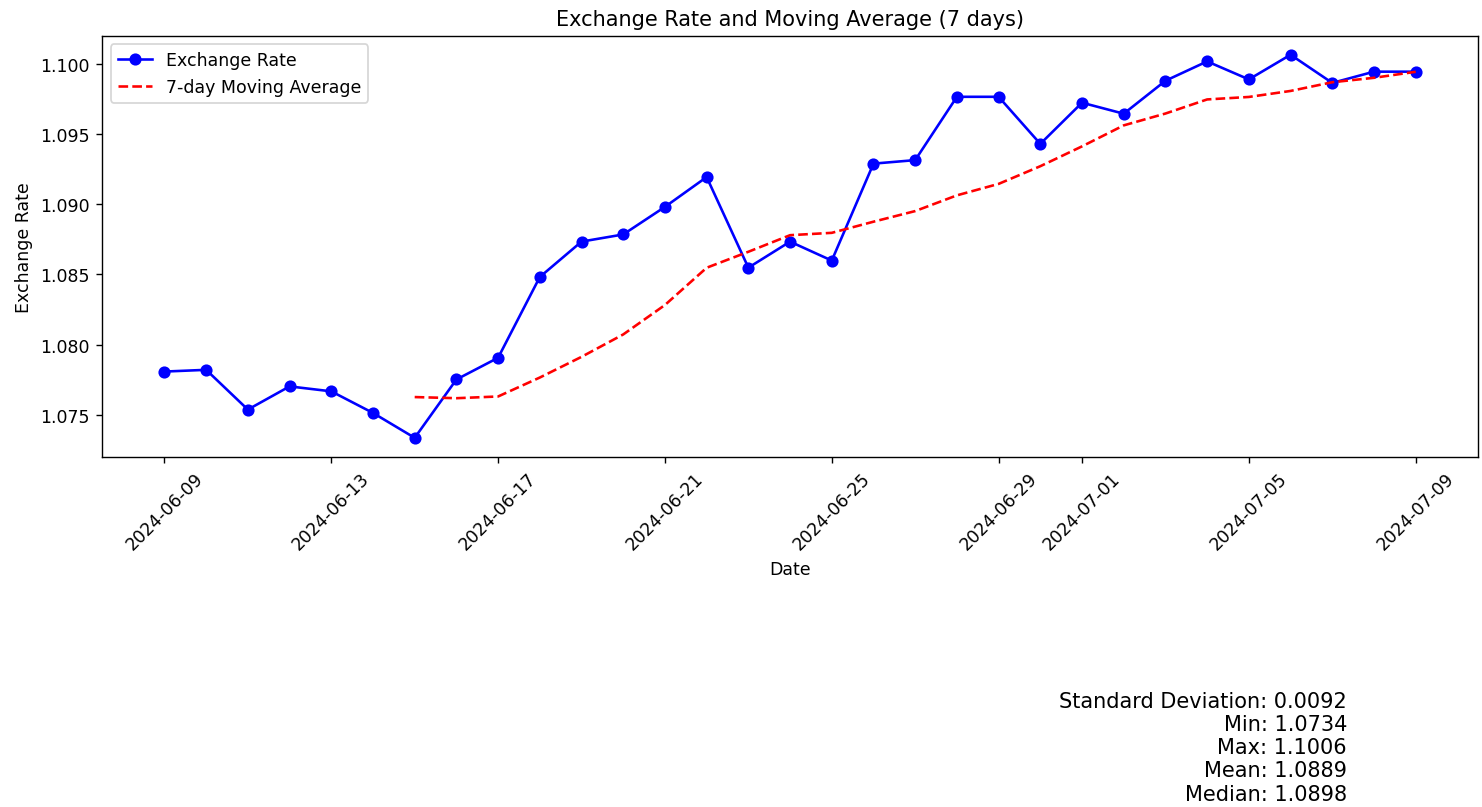
**README.md**: [README.md](https://github.com/AsifSyedLive/cs_exchange_rate/blob/master/README.md) Provides an overview and instructions on setting up and running the project.

**requirements.txt:** This [file](https://github.com/AsifSyedLive/cs_exchange_rate/blob/master/requirements.txt)Lists all the dependencies required to run the project

**Limitations:** Please refer Limitations section in [README.md](https://github.com/AsifSyedLive/cs_exchange_rate/blob/master/README.md)

**Future Enhancements:** Please refer Limitations section in [README.md](https://github.com/AsifSyedLive/cs_exchange_rate/blob/master/README.md)

# **Output Result**



# **Test Result**

Test Result document is uploaded to [GitHub](https://github.com/AsifSyedLive/cs_exchange_rate/blob/master/docs/TestDoc_ExchangeRates.docx)

# **Code Snippets**

Code is available in GitHub. Snippets of scripts surrounding core scripts are provided below for easier lookup along with links to main scripts.

## Utilities

* **Config Loader:** Generic script to load configurations

import os  
import json  
from dotenv import load\_dotenv  
from config import \*  
  
  
class ConfigLoader:  
 def \_\_init\_\_(self, env\_file='.env', common\_config\_file='config\_common.json', module\_config\_file=None):  
 self.env\_file = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '..'))+"/"+env\_file  
 self.common\_config\_file = os.path.join(os.path.dirname(\_\_file\_\_), '..', 'config', common\_config\_file)  
 if module\_config\_file:  
 self.module\_config\_file = os.path.join(os.path.dirname(\_\_file\_\_), '..', 'config', module\_config\_file)  
  
 self.env\_config = {}  
 self.common\_config = {}  
 self.module\_config = {}  
  
 self.load\_env\_config()  
 self.load\_json\_config(self.common\_config\_file, self.common\_config)  
 if module\_config\_file:  
 self.load\_json\_config(self.module\_config\_file, self.module\_config)  
  
 def load\_env\_config(self):  
 try:  
 load\_dotenv(self.env\_file)  
 self.env\_config = {  
 'BASE\_PATH': os.getenv('BASE\_PATH'),  
 'API\_KEY': os.getenv('API\_KEY'),  
 'LOG\_DIR': os.getenv('LOG\_DIR'),  
 'LOG\_FILE': os.getenv('LOG\_FILE')  
 }  
 except Exception as e:  
 print(f"Error loading environment variables: {e}")  
  
 def load\_json\_config(self, config\_file, config\_dict):  
 try:  
 with open(config\_file, 'r') as file:  
 config\_dict.update(json.load(file))  
 except FileNotFoundError:  
 print(f"Config file '{config\_file}' not found.")  
 except json.JSONDecodeError:  
 print(f"Error decoding JSON config file '{config\_file}'.")  
  
 def get\_env\_variables(self):  
 return self.env\_config  
  
 def get\_common\_config(self):  
 return self.common\_config  
  
 def get\_module\_config(self):  
 return self.module\_config

* **Logger:** Generic script to write to log file

import logging  
  
  
def setup\_logger(name, log\_file, level=logging.DEBUG):  
 formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(name)s - %(message)s')  
  
 handler = logging.FileHandler(log\_file)  
 handler.setFormatter(formatter)  
 handler.flush()  
  
 logger = logging.getLogger(name)  
 logger.setLevel(level)  
 logger.addHandler(handler)  
  
 return logger

## Project Level Configurations

* **.env:** Project Level and Senstive values can be stored here

PROJECT="cs\_exchange\_rate"  
BASE\_PATH="C:\\Users\\asifs\\${PROJECT}"  
LOG\_DIR="${BASE\_PATH}\\log"  
LOG\_FILE="${LOG\_DIR}\\exchange\_rate"  
API\_KEY='xxxxxxxxxxxxxxxxx'

## Module Level Configurations

* **config\_common.json:** Configuration to be available for all modules

{  
 "defaults\_exchange\_rate": {  
 "days": 30,  
 "base\_currency": "AUD",  
 "target\_currency": "NZD"  
 }  
}

* **config\_exchange\_rate\_analyze.json:** Configuration specific to module exchange\_rate\_analyze

{  
 "moving\_average": 7,  
 "fig\_width": 12,  
 "fig\_height": 10  
}

* **config\_exchange\_rate\_fetcher.json:** Configuration specific to module exchange\_rate\_fetcher

{  
 "api\_url": "https://api.exchangeratesapi.io/v1",  
 "end\_point": "timeseries",  
 "configure\_start\_date": "",  
 "configure\_end\_date":""  
}

* **config\_exchange\_rate\_preprocess.json:** Configuration specific to module exchange\_rate\_preprocess

{  
 "place\_holder\_var1": "place\_holder\_value2",  
 "place\_holder\_var2": "place\_holder\_value2"  
}

## Main Scripts

* [main.py](https://github.com/AsifSyedLive/cs_exchange_rate/blob/master/main.py) – Main Script for execution
* [exchange\_rate\_fetcher.py](https://github.com/AsifSyedLive/cs_exchange_rate/blob/master/exchange_rate/exchange_rate_fetcher.py) – Script to fetch exchange rates from API
* [exchange\_rate\_preprocess.py](https://github.com/AsifSyedLive/cs_exchange_rate/blob/master/exchange_rate/exchange_rate_preprocess.py)– Scripts to preprocess for data anomalies
* [exchange\_rate\_analyze.py](https://github.com/AsifSyedLive/cs_exchange_rate/blob/master/exchange_rate/exchange_rate_analyze.py) - Script to perform Analysis