**INVESTMENT PORTFOLIO TRACKER WITH RISK ANALYSIS**

**Problem Overview:**

Many individuals struggle to track their investments effectively, particularly across multiple asset classes (stocks, bonds, crypto, etc.). They also find it challenging to understand how risky their current portfolio is and how to rebalance their portfolio to meet their risk tolerance.

**Objective:**

Build an application that allows users to input their current portfolio (e.g., stocks, bonds, crypto, etc.), provides visualizations of portfolio performance over time, and offers basic risk analysis (e.g., portfolio variance, expected return) with suggestions on how to rebalance their portfolio based on their risk tolerance.

**Key Features of the App:**

1. **Portfolio Input and Tracking**:
   * Allow users to input their current investments, including asset classes, asset names, amounts, and purchase dates.
   * Connect to financial APIs (like Yahoo Finance, Alpha Vantage, or yfinance) to retrieve real-time asset prices.
   * Display portfolio value changes over time.
2. **Risk Analysis**:
   * Calculate and display metrics such as portfolio variance, expected return, Sharpe ratio, and other risk indicators.
   * Allow users to specify their risk tolerance and suggest how they can rebalance their portfolio to align with that tolerance.
3. **Visualizations**:
   * Create charts showing portfolio performance over time.
   * Show pie charts of asset allocations by category (e.g., stocks, bonds, crypto).
   * Display risk vs. return graphs based on portfolio historical performance.
4. **Rebalancing Suggestions**:
   * Provide suggestions for rebalancing the portfolio based on user preferences (e.g., moving assets from higher-risk categories to lower-risk categories).
   * Help users diversify their portfolio by showing suggestions for additional assets.

**Implementation Steps**

Here’s a basic breakdown of the app structure:

1. **User Interface (Optional)**
   * Use a CLI (Command Line Interface) or a simple web interface using something like Flask.
   * Allow users to input their portfolio, set risk tolerance, and view analysis and suggestions.
2. **Portfolio Data Management**
   * Allow users to input assets (stocks, bonds, etc.) manually or upload a CSV file with their portfolio.
   * Store portfolio data in a Pandas DataFrame for ease of analysis.
   * Implement an API call to a financial data service (like yfinance) to retrieve real-time asset prices and historical data.
3. **Portfolio Analysis**
   * Use historical price data to calculate the expected return, variance, and other key metrics.
   * Implement formulas for calculating risk metrics such as variance, covariance, and Sharpe ratio.
   * Analyze the portfolio’s risk and return characteristics.
4. **Suggestions Engine**
   * Create rules to generate rebalancing suggestions based on the user's risk tolerance (e.g., reducing exposure to high-volatility assets).
   * Suggest diversification strategies using basic Markowitz Modern Portfolio Theory.
5. **Visualization**
   * Use Matplotlib or Plotly to generate charts:
     + A line chart showing portfolio value over time.
     + A pie chart to show the percentage of the portfolio allocated to different asset classes.
     + A scatter plot showing risk vs. return for each asset in the portfolio.

**Example Code Snippet for Starting**

Here’s a very simple example of how you might start implementing this app.

import yfinance as yf

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Sample Portfolio

portfolio = {

'Assets': ['AAPL', 'MSFT', 'GOOGL', 'BTC-USD'],

'Quantity': [10, 5, 8, 0.1]

}

# Fetch Real-time Data

def fetch\_data(ticker):

data = yf.download(ticker, period='1y', interval='1d')

return data['Close']

def calculate\_portfolio\_value(portfolio):

portfolio\_value = 0

for asset, quantity in zip(portfolio['Assets'], portfolio['Quantity']):

price\_data = fetch\_data(asset)

latest\_price = price\_data.iloc[-1]

portfolio\_value += latest\_price \* quantity

return portfolio\_value

# Risk Analysis

def calculate\_expected\_return(prices):

log\_returns = np.log(prices / prices.shift(1))

return log\_returns.mean() \* 252

def calculate\_risk(prices):

log\_returns = np.log(prices / prices.shift(1))

return log\_returns.std() \* np.sqrt(252)

# Visualization

def plot\_portfolio\_value(asset, price\_data):

plt.figure(figsize=(10, 6))

plt.plot(price\_data.index, price\_data, label=asset)

plt.title(f'{asset} Price Over Time')

plt.xlabel('Date')

plt.ylabel('Price')

plt.legend()

plt.show()

# Example usage

for asset in portfolio['Assets']:

data = fetch\_data(asset)

plot\_portfolio\_value(asset, data)

print(f"Expected return for {asset}: {calculate\_expected\_return(data):.2f}")

print(f"Risk (volatility) for {asset}: {calculate\_risk(data):.2f}")

This code provides a framework for:

1. Fetching real-time and historical price data using yfinance.
2. Calculating portfolio value, expected returns, and risk.
3. Plotting data using Matplotlib.

**Potential Enhancements:**

* **Authentication and User Profiles**: Store user portfolios and preferences.
* **Advanced Analysis**: Incorporate more advanced financial metrics like Value at Risk (VaR), Monte Carlo simulations, or other predictive models.
* **Automated Alerts**: Notify users when their portfolio drifts beyond certain risk thresholds.
* **Integration**: Allow direct API connections to brokerage accounts (via OAuth) for automated tracking.

This app will give you practical experience in using financial data, making calculations related to risk and return, and visualizing data. The project touches on various Python libraries such as Pandas, NumPy, yfinance, and Matplotlib, providing a comprehensive practice experience with real-world applications.