# Application Documentation for app.py

**Overview**

This Flask application provides several API endpoints related to stock market data, user management, and Prometheus monitoring metrics. It integrates with third-party APIs such as Yahoo Finance and Alpha Vantage to retrieve stock-related data. Additionally, it provides system performance metrics using Prometheus and stores news data using SQLite.

The main components of the application include:

1. **Flask framework**: Serves the application and exposes RESTful API endpoints.
2. **SQLAlchemy**: Handles interactions with an SQLite database.
3. **Prometheus Client**: Provides monitoring capabilities.
4. **Alpha Vantage & Yahoo Finance**: External APIs used to fetch stock-related data.
5. **Cross-Origin Resource Sharing (CORS)**: Enabled to allow cross-origin requests.

**Dependencies**

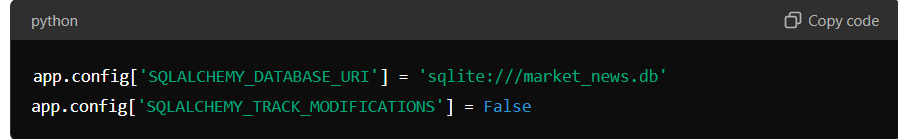
The application uses several external libraries to function. These dependencies should be included in the requirements.txt file.

* **Flask**: Web framework to create API endpoints.
* **Flask-CORS**: For enabling Cross-Origin Resource Sharing.
* **Flask-SQLAlchemy**: For integrating SQLAlchemy with Flask.
* **Prometheus Client**: For exposing metrics that Prometheus can scrape.
* **Yahoo Finance (yfinance)**: For fetching stock data.
* **Alpha Vantage**: For retrieving stock time series and financial data.
* **Pandas**: For handling data structures.
* **Requests**: For making HTTP requests.
* **psutil**: For fetching system resource metrics on Windows.

**Flask Application Configuration**

The application uses Flask as the framework and SQLAlchemy to interact with a local SQLite database.

**Flask Configuration**

* **CORS** is enabled to allow requests from any domain.
* **Database**: An SQLite database market\_news.db stores news articles and user data.
* 

**Prometheus Server**

A Prometheus server is started on port 8000 to expose application metrics.



**Database Models**

Two models are defined: News and User.

**News Model**

Represents news articles related to stocks.

* **Fields**:
  + id: Primary key
  + title: Title of the news article
  + content: Full content of the article
  + posted\_on: The date and time the article was posted (auto-generated)
  + tickers: Comma-separated list of stock tickers relevant to the article

A screen shot of a computer

Description automatically generated

**User Model**

Represents a user in the system.

* **Fields**:
  + id: Primary key
  + username: Unique username
  + password: Plain text password (for simplicity; this should be hashed in production)

A computer screen shot of a computer code

Description automatically generated

**API Endpoints**

**1. /metrics**

**Description: Exposes Prometheus metrics for the application.**

* **Method**: GET
* **Returns**: System, request, and custom application metrics in Prometheus format.

A black rectangular object with a black border

Description automatically generated**2. /stock/<ticker>**

**Description: Fetches stock data for a specific ticker.**

* **Method**: GET
* **URL Parameters**:
  + ticker: The stock ticker symbol (e.g., AAPL for Apple).
* **Response**:
  + JSON with the current stock price, historical prices (last month), and stock details (P/E ratio, market cap, etc.).
  + Fetches data from Yahoo Finance using the yfinance library.
  + Increments Prometheus counters for the number of requests and latency.

A computer screen with text

Description automatically generated**3. /top-stocks**

**Description: Fetches the latest stock prices for a predefined set of stocks.**

* **Method**: GET
* **Response**:
  + JSON object of stock symbols and their latest prices from Yahoo Finance.

A black screen with white text

Description automatically generated

**4. /top-stocks/historical**

**Description: Fetches historical stock data for a predefined set of stocks over the last 6 months.**

* **Method**: GET
* **Response**:
  + JSON containing historical prices for each ticker and the corresponding dates.

A computer screen with text

Description automatically generated

**Prometheus Metrics**

The application provides several custom metrics using Prometheus, including:

* **REQUEST\_COUNT**: Total number of requests received.
* **REQUEST\_LATENCY**: Measures request processing time.
* **ERROR\_COUNT**: Number of failed requests.
* **SUCCESS\_COUNT**: Number of successful requests.
* **TICKER\_REQUEST\_COUNT**: Number of requests per stock ticker.

**Custom System Metrics**

* **MEMORY\_RSS**: Resident Set Size memory usage (in bytes).
* **MEMORY\_VMS**: Virtual memory size (in bytes).
* **CPU\_USAGE**: CPU usage percentage.

These metrics are updated via the update\_system\_metrics function, which checks for platform-specific memory and CPU information.

**System Metrics**

**Function: update\_system\_metrics**

This function updates custom metrics for memory and CPU usage:

* **For Linux**: Memory usage is read from the /proc/<process\_id>/status file.
* **For Windows**: Memory and CPU usage are retrieved using the psutil library.

**Utilities**

**1. Database Initialization**

The init-db CLI command initializes the SQLite database, creating tables for the News and User models and seeding it with sample news data and a default user.

A black screen with white text

Description automatically generated

**2. Error Handling**

Each API endpoint wraps the main logic inside try-except blocks to catch errors and increment the ERROR\_COUNT metric in case of failures.

A black screen with a black rectangle

Description automatically generated

**Conclusion**

This Flask application integrates multiple features for stock data retrieval, monitoring with Prometheus, and simple database operations. It demonstrates how to expose custom metrics for Prometheus to monitor application and system performance. By utilizing third-party APIs like Yahoo Finance and Alpha Vantage, the app provides real-time stock data that can be consumed by frontend applications or other services.

Top of Form

Bottom of Form