Alerts Management Application

The Alerts Management Application is a Flask-based system designed to manage and store alerts securely. It offers functionalities to create alerts, to delete alerts, to get triggered alerts via a RESTful API.

Available Tokens

BTC, ETH, USDC, XRP, USDT, SUI, ICP, BNB, ADA, FIL

API used to get current price

- site: https://www.cryptocompare.com/
- API_KEY = 'a84341ca60b24f5f3107af219b1b52acd53cac0defbdf2527cd42d1442099f1b'
- url = https://min-api.cryptocompare.com/data/pricemulti?fsyms=BTC,ETH,USDC,XRP,USDT,SUI,ICP,BNB,ADA,FIL&tsyms=USD&api_key=a84341ca60b24f5f3107af219b1b52acd53cac0defbdf2527cd42d1442099f1b

Table of Contents

Introduction

The Alerts Management Application is a built using Python and Flask, designed to facilitate efficient alert-making functionalities. Leveraging MongoDB for data storage, it allows users to manage alerts through a RESTful API while ensuring data security.

Features

- Create new alerts to the database.

- Retrieve all alerts.
- Delete alerts from the database.
- Basic authentication to restrict access to certain endpoints.
- JWT token-based authentication.

Folder Structure

The application follows a well-organized folder structure:

```
Alerts_Management_API/
|-- app/
|   |-- __init__.py
|   |-- models.py
|   |-- routes.py
|   |-- auth.py
|   |-- alerts.py
|   |-- email_notifier.py
|-- config.py
|-- requirements.txt
|-- run.py
|-- Dockerfile
|-- docker-compose.yml
```

- **models.py**: Defines the schema for the User model, Alert model.
- **routes.py**: Handles API endpoints for authentication and alerts.
- **run.py**: Main entry point of the application.
- **config.py**: Stores configuration such as database credentials.

Installation

To set up the application locally, follow these steps:

1. Clone the repository: `git clone https://github.com/saikrishnayadav764/Alerts_Management_API.git`
2. Change Directory: `cd Alerts_Management_API-main`
3. Install dependencies: `pip install -r requirements.txt`
4. Configure variables: MONGO_URI = “mongodb+srv://naruto:naruto@cluster0.be644zi.mongodb.net/db”
5. In windows Install redis from this link(https://github.com/tporadowski/redis/releases/download/v5.0.14.1/Redis-x64-5.0.14.1.msi)
6. Start the server: `python run.py`
7. Access Your Application: http://localhost:5000

To set up the application in docker, follow these steps: 1. Build Docker Image: docker-compose build 2. Run Docker Containers: docker-compose up 3. Access Your Application: http://localhost:8080

Usage

1. The server starts at http://localhost:5000 by running `python run.py`. Once the server is running, you can access the defined API endpoints.
2. In docker the server starts at http://localhost:8080

API Endpoints

Alerts

- **POST /auth/login**: To get access token.
- **POST /auth/register**: To register new user.
- **POST /alerts/create**: creates a new alert.
- **GET /alerts**: Retrieves all alerts.
- **GET /alerts?status=created**: Retrieves all alerts with status created.
- **GET /alerts?status=triggered**: Retrieves all previous alerts with status triggered.
- **DELETE /alerts/**: Delete alert.
- **GET /alerts/check**: Shows triggered alerts at that particular time and notifies user through gmail, At the same it updates status to triggered.

Authentication

- **POST /api/login**: Authenticate users and generate JWT tokens.

Curl Commands

Execute these `curl` commands in your terminal to interact with the API endpoints:

Register User (Register)

```
curl -X POST -H "Content-Type: application/json" -d "{\"username\": \"harry\", \"password\": \"harry\", \"email\": \"harry@gmail.com\"}" https://nice-jade-coati-tie.cyclic.app/auth/register
```

Replace `username` and `password` with the credentials of the allowed users.

Obtain Token (Login)

```
curl -X POST -H "Content-Type: application/json" -d "{\"username\": \"admin\", \"password\": \"admin\"}" https://nice-jade-coati-tie.cyclic.app/auth/login
```

Replace `username` and `password` with the credentials of the allowed users.

Creating a new alert

```
curl -X POST -H "Content-Type: application/json" -H "Authorization: Bearer <access_token>" -d "{\"target_price\": \"3000\", \"crypto_symbol\": \"ETH\"}" https://nice-jade-coati-tie.cyclic.app/alerts/create
```

Retrieving all alerts

```
curl -X GET -H "Authorization: Bearer <access_token>" https://nice-jade-coati-tie.cyclic.app/alerts/
```

Deleting a alert

```
curl -X DELETE -H "Authorization: Bearer <access_token>" https://nice-jade-coati-tie.cyclic.app/alerts/:alert_id
```

Replace `:alert_id` with the actual alert_id of the alert you want to delete.

Importing the Postman Collection

To import the Alerts Management API collection into Postman, follow these steps:

1. Download the Postman collection file by clicking here.
2. Open Postman.
3. Click on the "Import" button in the top left corner of the window.
4. In the dialog that appears, click on the "Upload Files" tab.
5. Select the downloaded collection file (`.json` format) from your computer.
6. Once the file is uploaded, click on the "Import" button to import the collection into Postman.

After importing the collection, you'll be able to see all the API endpoints along with example requests and responses. You can then use these requests to interact with the Alerts Management API directly from Postman.

Existing Users

The application has the following default users to authenticate:

- username: admin, password: admin

Technologies Used

- Python 3.x
- flask==3.0.1
- flask_jwt_extended==4.6.0
- Flask-Caching==2.1.0
- redis==5.0.1
- flask_pymongo==2.3.0
- Werkzeug==3.0.1
- requests==2.31.0
- uuid==1.30

Deployed Link

[https://nice-jade-coati-tie.cyclic.app/alerts]