Sensibull Backend Developer Programming Challenge

Problem Statement

- Assume that you are integrating with a stock exchange (sensibull). Write a service that allows a user to execute orders and get the status.
- The way this exchange works is that once you place an order, the exchange will either complete or error the order. This could take from a few seconds to a few minutes. Once the order is in complete or error state, no further changes are made to the order.
- There is 2 sets of APIs
    - First provided by sensibull, which your server communicates with. These sensibull APIs that allow your server to place orders and check the status. The exact APIs and parameters are mentioned in the **'Input Api Details'** section below
    - Second set of APIs are the APIs we expect you to create. The exact APIs and parameters are mentioned in the **'Output API Requirements'** section below

Requirements

- Create a http server that provides 4 api endpoints. The exact APIs parameters required are mentioned in **'Output API Requirements'** section below.
    - Place Order
        - This api takes a symbol and quantity and places an order with sensibull using the provided APIs
    - Modify Order
        - This api takes the order identifier and quantity and updates an existing order by hitting senisbull's APIs
        - Only open orders can be modified. If the order state is **not open** in your database, **do not forward** the request to sensibull APIs
    - Cancel Order
        - This api takes the order identifier and cancels an existing order
        - Only open orders can be cancelled. If the order state is **not open** in your database, **do not forward** the request to sensibull APIs
    - Order Status
        - This api must return the order status from your database. **Do not hit** the sensibull APIs for this request

- All orders placed / statuses must be saved in some database
- Have a background process or job that polls the sensibull orders status api every 15s for all **open orders** and **saves** the updates

Output API Requirements

This service must expose the following endpoints

Place Order

Endpoint: `/order-service`

Method: `POST`

Request JSON Body: symbol and quantity `{ "symbol": "HDFC", quantity: 10 }`

Response: Returns the order details on success

Sample Response Expected:

```
{
    "success": true,
    "payload": {
        "identifier": "xxxxx",
        "symbol": "HDFC",
        "quantity": 10,
        "filled_quantity": 0,
        "order_status": "open"
    }
}
```

Modify Order

Endpoint: `/order-service`

Method: `PUT`

Request JSON Body: Order id and new quantity `{ "identifier": "xxxx", new_quantity: 20 }`

Response: Returns the order details on success

Sample Response Expected:

```
{
    "success": true,
    "payload": {
        "identifier": "xxxx",
        "symbol": "HDFC",
        "quantity": 20,
        "filled_quantity": 8,
        "order_status": "open"
    }
}
```

Cancel Order

Endpoint: `/order-service`

Method: `DELETE`

Request JSON Body: Order id `{ "identifier": "xxxx" }`

Response: Returns the order details on success

Sample Response Expected:

```
{
    "success": true,
    "payload": {
        "identifier": "xxxxxxx",
        "symbol": "HDFC",
        "quantity": 5,
        "filled_quantity": 0,
        "order_status": "cancel"
    }
}
```

Order Status

Endpoint: `/order-service/status`

Method: `POST`

Request JSON Body: Order id `{ "identifier": "xxxx" }`

Response: Returns the order details on success

Sample Response Expected:

```
{
    "success": true,
    "payload": {
        "identifier": "xxxxxxxx",
        "symbol": "HDFC",
        "quantity": 5,
        "filled_quantity": 0,
        "order_status": "open"
    }
}
```

Input API Details

The api's listed below are provided by sensibull.

Common parameter details

Order Status

Sensisbull server sends order status string value. This value is oneof **"open", "complete", "error", "cancel"**

- When a order is placed, it starts in "open" state.
- When order is completely filled, it goes into "complete" state.
- If orders is cancelled, status becomes "cancel"
- If any error occurs, then order goes into "error" state

Headers

All api requests must have the "X-AUTH-TOKEN" header. This must be a random string of atleast 12 characters (for example a uuid). This is used to identify the client. All api requests from your server **must use the same value** for this header. You can generate this on server startup or hardcode this value. (In actual production, this will be some auth token used to validate the server to server communication, but for this exercise, a unique string should be enough)

Payload

The success api has the form

```
{
    "success": true,
    "payload": {
        "order": {
            "order_id": "3896f5f8-2258-412e-b42f-71b9d1351121",
            "order_tag": "yyyyyy",
            "symbol": "HDFC",
            "request_quantity": 200,
            "filled_quantity": 0,
            "status": "open"
        },
        "message": "order create success"
    }
}
```

Parameters of the order are:

order_id: UUID which uniquely identifies an order

order_tag: The string value which returned as is as given in the place order api

symbol: The symbol string as given in the place order api

requested_quantity: The quantity as given in the place or modify api

filled_quantity: The total orders filled so far (This parameter is not relevant for this exercise, just save the value as is)

status: One of the order status strings as mentioned above in order status section

On error, reponse will be the following with the appropriate http status code

```
{
    "success": false,
    "err_msg": "some error occured"
}
```

Sensibull Order Placement

(NOTE: This API is provided by Sensibull)

Method: `POST`

Headers: `X-AUTH-TOKEN`

Endpoint: `https://prototype.sbulltech.com/api/order/place`

Request JSON Body:

symbol, quantity and order tag `{ "symbol": "HDFC", quantity: 10, order_tag: "yyyyyy" }`

symbol: Must be a string of atleast 3 characters. For example, 'BAJAJFIN', 'HDFC', 'SBIN'

quantity: Must be a number > 0

order_tag: Must be string. This value is returned as is on all api request for a particular order

Response: On success, orders details are returned

Response

```
{
    "success": true,
    "payload": {
        "order": {
            "order_id": "3896f5f8-2258-412e-b42f-71b9d1351121",
            "order_tag": "yyyyyy",
            "symbol": "HDFC",
            "request_quantity": 200,
            "filled_quantity": 0,
            "status": "open"
        },
        "message": "order create success"
    }
}
```

Sensibull Order Modification

(NOTE: This API is provided by Sensibull)

Method: `PUT`

Headers: `X-AUTH-TOKEN`

Endpoint: `https://prototype.sbulltech.com/api/order/{orderId}`

Request JSON Body: the new quantity `{ quantity: 10}`

Response: On success, orders details are returned

Response

```json
{
    "success": true,
    "payload": {
        "order": {
            "order_id": "3896f5f8-2258-412e-b42f-71b9d1351121",
            "order_tag": "yyyyyyyy",
            "symbol": "HDFC",
            "request_quantity": 10,
            "filled_quantity": 0,
            "status": "open"
        },
        "message": "order update success"
    }
}
```

Sensibull Order Cancellation

(NOTE: This API is provided by Sensibull)

Method: `DELETE`

Headers: `X-AUTH-TOKEN`

Endpoint: `https://prototype.sbulltech.com/api/order/{orderId}`

Response: On success, orders details are returned

Response

```json
{
    "success": true,
    "payload": {
        "order": {
            "order_id": "82635268-9ef7-4a37-87e6-af3f18bd519d",
            "order_tag": "yyyyyy",
            "symbol": "HDFC",
            "request_quantity": 200,
            "filled_quantity": 169,
            "status": "cancel"
        },
        "message": "order cancel success"
    }
}
```

Sensibull Order Status

(NOTE: This API is provided by Sensibull)

Method: `POST`

Headers: `X-AUTH-TOKEN`

Endpoint: `https://prototype.sbulltech.com/api/order/status-for-ids`

Request JSON Body: the list of order ids

`{ "order_ids": ["27a3e4ca-f54a-41f6-8878-0a02003f044d", "82635268-9ef7-4a37-87e6-af3f18bd519d"] }`

Response: On success, orders details are returned

Response

```
{
    "success": true,
    "payload": [
        {
            "order_id": "27a3e4ca-f54a-41f6-8878-0a02003f044d",
            "order_tag": "yyyyyy",
            "symbol": "HDFC",
            "request_quantity": 200,
            "filled_quantity": 194,
            "status": "open"
        },
        {
            "order_id": "82635268-9ef7-4a37-87e6-af3f18bd519d",
            "order_tag": "zzzzzzz",
            "symbol": "ACC",
            "request_quantity": 200,
            "filled_quantity": 169,
            "status": "cancel"
        }
    ]
}
```

General Instructions

The submission will be evaluated for:

- Correctness
- Code Readability
  - Please write well formatted, readable code with appropriate variable naming.
  - Please ensure that the coding conventions, directory structure and build approach to your project follow the common conventions.

Technical Requirements

- You can implement the API in any language of your choice, in a framework of your choice.
- Dockerized Solution:
  - The API server you use to deploy this service and the RDBMS used to store data needs to be consolidated into a `docker-compose.yml` file so that the entire service assembly can be brought up and run via `docker-compose up`.
  - Please run any DB migrations in a container within this `docker-compose` so that they run automatically at first start.
- Your API server must be accessible on port `19093`
  - This is so that our automated test scripts can consume the API and run test cases.
- No authentication is required for these APIs

Submission

- Please create a `tar.gz` file containing a git repo of the solution and upload it to a shared Google Drive/DropBox folder, and share that link with us.
- Use Git for version control.
  - We would like to see the Git metadata (the .git folder) in the tarball so we can look at your commit logs and understand how your solution evolved.
  - Frequent commits are a big plus.
- Please make sure that the code has the necessary instructions to build the app.
- Please include a file detailing any design / architecture and any potential improvement if needed.
- Do not include library folders like `node_modules` for Nodejs or `target` for rust code in the `.zip` or `.tar.gz` file
- Do not make either your solution or this problem statement publicly available by, for example, using GitHub, Gitlab or Bitbucket or by posting this problem to a blog or forum.