Background:

Every day prices of agriculture commodities change across different mandis/markets in the country. On Gramoday App, daily prices are reported by users whose occupation is mandi commission agent. Each mandi commission agent maps to one market commodity combination for which he creates a daily price report.

Prices can be reported across multiple units like Kg, Quintal, Bag, etc. with each unit having a conversion factor in terms of base unit (kg).

One market commodity combination can have a daily price report contributed by 2 users, in which case, an aggregate (average of values) report needs to be created which is available for consumption on the platform.

Problem Statement:

You are required to build an express JS API web-service which captures user contributed reports and returns an aggregate report in response.

Each report consists of a market-commodity combination for which prices in the *Mandi(Market)* are provided in a certain unit (along with their conversion factor to base unit - Kg).
You need to combine the reports per market-commodity by calculating the average of the report prices.

The algorithm to generate the aggregate report is as below:

1. Look for an existing report with marketID-cmdtyID in the DB [1].
2. Convert the prices into base price based on the base unit [2]
3. Calculate the mean of prices.
4. Save the aggregated report with price per Kg.
5. Return the reportID of the generated report.

You need to implement the below 2 APIs:

1. Send a create report request.
Below are 2 requests sent for the same market-commodity

Request-1:

```
POST /reports
{
  "reportDetails": {
    "userID": "user-1",
    "marketID": "market-1",
    "marketName": "Vashi Navi Mumbai",
```

```
    "cmdtyID": "cmdty-1",
    "marketType": "Mandi",
    "cmdtyName": "Potato",
    "priceUnit": "Pack",
    "convFctr": 50,
    "price": 700
  }
}
```

Response-1:

```
{
      status: "success",
      reportID: "949832f8-48c7-4cb2-8dcd-98f046a9a2e3"
}
```

Request-2:

```
POST /reports
{
    "reportDetails": {
      "userID": "user-2",
      "marketID": "market-1",
      "marketName": "Vashi Navi Mumbai",
      "cmdtyID": "cmdty-1",
      "cmdtyName": "Potato",
      "priceUnit": "Quintal",
      "convFctr": 100,
      "price": 1600
    }
}
```

Response-2:

```
{
      status: "success",
      reportID: "949832f8-48c7-4cb2-8dcd-98f046a9a2e3"
}
```

2. Get the aggregated report

Request

```
GET /reports?reportID=949832f8-48c7-4cb2-8dcd-98f046a9a2e3
```

Response:

```
{
  "_id": "949832f8-48c7-4cb2-8dcd-98f046a9a2e3",
  "cmdtyName": "Potato",
```

```
    "cmdtyID": "VE-42",
    "marketID": "market-1",
    "marketName": "Vashi Navi Mumbai",
    "users": ["user-1", "user-2"],
    "timestamp": 1616198400000,
    "priceUnit": "Kg",
    "price": 15
}
```

[1] You can use any database - relational/non-relational for this task as long as the API response is as required.
[2] You can assume string values for *user, market,* or *cmdty* entities for this problem
[3] **`convFctr`** converts the **`priceUnit`** into the base unit of Kg

Bonus Task:
Write an API test to capture the above scenario and verify that the `/reports POST and /reports GET` API returns the correct response.

Output:
Please create a repository on github and push your source code there. Provide the instructions to run the app in a README.md and add details that might be needed to verify that your code runs.

Please share details on your github repository (access link etc) with admin@agrilinks.in by 21st April 10am.