COMP3322 Modern Technologies on World Wide Web

Assignment Four

Total 12 points

Deadline: 23:59 April 29, 2024

Overview

Write an express.js program called **index.js**, which provides the APIs for users to access data from a MongoDB server storing monthly statistics on passengers who have travelled across Hong Kong control points.

Objectives

1. A learning activity to support ILO 1 and ILO 2.
2. To practice using Node, Express, MongoDB, and Mongoose to create a simple REST API.

Specification

Assume you are using the MongoDB server running on **the course's node.js docker container** with the service name ***mongodb*** listening to the default port 27017.

The "**HKPassFlow**" database is a collection of monthly statistics on passenger travel across control points in Hong Kong. The data is sourced from the Hong Kong Government Immigration Department and covers daily inbound and outbound trips since 2021. The data is categorized by type of traveller, including Hong Kong Residents, Mainland Visitors, and Other Visitors.

https://data.gov.hk/en-data/dataset/hk-immd-set5-statistics-daily-passenger-traffic

The monthly data in the "HKPassFlow" database is generated by a data analytics program. This data is stored in a collection called "**monthlog**" and each document within the collection contains seven fields: _id, Year, Month, Local, Mainland, Others, and Total.

Year	Month	Local	Mainland	Others	Total
2021	1	-27561	-1788	-650	-29999
2021	2	30690	1701	-373	32018
2021	3	11913	1179	-311	12781
2021	4	729	473	-328	874
2021	5	-7436	-45	-469	-7950
2021	6	-9139	-90	-389	-9618
2021	7	-36736	-1686	-156	-38578
	⋮			⋮	
2023	3	-173144	7078	2744	-163322
2023	4	-62206	117839	15015	70648
2023	5	149674	-115064	-15796	18814
2023	6	-196818	22308	14393	-160117
2023	7	-74082	42460	-5692	-37314

[Note: a negative value in the dataset indicates that more people departed Hong Kong than arrived, and vice versa. For instance, in March 2023, the data shows that there was a total of 163322 more departures from Hong Kong than arrivals.]

To test your API, we provide the monthly dataset that contains monthly data from Jan 2021 to Dec 2023. You can download the dataset (**monthlog.csv**) from the course's Moodle site. **Import the data to the HKPassFlow database for the tests.**

You will be using the provided framework for developing your program. You can download the template file (template.txt) from the course's Moodle site.

index.js

```
const express = require('express')
const app = express();




     /* Implement the logic here */











app.listen(3000, () => {
  console.log('App listening on port 3000!')
});
```

Task A

Use the command **mongoimport** to import the CSV file to the MongoDB server. Here are the steps to import the data to your docker's mongodb server.

1. Use Windows Explorer or Mac Finder to go to the data/db folder (which is inside the Node-dev folder).
2. Copy the monthlog.csv file there.
3. Access the docker desktop and open a terminal for the **c3322-mongo container**.
4. In the terminal, type this command (in one line):

```
mongoimport -d=HKPassFlow -c=monthlog --type=csv --headerline --file=monthlog.csv
```

Write the code to set up a connection to the MongoDB server **using Mongoose**.
Write the code in your Express app to (i) set up a connection to the MongoDB database using **Mongoose** for accessing the data, and (ii) set up a schema and a model for accessing the "monthlog"

collection. Write the code that **monitors** the database connection and **terminates** the program if the connection to the database **is lost**.

Task B

Write an endpoint to handle **the GET request** for retrieving

(1) the statistics of the passenger traffic for a specific month and year, or
(2) the summary statistics for a particular year.

The returned data should be a **JSON string**. For example, to retrieve the traffic statistics for June 2023, the client sends the GET request to `http://localhost:3000/HK/stat/2023/6 or http://localhost:3000/HK/stat/2023/06.` The endpoint returns the following JSON string:
{"Year":2023,"Month":6,"Local":-196818,"Mainland":22308,"Others":14393,"Total":-160117}

When the client sends the GET request to `http://localhost:3000/HK/stat/2023`. The function retrieves all the records in 2023 and then adds up all the values in each field to generate the summary value for that field. The endpoint returns the following JSON string:
{"Year":"2023","Local":-385096,"Mainland":171574,"Others":96525,"Total":-116997}.

This endpoint performs the following actions:

- Only respond to GET requests.
- For retrieving the statistics on a specific month and year, connect to the database to retrieve all fields (**except the _id field**) in the collection.
- For retrieving the summary statistics for a specific year, connect to the database to retrieve all documents in that year and add all the values in each field to form the aggregated value for that field.
- Send the data as a single JSON string to the client.

This endpoint performs the following checking:

- It only handles requests between the years 2021 to 2025. If a request carries a Year value not in this range, return an error JSON string with HTTP status code **400**. If a request carries a Month value not in the range between 1 to 12, return an error JSON string with HTTP status code **400**. In addition, we can combine the two checks. For example, for a request with the path `/HK/stat/2008/14`, the program gives this response:
  `{"error":"Wrong year - must be between 2021 - 2025. Wrong month."}` with status code 400.
- If the request carries a year-month (within 2021 - 2025) that the database does not have the document, return an error JSON string with the HTTP status code **404**. For example, in a request asking for the data on 2024-05, the program gives this response:
  `{"error":"No data for 5/2024"}`

Task C

Write an endpoint to handle **the POST request** for adding a new record to the MongoDB database. To add the new record, the client sends a POST request to the URL
`http://localhost:3000/HK/stat/`

The request includes a JSON string in the message body that contains the **arrival and departure** data for a specific month and year. For example, below is a JSON string for Jan 2024.

```
[
 {
   "Year": 2024, "Month": 1, "Flow": "Arrival", "Local": 7918911, "Mainland":
2983542, "Others": 842354
 },
 {
   "Year": 2024, "Month": 1, "Flow": "Departure", "Local": 7468171, "Mainland":
3023476, "Others": 908338
 }
]
```

You can assume that when the client provides a JSON string, it always contains the arrival and departure data. However, they are not always in the same order in the JSON string. For example, below is a JSON string for Feb 2024 and the order is different.

```
[
 {
   "Year": 2024, "Month": 2, "Flow": "Departure", "Local": 7531843, "Mainland":
3301280, "Others": 724978
 },
 {
   "Year": 2024, "Month": 2, "Flow": "Arrival", "Local": 7491230, "Mainland":
3244855, "Others": 756638
 }
]
```

This endpoint performs the following actions:

- Only response to POST requests.
- If no existing matched year-month document is in the database, the program should **compute the statistics** for that specific year-month before inserting the record.
  To compute the statistics for the "Local", "Mainland", and "Others" fields, the program has the arrival data **minus** the departure data for each field. To calculate the "Total" field, the program first sums the arrival data and the departure data, then it has the total arrival minus total departure to get the total statistics. For example, here are the statistics for 2024-3 record:

	Arrival	Departure	
Local	8374047	9291362	-917315
Mainland	2466042	2489151	-23109
Others	936189	946946	-10757
Total	11776278	12727459	-951181

- Add the new document (with all fields) into the database.
- If successfully inserted the record, the program sends the new record to the client in JSON format. For example, this is the response for adding 2024-3 data to the database.
  `{"Year":2024,"Month":3,"Local":-917315,"Mainland":-23109,"Others":-10757, "Total":-951181}`

This endpoint performs the following checking:

- If the POST request does not carry data, the program should send an error JSON string with the HTTP status code **400**. For example, `{"error":"POST request - missing data."}`.

- If there exists a matched year-month record in the database already, the program should send an error JSON string with the HTTP status code **409**. For example, in a request to add the data for 2023-8, the program gives this response:
  `{"error":"Record exists for 8/2023; cannot overwrite."}`

To test this endpoint, we can use the Curl command to send a POST request with the JSON data to the Express server. You can download two files from the course's Moodle site for testing. curl-cmd.txt – contains sample curl commands and monthflow.csv – contains the data for 2024-1, 2024-2, and 2024-3.

TASK D

Write an endpoint to handle **the GET request** for retrieving the monthly statistics of different groups of passenger traffic for a specific year. To retrieve the monthly statistics for local residents in 2023, the client sends the GET request to `http://localhost:3000/HK/stat/2023/local.` To retrieve the monthly statistics for mainland visitors in 2022, the client sends the GET request to `http://localhost:3000/HK/stat/2022/mainland.` To retrieve the monthly statistics for other visitors in 2021, the client sends the GET request to `http://localhost:3000/HK/stat/2021/others.`

The returned data should be a **JSON string** and ordered by month. Here is the returned JSON string for the request `/HK/stat/2023/mainland`:

```
[{"Month":1,"Mainland":11988},{"Month":2,"Mainland":34161},{"Month":3,"Mainland":7078},{"Month":4,"Mainland":117839},{"Month":5,"Mainland":-115064},{"Month":6,"Mainland":22308},{"Month":7,"Mainland":42460},{"Month":8,"Mainland":-67134},{"Month":9,"Mainland":109507},{"Month":10,"Mainland":-91595},{"Month":11,"Mainland":6670},{"Month":12,"Mainland":93356}]
```

Here is another example JSON string for the request /HK/stat/2021/others:

```
[{"Month":1,"Others":-650},{"Month":2,"Others":-373},{"Month":3,"Others":-311},{"Month":4,"Others":-328},{"Month":5,"Others":-469},{"Month":6,"Others":-389},{"Month":7,"Others":-156},{"Month":8,"Others":180},{"Month":9,"Others":110},{"Month":10,"Others":314},{"Month":11,"Others":296},{"Month":12,"Others":-39}]
```

This endpoint performs the following actions:

- Only respond to GET requests.
- For retrieving the statistics of a specific group for a particular year, retrieve all documents pertaining to that year from the database. Only the Month and target group fields should be returned.
- Send the data as a single JSON string to the client.

This endpoint performs the following checking:

- It only handles requests between the years 2021 to 2025. If a request carries a Year value not in this range, return an error JSON string with HTTP status code **400**. For example, `{"error":"Wrong year - must be between 2021 - 2025."}` with status code 400.

- If the request specifies a year (within 2021 - 2025) that the database does not have the documents, return an error JSON string with the HTTP status code **404**. For example, in a request asking for data on 2025, the program gives this response:
  `{"error":"No data for 2025"}`

Task E

Write **a routing endpoint** to intercept all other request types and paths, which are not defined in previous tasks. Return a JSON string with the HTTP status code **400**. For example, for the request `POST /HK/stat/2023/10`, we get the response `{"error":"Cannot POST /HK/stat/2023/10"}`; for the request `GET /HK/stat/`, we get the response `'{"error":"Cannot GET /HK/stat/"}`.

Resources

You are provided with the following files.

- `template.txt` – the framework for the index.js file.
- `monthlog.csv` – the dataset between 2021 Jan – 2023 Dec.
- `curl-cmd.txt` – sample curl commands
- `monthflow.csv` – contains 2024 Jan, 2024 Feb, and 2024 Mar data.

Testing platform

We shall run the server program in the node-dev container set and use Curl and Firefox to test the API.

Submission

Please finish this assignment before 23:59 on April 29, 2024 Monday. Submit the following files:

1. A JSON file – use mongoexport to export the whole collection from the HKPassFlow database.
   Similar to the mongoimport command, you have to open a terminal at the data/db folder and type the following command (in one line):

```
mongoexport -d=HKPassFlow -c=monthlog --jsonArray --sort='{_id: 1}' --out=3035111999.json
```

   Replace 3035111999 with your student ID and upload this JSON file.
2. The complete index.js program.
3. The package.json file of your express program.

Grading Policy

Points	Criteria
2.0	Task A ▪ Database set up, import the dataset, and export the collection. ▪ The program can connect and access the MongoDB database. ▪ The program can detect that the database connection is broken.
3.0	Task B ▪ Correctly handle the GET request to retrieve the passenger traffic statistics for a specific year and month and return a JSON string in the required format. ▪ Correctly handle the GET request to generate the summary passenger traffic statistics for a specific year and return a JSON string in the required format. ▪ Error handling
3.0	Task C

	▪ Correctly handle the POST request to insert a new document into the collection for a specific year and month and return a JSON string in the required format. ▪ Error handling
3.0	Task D ▪ Correctly handle the GET request to retrieve the monthly traffic statistics of a specific group of people for a particular year and return a JSON string in the required format. ▪ Error handling
1.0	Task F ▪ Error handling of all unknown methods and paths
-4.0	Using any external libraries.

Plagiarism

Plagiarism is a very serious academic offence. Students should understand what constitutes plagiarism, the consequences of committing an offence of plagiarism, and how to avoid it. ***Please note that we may request you to explain to us how your program is functioning as well as we may also make use of software tools to detect software plagiarism.***