Student Name: Tay Kai Xiang Matriculation Number: A0200236Y

Link to GitHub repository: https://github.com/kaixiangtay/CS3219_OTOT_F

Pre-requisites (Software to be installed on local device):
Postman : https://www.postman.com/downloads/
NodeJS and npm: https://docs.npmjs.com/downloading-and-installing-node-js-and-npm
MongoDB: https://docs.mongodb.com/manual/installation/
Redis: https://redis.io/download

References
Mock Data: http://jsonplaceholder.typicode.com/
Redis Crash course: https://www.youtube.com/watch?v=jgpVdJB2sKQ

Instructions
Step 1: Clone the project repository **CS3219_OTOT_F** into the desired directory of local device.

Step 2: Open up the terminal /console window and go to the directory in Step 1.

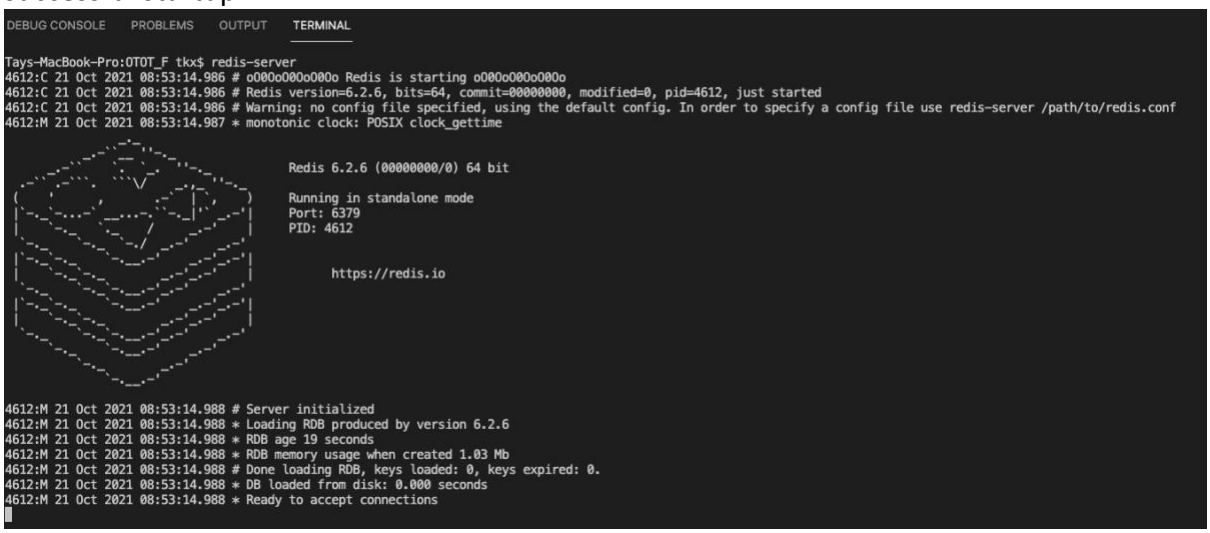
Step 3: Install the node application dependencies as specify in the package.json file required using the following command:

npm install

Step 4: Open up 2 additional console windows and execute the following commands to start up redis server and redis cli respectively:

redis-server
redis-cli

Step 5: You should see a similar response on the redis-server console window upon successful startup.



Step 6: To verify our redis server connection in redis-cli, we can do a ping command and if successful a PONG command will be returned.

```
DEBUG CONSOLE    PROBLEMS    OUTPUT    TERMINAL

not connected> ping
PONG
127.0.0.1:6379>
```

Step 7: Then, inside the first terminal window, perform the following command to start up the application:

npm start

Step 8: You should a similar response in the console where our 5000 photos have been saved into the MongoDb database.

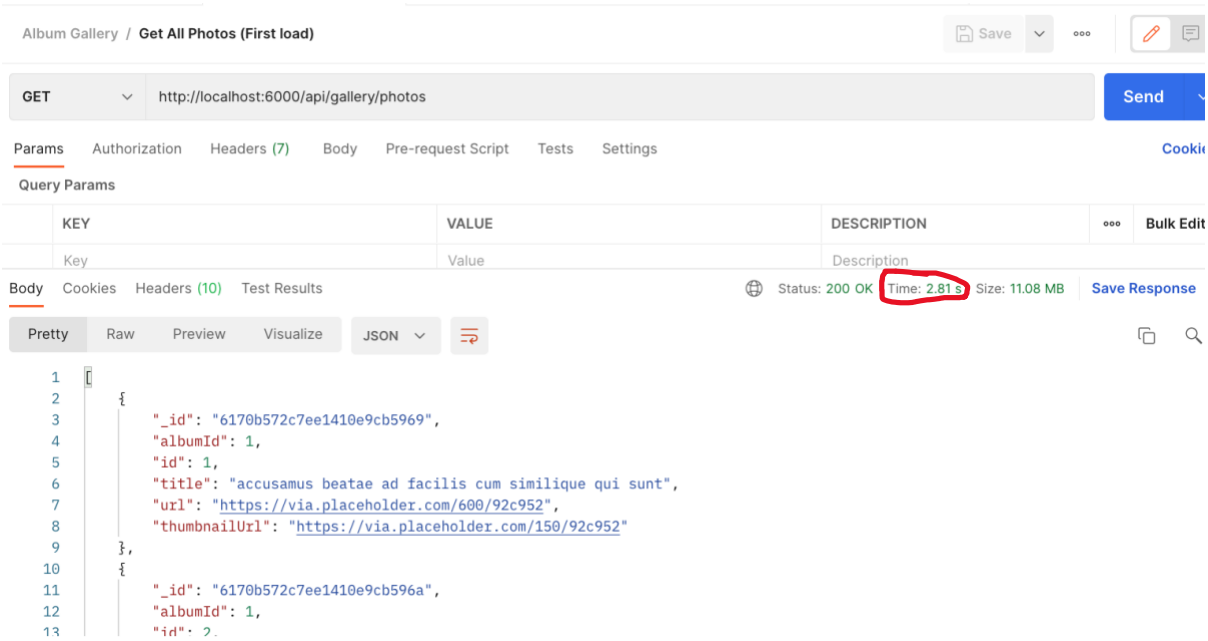
```
DEBUG CONSOLE    PROBLEMS    OUTPUT    TERMINAL

  '4955': new ObjectId("6170baa1103348efded9af77"),
  '4956': new ObjectId("6170baa1103348efded9af78"),
  '4957': new ObjectId("6170baa1103348efded9af79"),
  '4958': new ObjectId("6170baa1103348efded9af7a"),
  '4959': new ObjectId("6170baa1103348efded9af7b"),
  '4960': new ObjectId("6170baa1103348efded9af7c"),
  '4961': new ObjectId("6170baa1103348efded9af7d"),
  '4962': new ObjectId("6170baa1103348efded9af7e"),
  '4963': new ObjectId("6170baa1103348efded9af7f"),
  '4964': new ObjectId("6170baa1103348efded9af80"),
  '4965': new ObjectId("6170baa1103348efded9af81"),
  '4966': new ObjectId("6170baa1103348efded9af82"),
  '4967': new ObjectId("6170baa1103348efded9af83"),
  '4968': new ObjectId("6170baa1103348efded9af84"),
  '4969': new ObjectId("6170baa1103348efded9af85"),
  '4970': new ObjectId("6170baa1103348efded9af86"),
  '4971': new ObjectId("6170baa1103348efded9af87"),
  '4972': new ObjectId("6170baa1103348efded9af88"),
  '4973': new ObjectId("6170baa1103348efded9af89"),
  '4974': new ObjectId("6170baa1103348efded9af8a"),
  '4975': new ObjectId("6170baa1103348efded9af8b"),
  '4976': new ObjectId("6170baa1103348efded9af8c"),
  '4977': new ObjectId("6170baa1103348efded9af8d"),
  '4978': new ObjectId("6170baa1103348efded9af8e"),
  '4979': new ObjectId("6170baa1103348efded9af8f"),
  '4980': new ObjectId("6170baa1103348efded9af90"),
  '4981': new ObjectId("6170baa1103348efded9af91"),
  '4982': new ObjectId("6170baa1103348efded9af92"),
  '4983': new ObjectId("6170baa1103348efded9af93"),
  '4984': new ObjectId("6170baa1103348efded9af94"),
  '4985': new ObjectId("6170baa1103348efded9af95"),
  '4986': new ObjectId("6170baa1103348efded9af96"),
  '4987': new ObjectId("6170baa1103348efded9af97"),
  '4988': new ObjectId("6170baa1103348efded9af98"),
  '4989': new ObjectId("6170baa1103348efded9af99"),
  '4990': new ObjectId("6170baa1103348efded9af9a"),
  '4991': new ObjectId("6170baa1103348efded9af9b"),
  '4992': new ObjectId("6170baa1103348efded9af9c"),
  '4993': new ObjectId("6170baa1103348efded9af9d"),
  '4994': new ObjectId("6170baa1103348efded9af9e"),
  '4995': new ObjectId("6170baa1103348efded9af9f"),
  '4996': new ObjectId("6170baa1103348efded9afa0"),
  '4997': new ObjectId("6170baa1103348efded9afa1"),
  '4998': new ObjectId("6170baa1103348efded9afa2"),
  '4999': new ObjectId("6170baa1103348efded9afa3")
}
}
```

Step 9: Inside the redis-cli, we can perform the following command **keys *** to check whether there are any keys stored inside the redis cache and we should see an empty array result since no data has been saved into the cache:

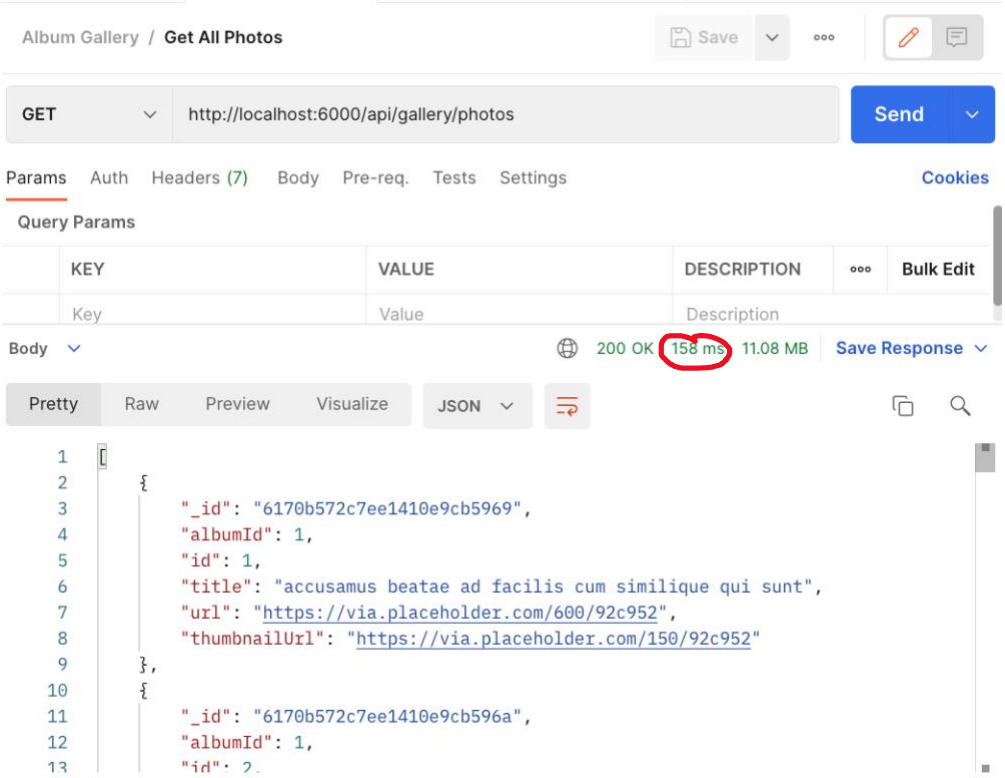
```
DEBUG CONSOLE    PROBLEMS    OUTPUT    TERMINAL

127.0.0.1:6379> keys *
(empty array)
127.0.0.1:6379>
```

Step 9: In Postman software, import the api payload file, AlbumGallery.json.

Step 10: In the Album Gallery collection, run the Get All Photos API call and we can expect the first API call to take a significant longer time since it is the first time the data has been saved into the cache. Below is a screenshot of the time taken for the data when it is saved into the cache:



Step 11: Repeat Step 10 again and observe the time it takes for the subsequent API calls.



You should notice that the time it takes will be way shorter (158 ms) as compared to the first time (2.81 secs).

Step 12: We can execute the **keys *** command in the redis-cli window and you should be able to see the key of getting all photos being saved in the cache.

```
DEBUG CONSOLE   PROBLEMS   OUTPUT   TERMINAL

127.0.0.1:6379> keys *
(empty array)
127.0.0.1:6379> keys *
1) "?allPhotos"
127.0.0.1:6379>
```

Step 13: Now, repeat the steps 10 to 12 for Get single photo API call from the Album Gallery collection.

First run

Album Gallery / **Get single photo**

GET http://localhost:6000/api/gallery/photos/1 **Send**

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

Body 200 OK 99 ms 114.05 KB Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "_id": "6170b572c7ee1410e9cb5969",
        "albumId": 1,
        "id": 1,
        "title": "accusamus beatae ad facilis cum similique qui sunt",
        "url": "https://via.placeholder.com/600/92c952",
        "thumbnailUrl": "https://via.placeholder.com/150/92c952"
    },
    {
        "_id": "6170b572c7ee1410e9cb596a",
        "albumId": 1,
        "id": 2,
```


Subsequent runs (Take shorter time too)

Album Gallery / **Get single photo**

GET http://localhost:6000/api/gallery/photos/1 **Send**

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

Body 200 OK 7 ms 114.05 KB Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "_id": "6170b572c7ee1410e9cb5969",
        "albumId": 1,
        "id": 1,
        "title": "accusamus beatae ad facilis cum similique qui sunt",
        "url": "https://via.placeholder.com/600/92c952",
        "thumbnailUrl": "https://via.placeholder.com/150/92c952"
    },
    {
        "_id": "6170b572c7ee1410e9cb596a",
        "albumId": 1,
        "id": 2,
```

New key added to the redis cache

```
DEBUG CONSOLE    PROBLEMS    OUTPUT    TERMINAL

127.0.0.1:6379> keys *
(empty array)
127.0.0.1:6379> keys *
1) "?allPhotos"
127.0.0.1:6379> keys *
1) "Photos?albumId=1"
2) "?allPhotos"
127.0.0.1:6379>
```

Step 14: When done, we can clear the cache by running the **flushAll** command in the redis-cli console window. You can check with the keys * command and there should be no more keys inside the redis cache.

```
DEBUG CONSOLE    PROBLEMS    OUTPUT    TERMINAL

127.0.0.1:6379> keys *
(empty array)
127.0.0.1:6379> keys *
1) "?allPhotos"
127.0.0.1:6379> keys *
1) "Photos?albumId=1"
2) "?allPhotos"
127.0.0.1:6379> flushAll
OK
127.0.0.1:6379> keys *
(empty array)
127.0.0.1:6379>
```

Step 15: You can shutdown the redis cache by executing **shutdown** command.

```
DEBUG CONSOLE    PROBLEMS    OUTPUT    TERMINAL

127.0.0.1:6379> keys *
(empty array)
127.0.0.1:6379> keys *
1) "?allPhotos"
127.0.0.1:6379> keys *
1) "Photos?albumId=1"
2) "?allPhotos"
127.0.0.1:6379> flushAll
OK
127.0.0.1:6379> keys *
(empty array)
127.0.0.1:6379> shutdown
not connected>
```