**Question-1**

**Step1:** Installed necessary dependency.

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Step2:** Created EMP database and updated in database.js file.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Step3:** Using Thunder client checking the different routes.

Url: <http://localhost:8000/api/employees>

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Checked the same URL on browser and I got the same output with id, name, salary and age

A screenshot of a computer

Description automatically generated

Used below URL and passed the same \_id and I get the same result

* 1. http://localhost:8000/api/employees/618cf962f36b27c5379212b7

A screenshot of a computer

Description automatically generated

**Step4:**

1. **a. How does the Step3:A, B,C work? Explain the workflow, route, and the way the query executed.**

First, I have installed all the dependencies, as the application uses Express for handling HTTP requests, Mongoose for MongoDB interactions, and Body Parser for parsing request bodies. Then

connects to the MongoDB database using the configuration (./config/database.js.), here Employee model is defined using Mongoose, specifying the schema for the "Employee" collection. Now using GET request (/api/employees), the server queries the MongoDB database using Mongoose's Employee.find(). If employees are found, it sends them in JSON format as a response. When a GET request is made to /api/employees/:employee\_id, the server uses Mongoose's Employee.findById to retrieve the details of a specific employee based on the provided ID. When a POST request is made to /api/employees, the server uses Mongoose's Employee.create to add a new employee to the database. After successful creation, it retrieves all employees and sends them back as a response. When a PUT request is made to /api/employees/:employee\_id, the server uses Mongoose's Employee.findByIdAndUpdate to update the details of a specific employee based on the provided ID. It then sends a success message as a response. Now delete endpoint, When a DELETE request is made to /api/employees/:employee\_id, the server uses Mongoose's Employee.remove to delete a specific employee based on the provided ID. It sends a success message as a response. And for the ouput when I use different routes the result will be displayed according to the request.

**b. What is the role of:**

* 1. **module.exports = mongoose.model('Employee', EmpSchema);**

A Mongoose model for the "Employee" collection is exported with this line. It specifies the format of the documents in the collection (e.g., fields like name, income, and age) and defines the schema (EmpSchema) for the collection. In essence, this line builds a model called "Employee" that can be used to communicate with MongoDB's "employees" collection.

* 1. **Employee.findByIdAndUpdate**

Employee To locate an employee by ID and change their details, utilise the Mongoose technique. Three parameters are required: the callback function, the data to be updated, and the employee's ID. As soon as the update is finished, the callback function is called. It is utilised in the app in the code that is given.In response to a PUT request, use the put('/api/employees/:employee\_id',...) route to update an existing employee's data

**c. Using the idea of Step3:C, try to update one of the records in the employee table. Find related route in the code and explain how it works.**

Currently I have 3 records.

A screenshot of a computer

Description automatically generated

Tried to update the james bond record by PUT request and successfully able to that

A screenshot of a computer

Description automatically generated

As you can see the second record is updated.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

**d. Using the idea of Step3:C, try to delete one of the records in the employee table. Find related route in the code and explain how it works.**

By using this url <http://localhost:8000/api/employees/655fb7d710c86d8db15e47aa>

Removing my record.

A screenshot of a computer

Description automatically generated

As you can see here the record is removed from the compass as well.

A screenshot of a computer

Description automatically generated

here the function handles HTTP DELETE requests to the endpoint /api/employees/:employee\_id. It extracts the employee ID from the request parameters.It uses Mongoose's deleteOne method to delete the employee with the specified ID from the MongoDB database.

for Response Handling:

If the employee is successfully deleted (result.deletedCount === 1), it sends a success message.

If no employee is found with the provided ID, it sends a 404-error indicating that the employee was not found.

If an error occurs during the deletion process, it sends a 500 Internal Server Error response.

**Question 2)** You are asked to redesign Question 1 by using the given dataset of Assignment2.

Step 1: Created a cars database that I have used for Assignment-2

A screenshot of a computer

Description automatically generated

Step 2: Redesign the route/code and changed the “model” and routes. That work with Carsales data.

Car.js model

A screen shot of a computer program

Description automatically generated

Created database.js that points to cars database

A screen shot of a computer program

Description automatically generated

* + Show all invoice-info: <http://localhost:6000/api/invoices/>

Worked same way using Find() function as question 1 if it found data then displayed otherwise not found or internal server error.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

* + Show a specific invoice (based on the \_id or invoiceID)

<http://localhost:6000/api/invoices/65636140f0359cebf62c4470>

Finding data based on the particular \_id, used FindbyId() functio.

A computer screen shot of text

Description automatically generated

You can see below it’s only displaying one data that a particular ID data.

A screenshot of a computer

Description automatically generated

* + Insert a new invoice.

<http://localhost:6000/api/invoices>

Inserting data using POST call. I have added below data of my favourite SUV – Urus.

By using simple create() function.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

As you can see below my data is inserted successfully.

A screenshot of a computer

Description automatically generated

* + Delete an existing invoice (based on the \_id or invoiceID)

<http://localhost:6000/api/invoices/65636140f0359cebf62c4470>

removing data by using DELETE and entering specific invoid\_id, here I have used findoneanddelete() function.

A screen shot of a computer program

Description automatically generated

As you can see the selected ID is deleted

A screenshot of a computer program

Description automatically generated

Now using get I’m trying to get the data of this ID but as we have removed data it’s giving me the result as invoice not found.

A screenshot of a computer program

Description automatically generated

* + Update "Manufacturer" & “price\_in\_thousands” of an existing invoice (based on the \_id or invoiceID)

Used <http://localhost:6000/api/invoices/6563672ffec31fa8e6c31b02> with PUT to update the manufacturer and price of the data that I have inserted (Urus) for that I have used findbyIDandUpdate() function.

A screen shot of a computer code

Description automatically generated

Here I have updated manufacturer to Ferrari and Price to 1200000.

A screenshot of a computer

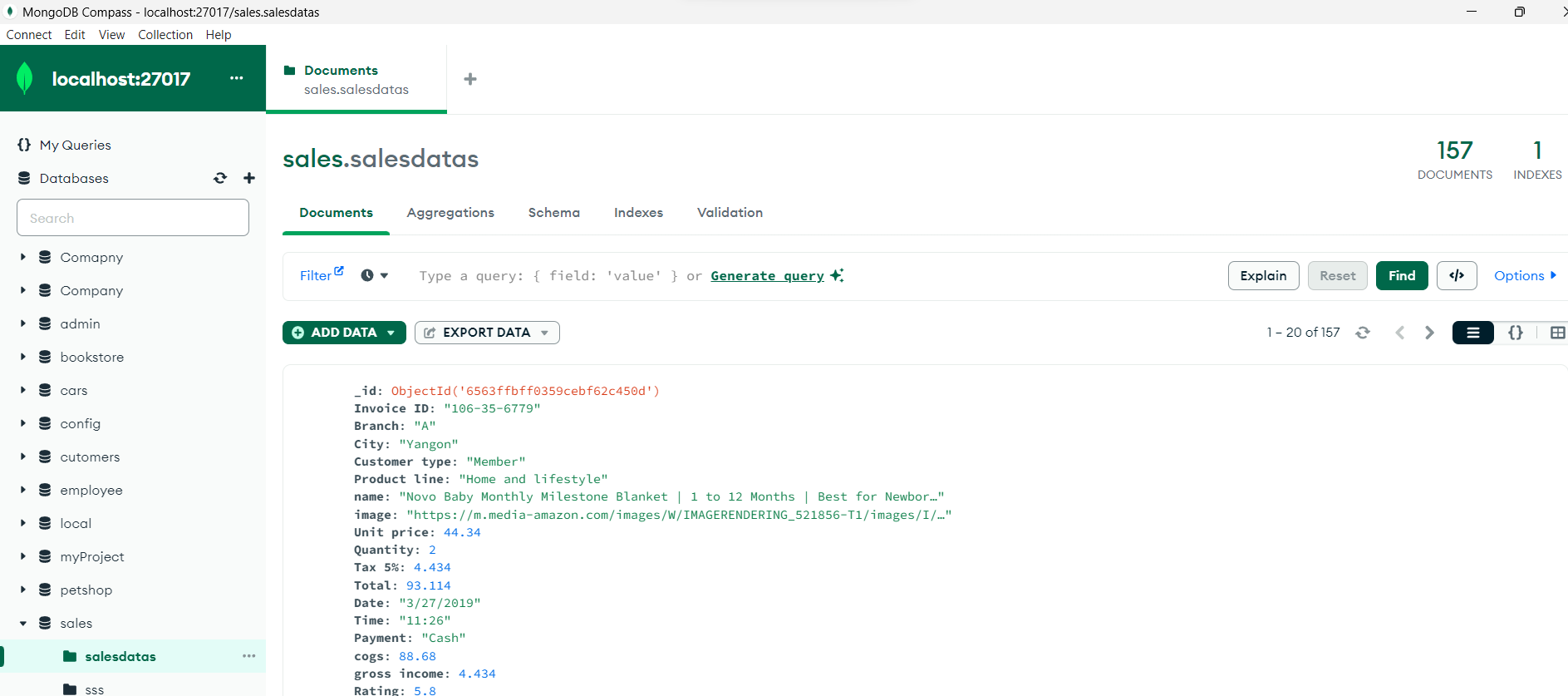
Description automatically generated

Here is the updated result.

A screenshot of a computer program

Description automatically generated

* + Using Handlebar and Form complete the followings.



Showing all information

A screenshot of a computer

Description automatically generated

I have updated hanelbars but still getting errors,

A computer screen shot of a program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

**Question 3)** Open the attached sample JS file. This program use setTimeout() to simulate a running two tasks in asynchronous way. Complete the following steps.

Step 1: and run it using nodemon. Look at the output of the program.

A screenshot of a computer program

Description automatically generated

First connects to the MongoDB database.

It logs '1' to the console.

It logs '2' to the console after successfully connecting to the database.

It logs '3' to the console after accessing the 'customers' collection.

It logs '4' to the console after creating a cursor and limiting the result to 10 documents.

It prints each document to the console using console.log in the loop.

It logs '5' to the console after the loop.

here we have a setTimeout of 5 seconds before calling findAll, there should be a delay before the script starts executing.

Step 2: What if you remove wait from Task1, any error? Explain what you have learned.

A screen shot of a computer program

Description automatically generated

The function await MongoClient.connect(url, { useNewUrlParser: true }) will no longer wait for the connect method to finish before going on to the next lines of code if the await keyword is removed. Rather, it will proceed with running the next few lines as soon as MongoClient.connect is called, which might lead to mistakes.

so, without await.

When MongoClient.connect(url, useNewUrlParser: true) is invoked, the script proceeds on to the following line without waiting for the connection to be formed.

In conclusion, by utilising await, the script makes sure that it waits for the asynchronous connect procedure to finish before starting any database-related operations. If await is not used, there's a chance that attempts to access the database before the connection is available may be made, which could result in problems.

Step3) What if you remove all await/async from the task1 and 2. How do you explain changes in the output compare to Step 1?

A screen shot of a computer program

Description automatically generated

In conclusion, eliminating await and async from this script's asynchronous actions may result in unexpected or incomplete output, a lack of synchronisation, and possible issues brought on by accessing the database before the connection is available. When working with asynchronous actions such as database connections, it is essential to utilise await and async in the right ways to guarantee the proper execution order.

Step4) Bonus: Can you design the given functionality/program using Promise?

A screen shot of a computer program

Description automatically generated

In the above code connectToDatabase provides a promise that, upon connection establishment, resolves with the MongoDB client.

Upon retrieving a promise from the 'customers' collection,' findAll provides a range of documents.

When the MongoDB connection is ended, closeConnection delivers a promise that is resolved.

By chaining these promises using. then and handling errors with .catch, you ensure a more organized and readable flow of asynchronous operations.