Assignment – 4

Ques 1-

At first we are having the 500 error because it is not using asynchronous approach and so the code is returning nothing hence catch is printing. I have used await and async to solve the issue.

A screenshot of a computer screen

Description automatically generated

Adding the data

A screenshot of a computer

Description automatically generated

Data added in the compass

A screenshot of a computer

Description automatically generated

Data returned on the local

A computer screen with a black screen

Description automatically generated

Added another data

A screenshot of a computer screen

Description automatically generated

Searching using the id.

A computer screen with a black screen

Description automatically generated

Updated using the put.

A screenshot of a computer screen

Description automatically generated

Updated on the mongo as well.

A screenshot of a computer

Description automatically generated

Deleting using the id.

A screenshot of a computer screen

Description automatically generated

After deletion in compass

A screenshot of a computer

Description automatically generated

Ques 2-

After updating the first question with assignment 2 json file. We created products and uploaded the data manually in mongo

A screenshot of a computer screen

Description automatically generated

Searching using asin

A computer screen with a black background

Description automatically generated

Searching on thunder client using asin

A screenshot of a computer screen

Description automatically generated

Searching all the products.

A screenshot of a computer screen

Description automatically generated

Searching using asin in thunder client

A screenshot of a computer screen

Description automatically generated

Product deleted

A screenshot of a computer screen

Description automatically generated

Updating the title and price

A screenshot of a computer screen

Description automatically generated

Listing all products using handlebars

A screenshot of a computer

Description automatically generated

A computer screen shot of a computer screen

Description automatically generated

Adding new products

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen with a black background

Description automatically generated

Ques 3-

Step 1-

A screenshot of a computer screen

Description automatically generated

The async function findAll() initiates a connection to MongoDB using MongoClient.connect(), which yields a client object upon successful connection.

Subsequently, it accesses the "mydb" database and the "customers" collection within that database using client.db("mydb") and db.collection('customers').

The statement collection.find({}).limit(10) retrieves all documents from the "customers" collection, restricting the result to a maximum of 10 documents.

The loop cursor.forEach(doc => console.log(doc)) iterates over the documents obtained from the cursor, printing each document to the console.

Finally, the finally block ensures the closure of the MongoDB client connection once the operation concludes.

**Step 2 – If await is removed from task 1**

A screenshot of a computer screen

Description automatically generated

Error Handling in Task1: If the await keyword is omitted, MongoClient.connect() will return a promise immediately. Any errors that occur during the connection process can be caught in the .catch() block. However, since Task1 is not awaited, the subsequent code will continue execution.

Potential Race Conditions: Without await, Task2 (await cursor.forEach(doc => console.log(doc));) may start executing before the connection is fully established and the client is assigned. This situation can create race conditions, leading to potential errors or unexpected behavior when attempting to interact with MongoDB before the connection is fully set up.

Key Learning Point - Importance of Await: The await keyword in async functions plays a critical role in ensuring that promises are resolved before moving to the next line of code. It maintains the flow and order of operations in asynchronous code, promoting code reliability and readability.

Handling Asynchronous Operations: Properly handling promises is essential, particularly in operations like database connections. Using await ensures that subsequent code waits for asynchronous operations to complete before advancing, enhancing code reliability and readability.

Effective Error Handling and Cleanup: Regardless of await usage, implementing robust error handling (e.g., using .catch()) and cleanup logic (e.g., closing connections) is vital to manage potential errors effectively and maintain resource efficiency.

**Step 3-**

A computer screen shot of a program

Description automatically generated

Synchronous Execution: In the absence of async/await, the code operates synchronously. This entails that each line executes sequentially, without pausing for asynchronous operations (e.g., database connections or queries) to finish.

Lack of Promise Waiting: Task1 (MongoClient.connect()) utilizes a catch block to manage connection errors, but it doesn't wait for the connection to complete before proceeding. Similarly, Task2 (cursor.forEach()) starts iterating immediately without ensuring that the cursor is fully populated.

Potential Promise Handling Issues: Without await, promises returned by asynchronous operations aren't explicitly handled. This may result in unhandled promise rejections or race conditions, especially with time-consuming asynchronous tasks.

Output Variations: Output may differ from scenarios using async/await. Without waiting for asynchronous tasks to finish, the code might log incomplete or empty results, especially in Task2 (cursor.forEach()), where iteration may begin before the cursor is fully loaded.

Challenges in Error Handling: Dealing with errors in asynchronous operations becomes more complex without async/await. While Task1's catch block can capture connection errors, managing errors in Task2 and ensuring proper cleanup (e.g., closing the client connection) becomes more intricate.

**Step 4-**

**Bonus question- Done using promise**

A computer screen with many colorful text

Description automatically generated

In this code:

The function findAll() is structured to return a promise that resolves once the iteration process is finished successfully or rejects if any errors occur. It handles the MongoDB connection and iteration using promises (via then and catch), replacing the usage of async/await. The setTimeout function is employed to delay the execution of findAll() by 5 seconds. After findAll() finishes its task, it logs 'iter' to the console. The code also features proper error handling for scenarios such as MongoDB connection errors, iteration errors, or issues during the closing process, ensuring that errors are appropriately logged and managed.