Rajeshwari Reddy Gummadi

U2276440

Reflection of Lab Exercise week 7

**Web and Mobile Application Development**

This week we learnt how to demonstrates CRUD (Create, Read, Update, Delete) operations in MongoDB using Node.js and Mongoose. The objective is to integrate MongoDB commands into a Node.js application, enabling dynamic database interactions.

What we learnt was to:

* Connect to a MongoDB database using Mongoose.
* Create a new database.
* Create a collection in the database.
* Insert one or multiple documents into a collection programmatically.

Project Setup

* Created a directory named WEEK8.
* Installed MongoDB and Mongoose using the terminal commands:
* npm install mongodb –save
* npm install mongoose –save
* mongodb node.js driver installed
* write npm init in terminal window and enter press.
* Press Enter to accept the default values
* This created a package.json file in your project folder.

**Task 1: connect with mongoDB and add a single document**

Open visual studio code open folder name week8 and create a new file index.js.

* Import mongoose in index.js file
* Connect mongoose with mongodb url
* Put the code and save
* Write node index.js in terminall to see the output.

A black screen with colorful text

Description automatically generated with medium confidence

* Database called week8 is created in mongodb

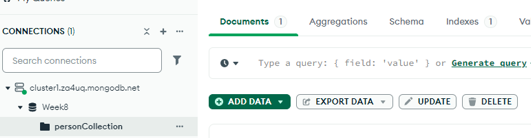
A screenshot of a computer

Description automatically generated

* Person collection (model name) is created

A black background with white text

Description automatically generated

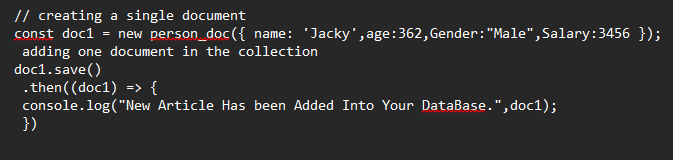


* Schema is created called as PersonSchema where you have name, age, gender, salary.

A screenshot of a computer program

Description automatically generated

* Single document is created in the person collection



A computer screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* This code will catch the error and print it to the console, making it easier to understand what went wrong. This is a common way to handle errors in JavaScript, especially when working with promises.

A black rectangle with white text

Description automatically generated

* **.catch((err) => { ... })**: This is a method that catches any errors that occur in the promise chain. If the promise is rejected, the function inside .catch will be executed.
* **(err)**: This is the parameter that represents the error object. When an error occurs, it gets passed to this function.
* **console.error (err);**: This line logs the error to the console. console.error is used to print error messages, which helps in debugging.

**Task 2: Adding multiple documents**

* Multiple document is created

A computer screen shot of a black screen

Description automatically generated

A screen shot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

**Task 3 fetching the data use of find:**

* This is the code to find the name, age and salary from the collection.

A screenshot of a computer program

Description automatically generated

**A screen shot of a computer code

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Task 4 find command with filtering criteria. Run the find command s for which gender = Female and age is greater than some given number.**

* Code to find all female who’s age is greater than or equal to 20

A computer screen with white text

Description automatically generated

**A screen shot of a computer program

Description automatically generated**

**Tasks 5 : write a query to return the total number of documents in the collection.**

* With this code we can count total number of documents

A screen shot of a computer

Description automatically generated

**A screen shot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Task 6 Delete the documents for a given criteria**

* If we want to delete any records use this code.

A screenshot of a computer screen

Description automatically generated

There were total 11 documents. So, there are two documents after filtering.

**A screen shot of a computer program

Description automatically generated**



A screenshot of a computer

Description automatically generated

**Task 7 update all document of which the gender is female and set their salary to 5555 Add following code in index.js save and run it**

* To update any records to set salary 5555 for Female this will be the code.

A screen shot of a computer

Description automatically generated

**A screen shot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**This is the full code index.js**

A screen shot of a computer program

Description automatically generated A screen shot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated