**CN5006 Lab Session Week 8 Portfolio**

**Prepared by:Radhe Shyam Yadav  
Date: *19-oct-2024***

**Portfolio for MongoDB CRUD Operations in Node.js Application**

Hello, I’m Radhe Shyam , and this portfolio showcases my project based on the MongoDB CRUD operations tutorial provided by Dr. Nadeem Qazi. The tutorial guided me through performing CRUD operations (Create, Read, Update, Delete) in MongoDB within a Node.js application. The goal of this tutorial was to help me integrate MongoDB commands into a Node.js application for seamless dynamic interactions with the database.

Technology stack

Here are some stack that I used to complete this week 8 lab tutorial:

**Visual studio code:** Code editor used for writing and editing the application**.**

**Node.js**: JavaScript runtime environment used for building server-side applications.

**MongoDB**: NoSQL database used for storing and managing application data.

**Mongoose:** MongoDB object modelling tool for Node.js, used to define schemas and interact with MongoDB**.**

By the end of this tutorial, here are some objective that I achieved:

-Connect to MongoDB,

-Create a new database and collection programmatically

-Insert documents

Query the database

-Update and delete documents

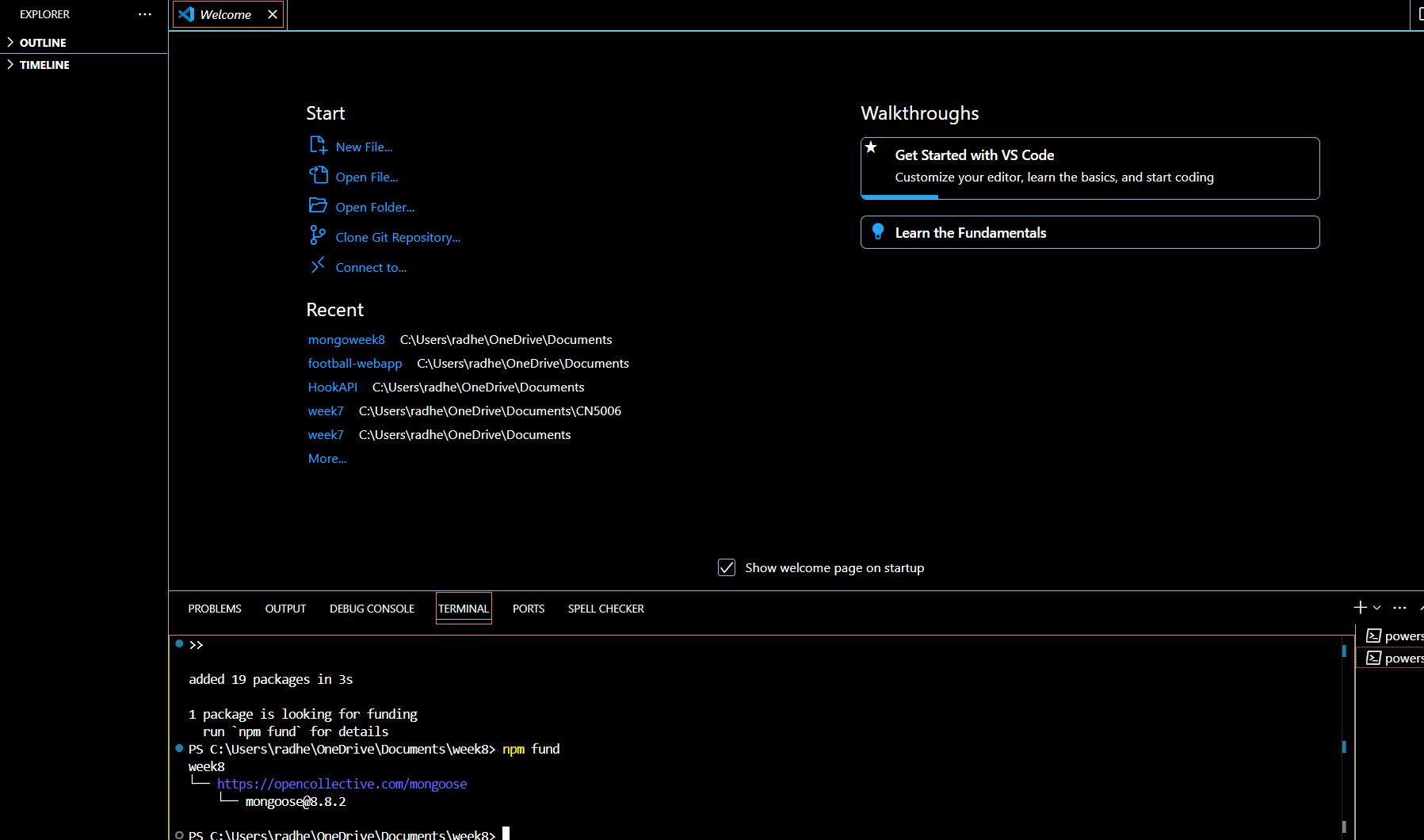
**Step 1: project setup**

To get started, I created a new directory for my project and initialized a Node.js project using the npm init command. Then I installed the necessary dependencies like Mongoose to interact with the MongoDB database.

npm init



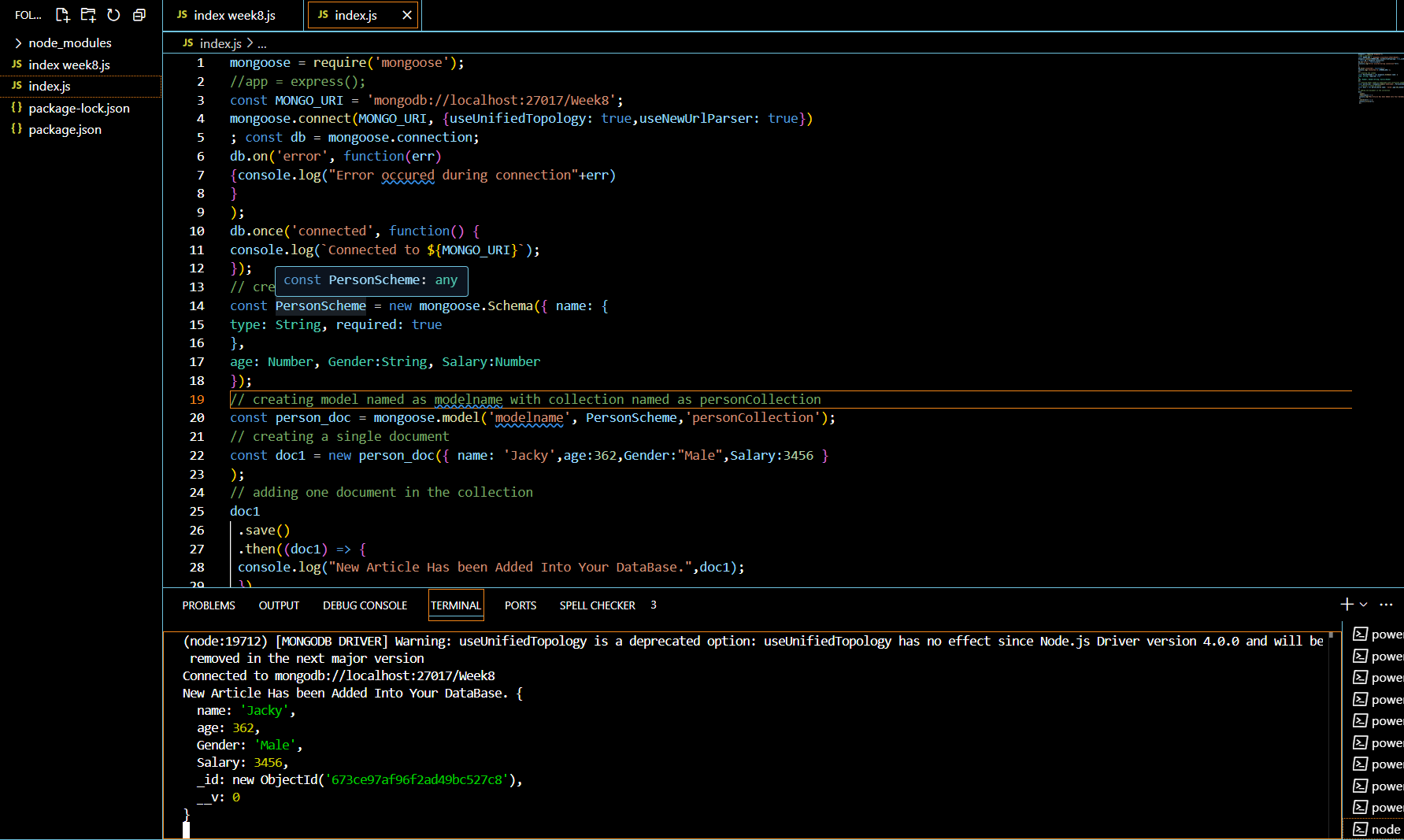
Npm install mangoose -save



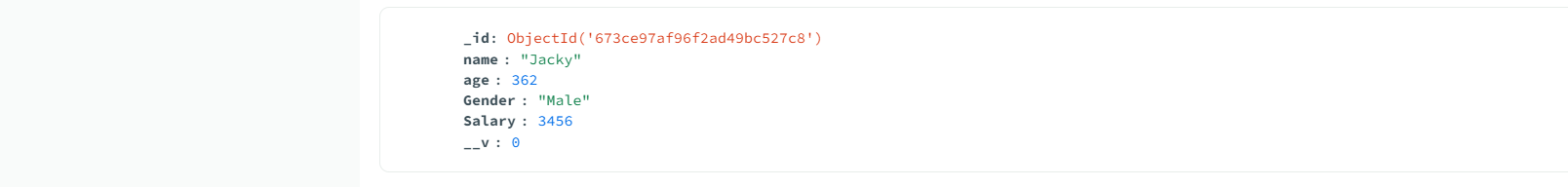
We connect with mongoDB and add a single document.

For this we create a new file in the correct directory location mongoweek8 called ‘index.js’. and we add add this code .

Here is the screenshot of the single document



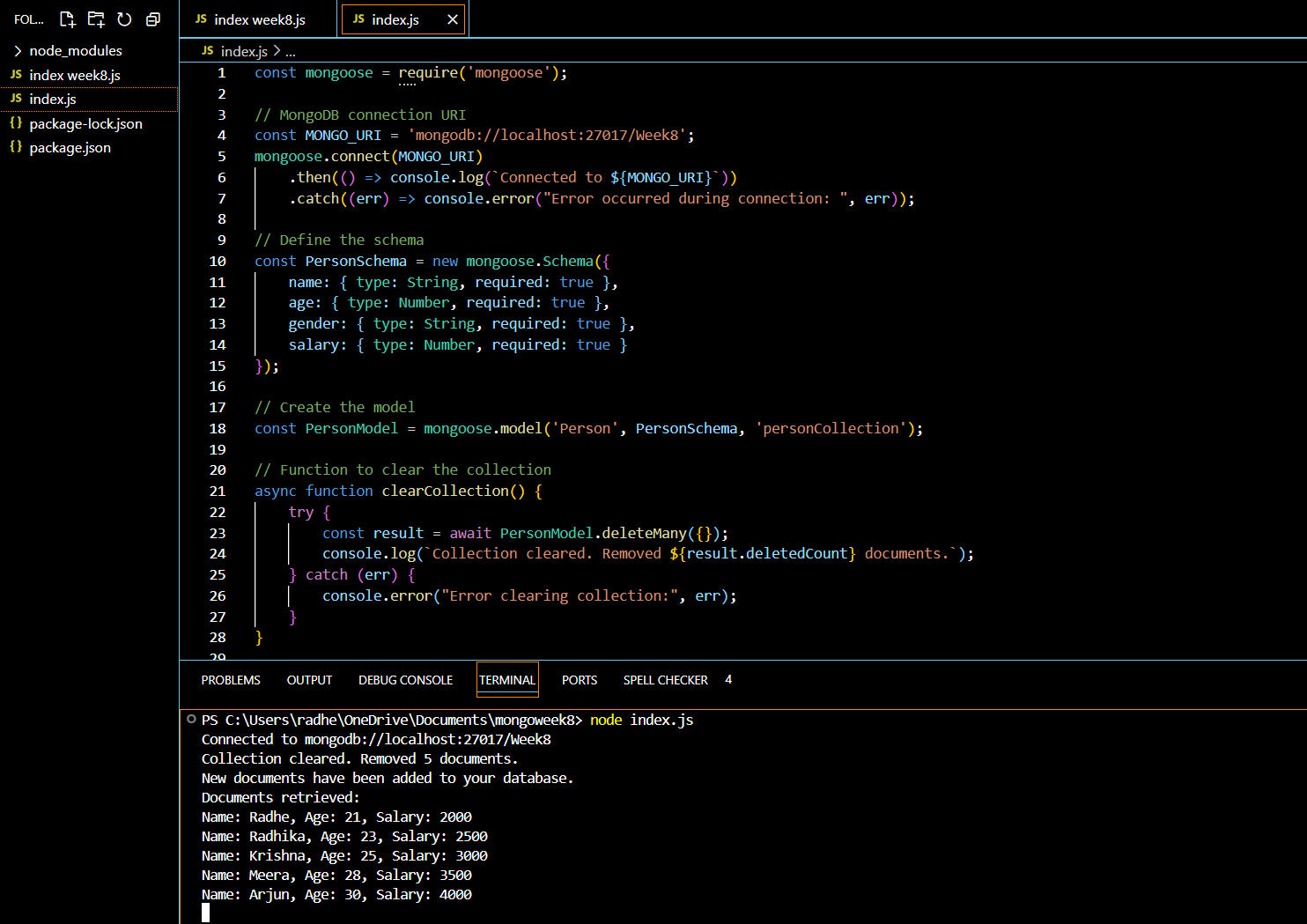
MongoDB connected



For adding multiple document we add this code

manypersons=[{ name: 'Simon',age:42,Gender:"Male",Salary:3456 } ,{ name: 'Neesha',age:23,Gender:"Female",Salary:1000 } ,{ name: 'Mary',age:27,Gender:"Female",Salary:5402 }, { name: 'Mike',age:40,Gender:"Male",Salary:4519 } ] person\_doc.insertMany(manypersons).then(function(){ console.log("Data inserted") // Success }).catch(function(error){ console.log(error) // Failure });

So final code with multiple document is mention bellow with picture,



**Code Example for Fetching All Records:**

Person.find().limit(5).then((docs) => {

console.log("Retrieved documents:", docs);

}).catch((err) => {

console.error("Error fetching documents:", err);

});

**Challenges & Solutions**

* **Handling Asynchronous Operations**: One of the challenges I faced was understanding how to handle asynchronous operations such as saving and querying documents. Using **Promises** (.then() and .catch()) helped manage the flow of these operations and handle errors effectively.
* **Connecting to MongoDB**: Initially, I faced issues with connecting to MongoDB locally. After ensuring that MongoDB was running on the correct port and properly configured in the URI, the connection was successful.

My technical abilities to integrate databases into programming programs were improved by the MongoDB CRUD Operations lesson, which was a worthwhile educational experience. I gained knowledge on how to use Mongoose to streamline database processes, such as building models and schemas and carrying out CRUD tasks. My comprehension of fundamental ideas like data modelling, asynchronous programming, and error handling has improved as a result of this practical approach.   
  
Even though I encountered difficulties with initial setup and asynchronous activities, resolving them helped me become more adept at addressing problems. I've gained useful knowledge from this lesson that I can use to create web apps in the real world, providing a strong basis for next development initiatives.

Reflection:

My technical abilities to integrate databases into programming programs were improved by the MongoDB CRUD Operations lesson, which was a worthwhile educational experience. I gained knowledge on how to use Mongoose to streamline database processes, such as building models and schemas and carrying out CRUD tasks. My comprehension of fundamental ideas like data modelling, asynchronous programming, and error handling has improved as a result of this practical approach.   
  
Even though I encountered difficulties with initial setup and asynchronous activities, resolving them helped me become more adept at addressing problems. I've gained useful knowledge from this lesson that I can use to create web apps in the real world, providing a strong basis for next development initiatives.

**Conclusion**

This tutorial provided valuable hands-on experience with MongoDB CRUD operations in a Node.js application. I learned how to:

* Integrate **Mongoose** into a Node.js project.
* Perform **Create**, **Read**, **Update**, and **Delete** operations on a MongoDB database.
* Use **Promises** to handle asynchronous operations and ensure smooth database interactions.

**Reference:**

* Qazi, N. (2024) Using MongoDB CRUD Operations in a Programming Application. CN5006 Web and Mobile Application Development. University of east London.
* Node.js (n.d.) Node.js Official Documentation. Available at: <https://nodejs.org> (Accessed: 25 November 2024).
* MongoDB (n.d.) MongoDB Manual. Available at: <https://www.mongodb.com/docs> (Accessed: 25 November 2024).
* Mongoose (n.d.) Mongoose Documentation. Available at: https://mongoosejs.com/docs (Accessed: 25 November 2024).
* Visual Studio Code (n.d.) Visual Studio Code Documentation. Available at: <https://code.visualstudio.com/docs> (Accessed: 25 November 2024).