**Roll No :- 2105048**
**Name:- Manav Jain**
**Date:-03/10/2023**

# Assignment No-11

## Aim:-

Create a web application that performs CRUD operations (database connectivity).

## LO Mapped:- LO1,LO2,LO3,LO4,LO5,LO6

## Theory: (overview of project) :-

The Student Database Management App is a robust system designed to facilitate the four fundamental operations in database management: Create, Read, Update, and Delete (CRUD). The system comprises three main components, each serving a specific role in this process: the frontend developed using React and Bootstrap, the backend powered by Express, and the database managed through MongoDB. This architecture combines the strengths of different technologies to create a versatile and efficient student information management system.

**Frontend (React and Bootstrap):** React is a popular JavaScript library for building user interfaces. It allows developers to create dynamic and responsive web applications by breaking down the UI into reusable components. In this context, React is used to design the user interface for the Student Database Management App. Bootstrap, a widely used CSS framework, complements React by providing a plethora of pre-designed UI elements, ensuring a clean and user-friendly design. The combination of React and Bootstrap enables the development of an interactive and visually appealing frontend, making it easier for users to interact with the database.

**Backend (Express):** The backend, powered by Express, serves as the intermediary layer between the frontend and the database. Express is a minimal and flexible Node.js web application framework, designed to simplify the creation of robust and scalable APIs. In this app, Express handles HTTP requests from the frontend and communicates with the database to execute CRUD operations. It plays a crucial role in managing the logic that governs how data is accessed, modified, and deleted, ensuring the security and reliability of the application.

**Database (MongoDB):** MongoDB, a NoSQL database, is used as the backend data store for this application. MongoDB's document-oriented, schema-less structure is well-suited for storing student data, as it allows for easy scalability and adaptability to changing data requirements.

Students' information can be stored as JSON-like documents, making it simple to add or remove fields as needed. MongoDB also supports the indexing of data for efficient retrieval and querying, enhancing the app's ability to deliver student information quickly and accurately.

In this system, when a new student needs to be registered, the frontend provides an intuitive interface for inputting the student's details. Once submitted, the Express backend processes the request, validates the data, and then inserts it into the MongoDB database, creating a new student entry.

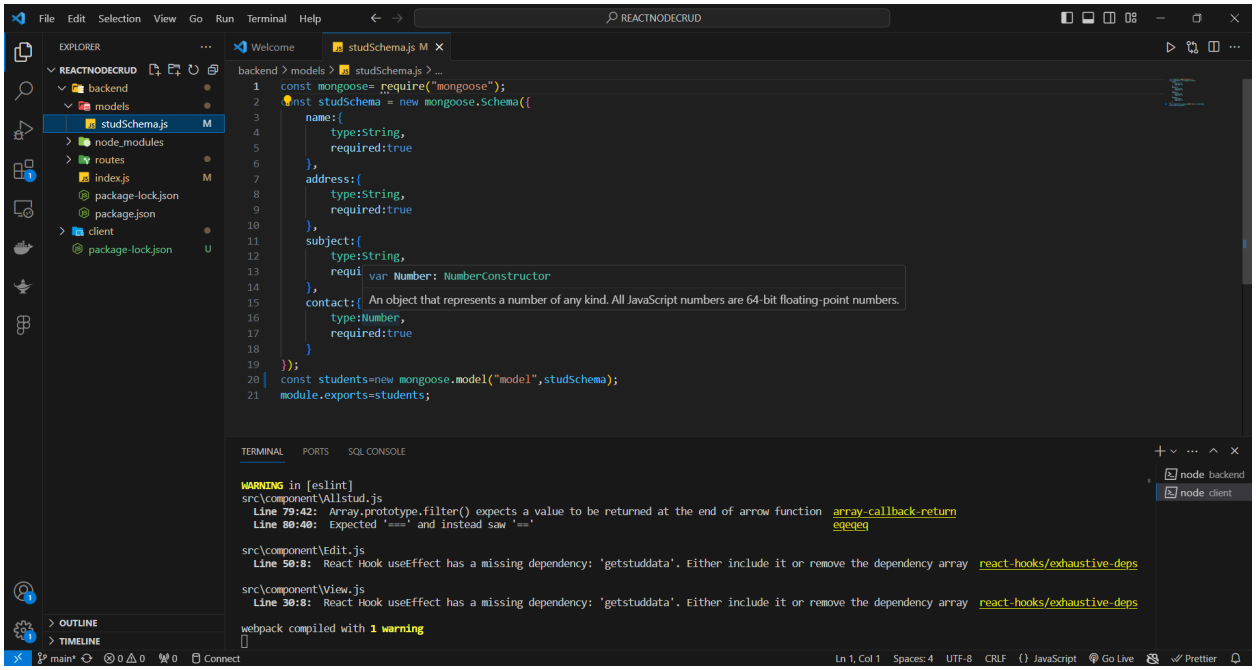
For updating information, the frontend offers a user-friendly form to modify the student's data. This change is then sent to the Express backend, which updates the corresponding record in the MongoDB database.

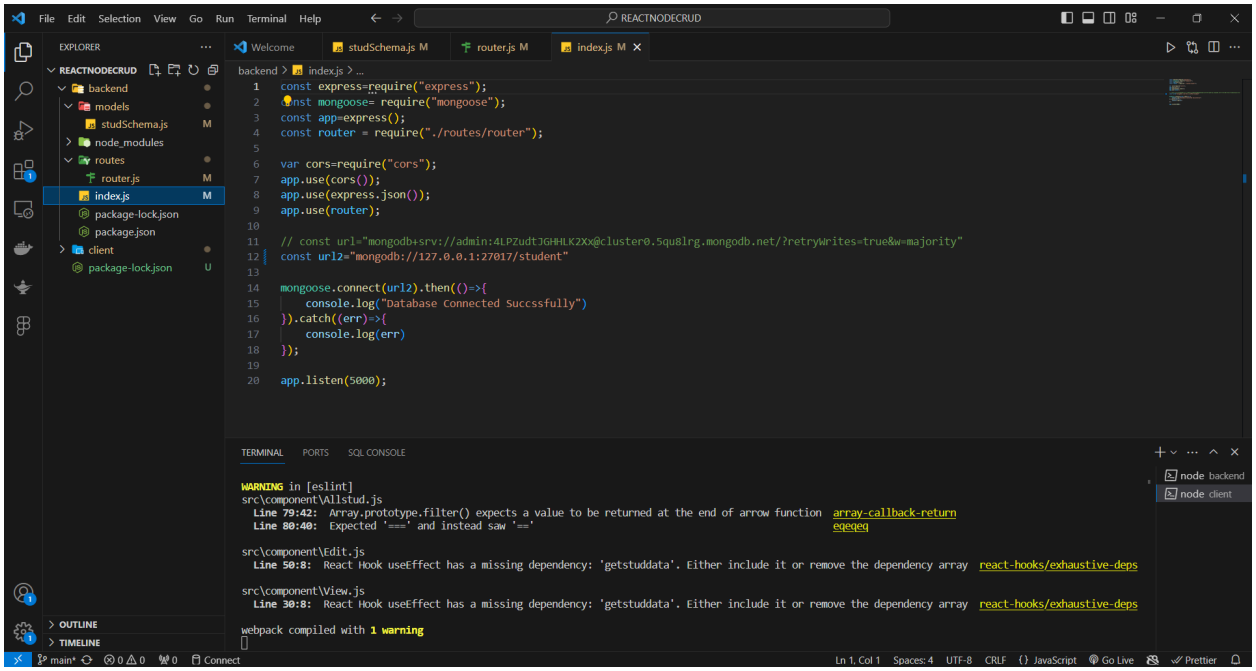
To view student information, the frontend communicates with the Express backend, which queries the MongoDB database, retrieves the requested data, and sends it back to the frontend for display.

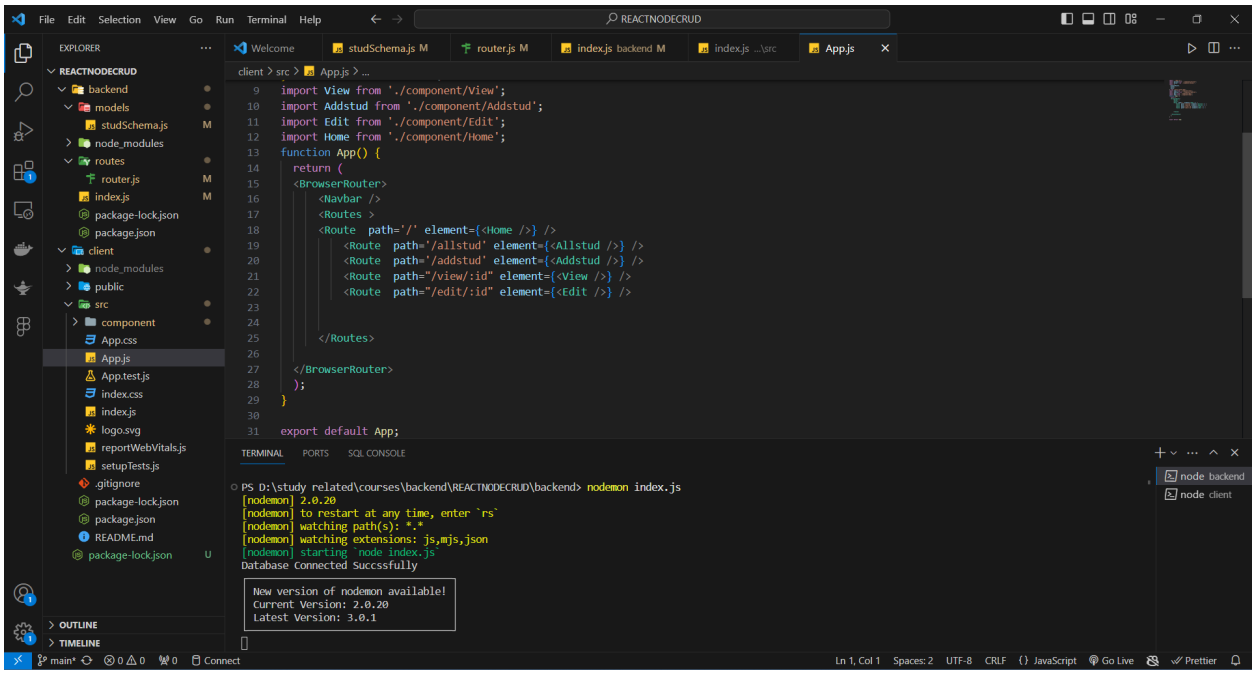
When a student's record needs to be removed, the frontend triggers a request to the backend, which locates and deletes the specified entry from the MongoDB database.

This architecture enables the app to handle the complete lifecycle of student data effectively. The combination of **React, Bootstrap, Express, and MongoDB** ensures a smooth and efficient experience for users, making student data management straightforward and intuitive.

Code :-

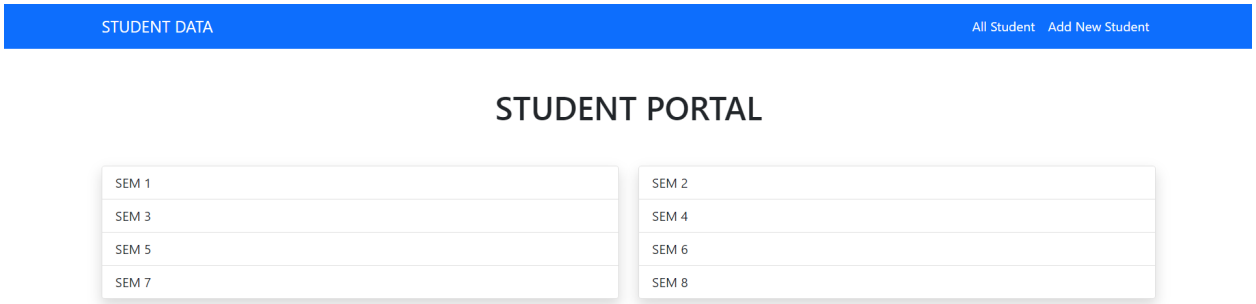

File Edit Selection View Go Run Terminal Help
REACTNODECRUD
EXPLORER
REACTNODECRUD
backend
models
studSchema.js M
node_modules
routes
index.js M
package-lock.json
package.json
client
package-lock.json U
Welcome
studSchema.js M
backend > models > studSchema.js > ...
const mongoose= require("mongoose");
const studSchema = new mongoose.Schema({
name:{
type:String,
required:true
},
address:{
type:String,
required:true
},
subject:{
type:String,
requi
},
contact:{
type:Number,
required:true
}
});
const students=new mongoose.model("model",studSchema);
module.exports=students;
var Number: NumberConstructor
An object that represents a number of any kind. All JavaScript numbers are 64-bit floating-point numbers.
TERMINAL PORTS SQL CONSOLE
node backend
node client
WARNING in [eslint]
src\component\Allstud.js
Line 79:42: Array.prototype.filter() expects a value to be returned at the end of arrow function array-callback-return
Line 80:40: Expected '===' and instead saw '==' eqeqeq
src\component\Edit.js
Line 50:8: React Hook useEffect has a missing dependency: 'getstuddata'. Either include it or remove the dependency array react-hooks/exhaustive-deps
src\component\View.js
Line 30:8: React Hook useEffect has a missing dependency: 'getstuddata'. Either include it or remove the dependency array react-hooks/exhaustive-deps
webpack compiled with 1 warning
OUTLINE
TIMELINE
main* 0 0 0 Connect
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} JavaScript Go Live Prettier

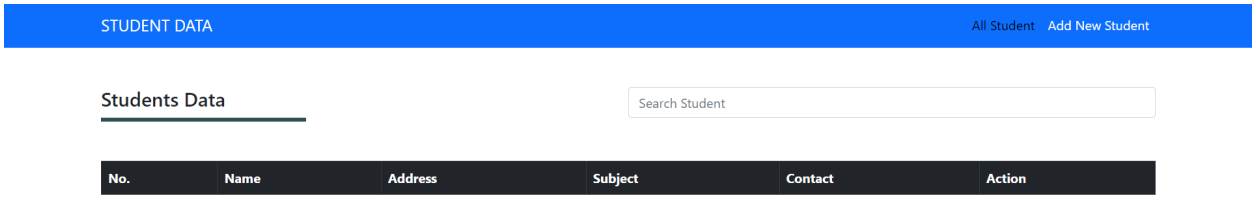

File Edit Selection View Go Run Terminal Help
REACTNODECRUD
EXPLORER
REACTNODECRUD
backend
models
studSchema.js M
node_modules
routes
router.js M
index.js M
package-lock.json
package.json
client
package-lock.json U
Welcome
studSchema.js M
router.js M
index.js M
backend > index.js > ...
const express=require("express");
const mongoose= require("mongoose");
const app=express();
const router = require("./routes/router");
var cors=require("cors");
app.use(cors());
app.use(express.json());
app.use(router);
// const url="mongodb+srv://admin:4LPZudtJGHHLK2Xx@cluster0.5qu8lrg.mongodb.net/?retryWrites=true&w=majority"
const url2="mongodb://127.0.0.1:27017/student"
mongoose.connect(url2).then(()=>{
console.log("Database Connected Succssfully")
}).catch((err)=>{
console.log(err)
});
app.listen(5000);
TERMINAL PORTS SQL CONSOLE
node backend
node client
WARNING in [eslint]
src\component\Allstud.js
Line 79:42: Array.prototype.filter() expects a value to be returned at the end of arrow function array-callback-return
Line 80:40: Expected '===' and instead saw '==' eqeqeq
src\component\Edit.js
Line 50:8: React Hook useEffect has a missing dependency: 'getstuddata'. Either include it or remove the dependency array react-hooks/exhaustive-deps
src\component\View.js
Line 30:8: React Hook useEffect has a missing dependency: 'getstuddata'. Either include it or remove the dependency array react-hooks/exhaustive-deps
webpack compiled with 1 warning
OUTLINE
TIMELINE
main* 0 0 0 Connect
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} JavaScript Go Live Prettier



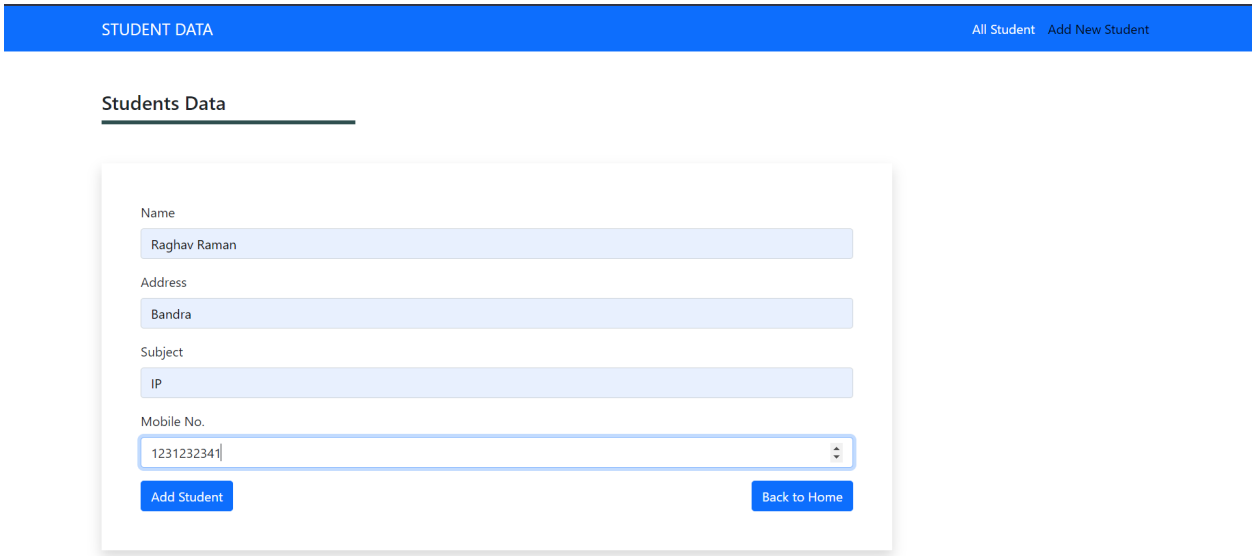

# Output :-

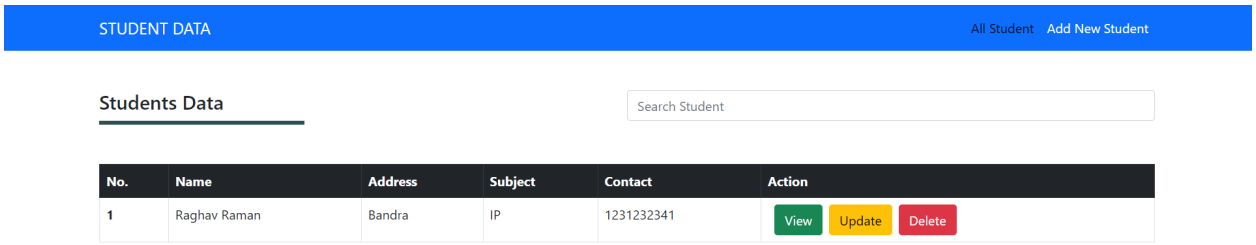
# 1)Frontend :-

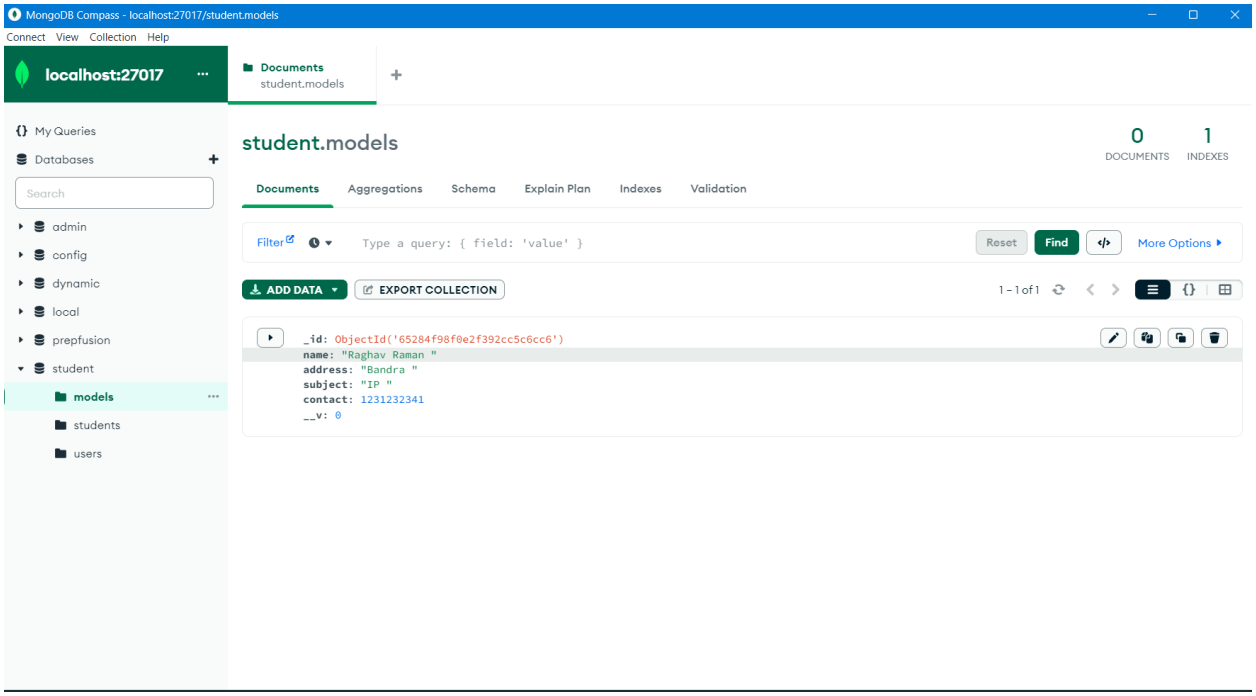



## 2)Create and Read operation :-


STUDENT DATA
All Student
Add New Student
Students Data
Name
Raghav Raman
Address
Bandra
Subject
IP
Mobile No.
1231232341
Add Student
Back to Home


STUDENT DATA
All Student
Add New Student
Students Data
Search Student
No.
Name
Address
Subject
Contact
Action
1
Raghav Raman
Bandra
IP
1231232341
View
Update
Delete




# 3)Update operation :-

STUDENT DATA

All Student Add New Student

## Edit Student Information

Student Name

Raghav Raman

Student Address

Mumbai

Student Subject

CNS

Student Mobile

1231232341

Update Data Back to Home

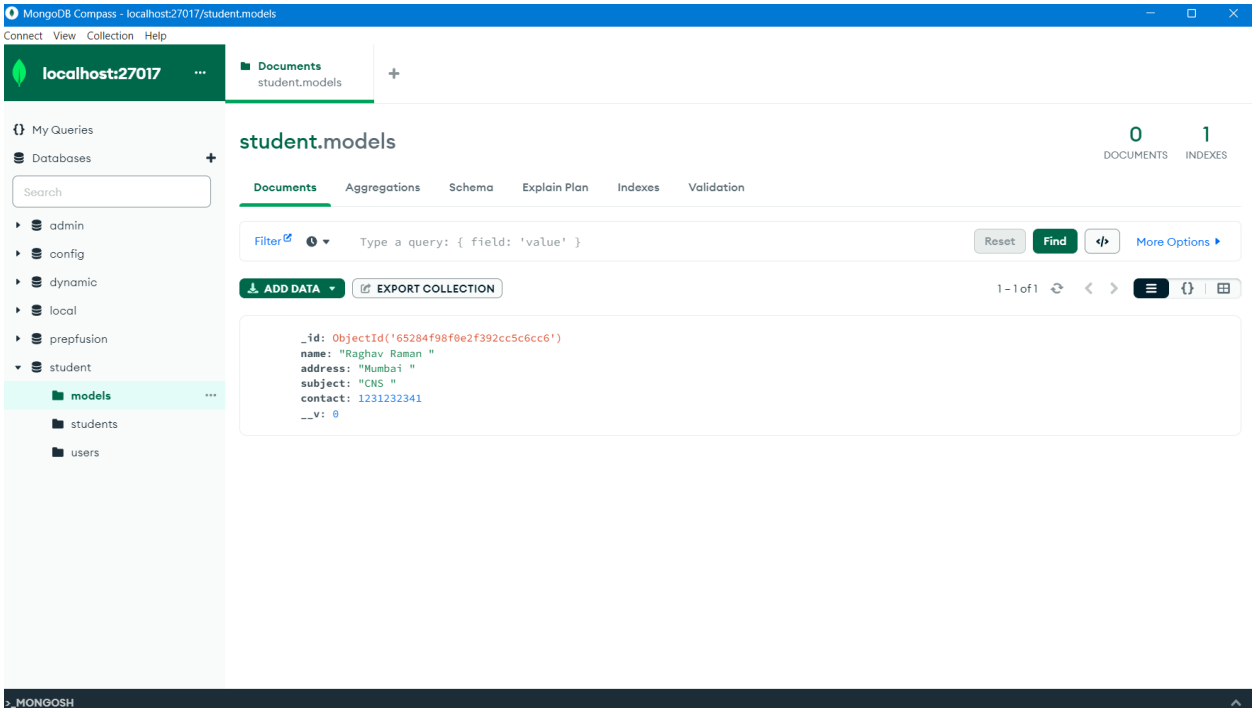
STUDENT DATA

All Student Add New Student

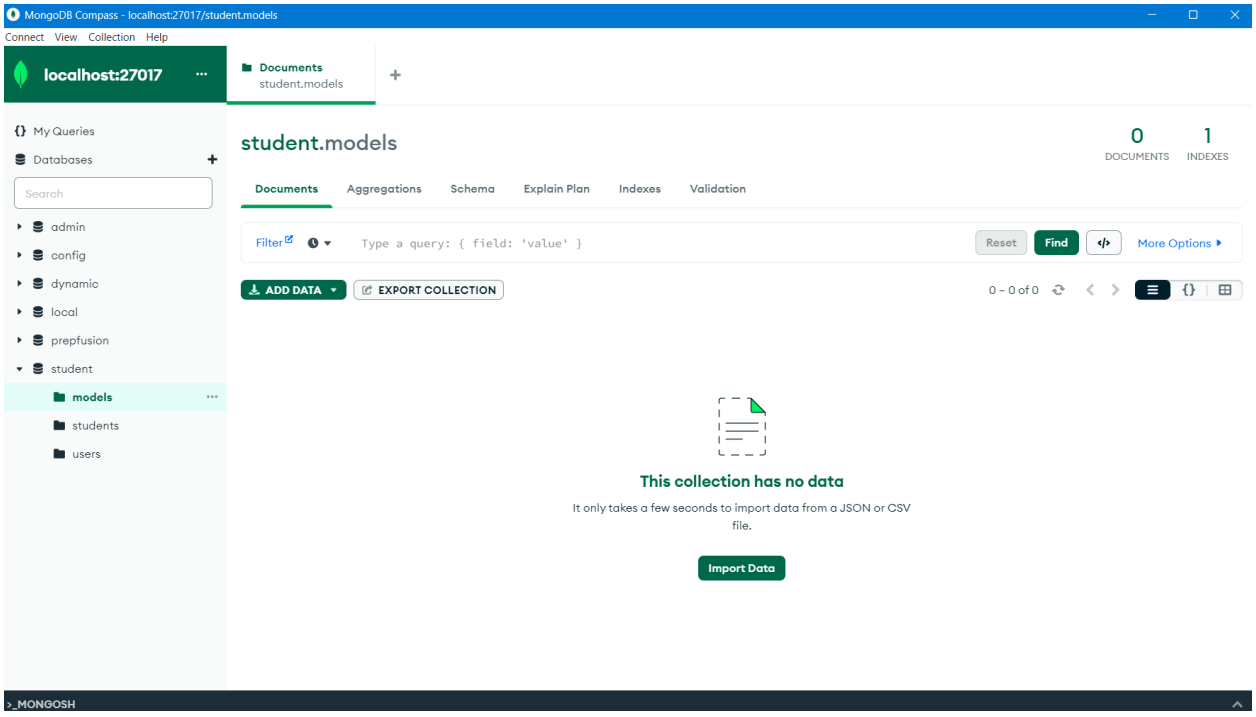
## Students Data

Search Student

| No. | Name | Address | Subject | Contact | Action |
|---|---|---|---|---|---|
| 1 | Raghav Raman | Mumbai | CNS | 1231232341 | View Update Delete |




# 4)Delete operation :-



**References :-**

1https://www.mongodb.com/docs/manual/crud/
2https://www.geeksforgeeks.org/mongodb-crud-operations/

**Conclusion :-**

Learnt to make a fullstack application using react in the frontend, backend with express.js and node.js and database with mongodb also learned to integrate the frontend with backend and connect to database.