24CS2502-Mern Full Stack Development

Exercise 3

Integrate MongoDB as the backend database for storing data

Aim:

To integrate MongoDB as the backend database in a MERN (MongoDB, Express.js, React, Node.js) stack application to efficiently store, manage, and retrieve data. By utilizing MongoDB, a NoSQL database, this project seeks to enhance scalability, performance, and flexibility in handling unstructured or semi-structured data, enabling smooth interaction between the front-end and back-end components of the application. The integration will ensure data persistence, provide efficient querying capabilities, and streamline data management processes in the MERN ecosystem.

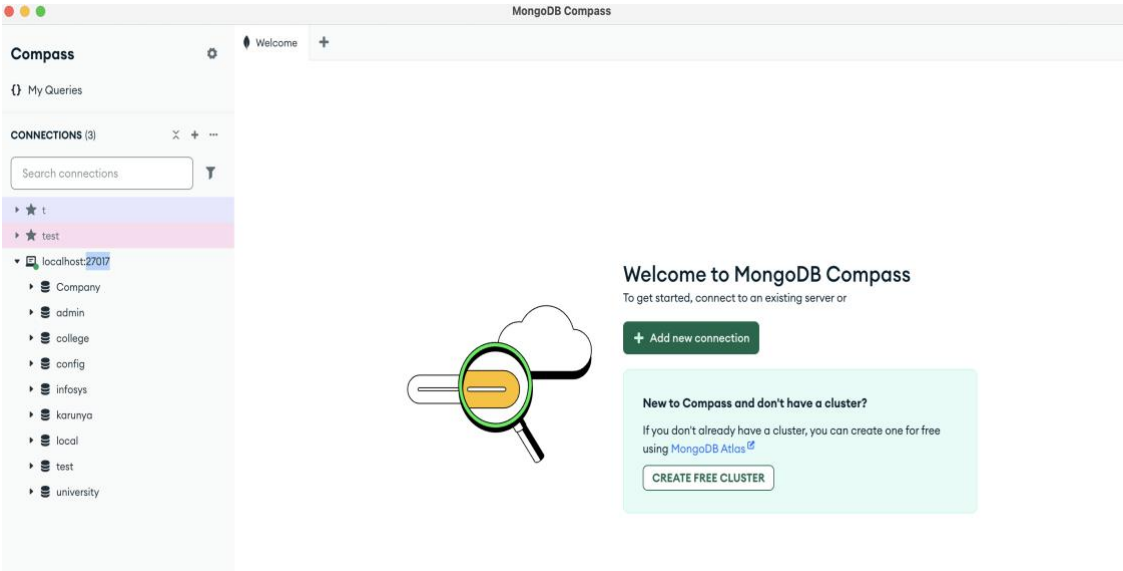
Question

Design a User Interface Form for the allotted application themes[Refer GCR] using Tailwind CSS utility classes. Integrate MongoDB database and implement the following steps to store the user form data collected via React front end in a MongoDB database backend.

Steps to Integrate MongoDB in a MERN Stack Project

1. Download and Install MongoDB Community Server and MongoDB Compass:
    - Go to the MongoDB download page and select the appropriate version for your operating system.
    - Also, install GUI tool called MongoDB Compass from MongoDB Compass

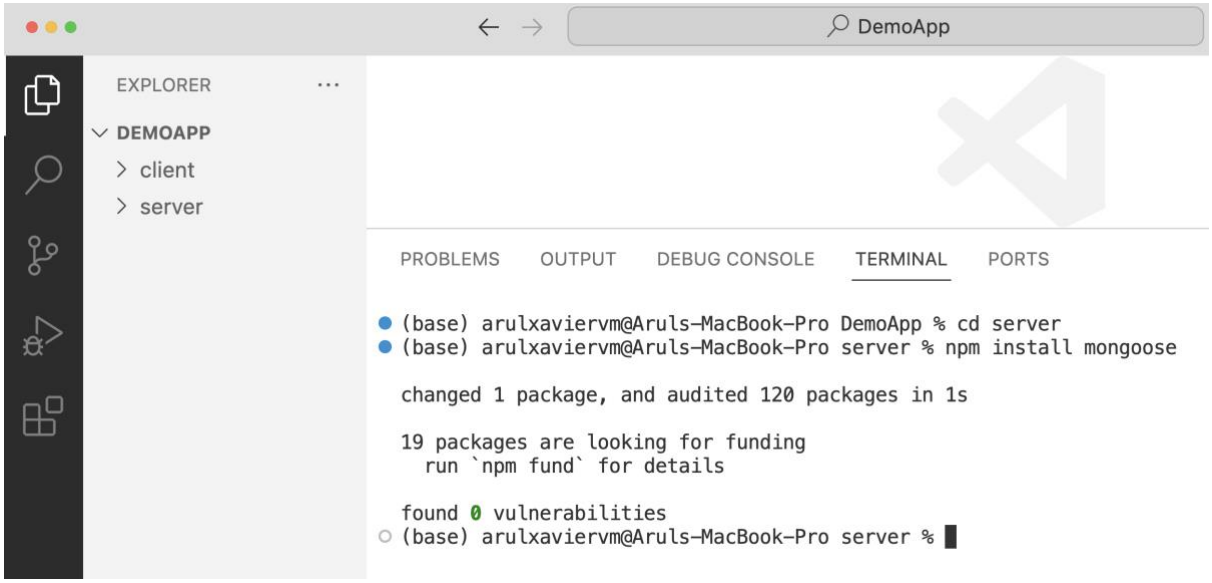
2. After installation, start the MongoDB Compass and connect localhost:27017



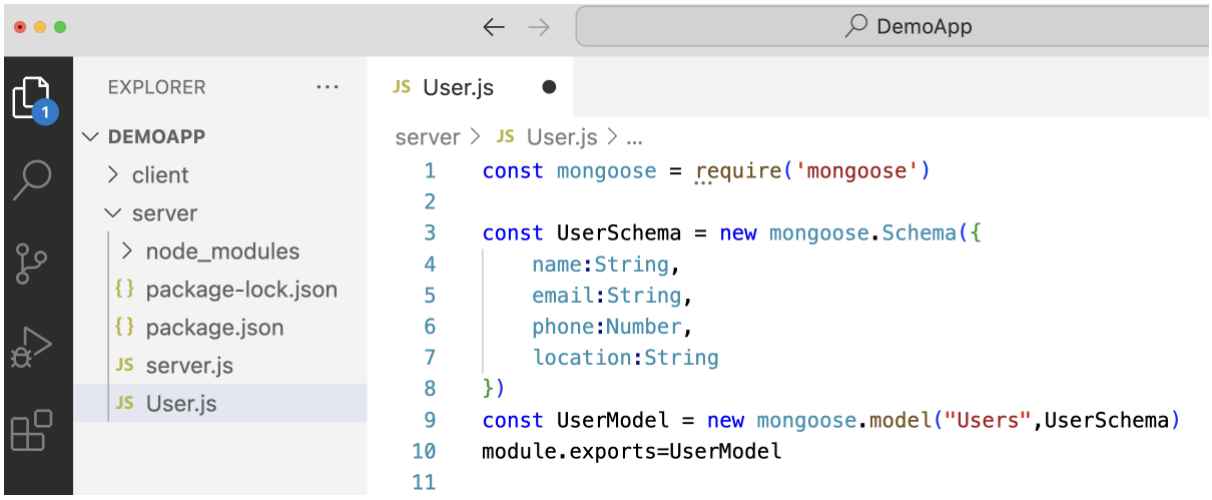
3. Open MERN project created earlier

4. Navigate to server directory and install “mongoose” module for database operations.

>> npm install mongoose



5. Define a simple schema and model for storing User data in a separate file User.js under the server directory



<u>User.js</u>

```
const mongoose = require('mongoose')
//define schema
const UserSchema = new mongoose.Schema({
    name:String,
    email:String,
    phone:Number,
    location:String
})
//create model object
const UserModel = new mongoose.model("Users",UserSchema)
module.exports=UserModel
```

6. Include the programming code to **connect mongodb database** and **create RESTFul API route for storing user data** inside Server.js as per the following
   - Include mongoose and UserModel

```
const UserModel = require('./User')
const mongoose = require('mongoose')
```

   - Connect to mongodb database

```
mongoose.connect('mongodb://127.0.0.1:27017/Company')
.then(() => console.log('DB connected'))
.catch(err => console.log(err))
```

   - Create server API Route to store user data

```
//Register API Route
app.post('/register',(req,res)=>{
    UserModel.create(req.body)
    .then(res.json('Data Saved Successfully'))
    .catch(err=>res.json(err))
})
```

7. Test the Server API Route using Postman

   Run the server program

   server > npm run dev

   > server@1.0.0 dev
   > nodemon server.js

   [nodemon] 3.1.7
   [nodemon] to restart at any time, enter `rs`
   [nodemon] watching path(s): *.*
   [nodemon] watching extensions: js,mjs,cjs,json
   [nodemon] starting `node server.js`
   Server running on port 9000
   DB connected

   Postman is a popular tool used primarily for **Testing Server API Routes**. It provides a user-friendly interface to send HTTP requests to web servers and analyze the responses, making it easy to test and interact with APIs (Application Programming Interfaces).

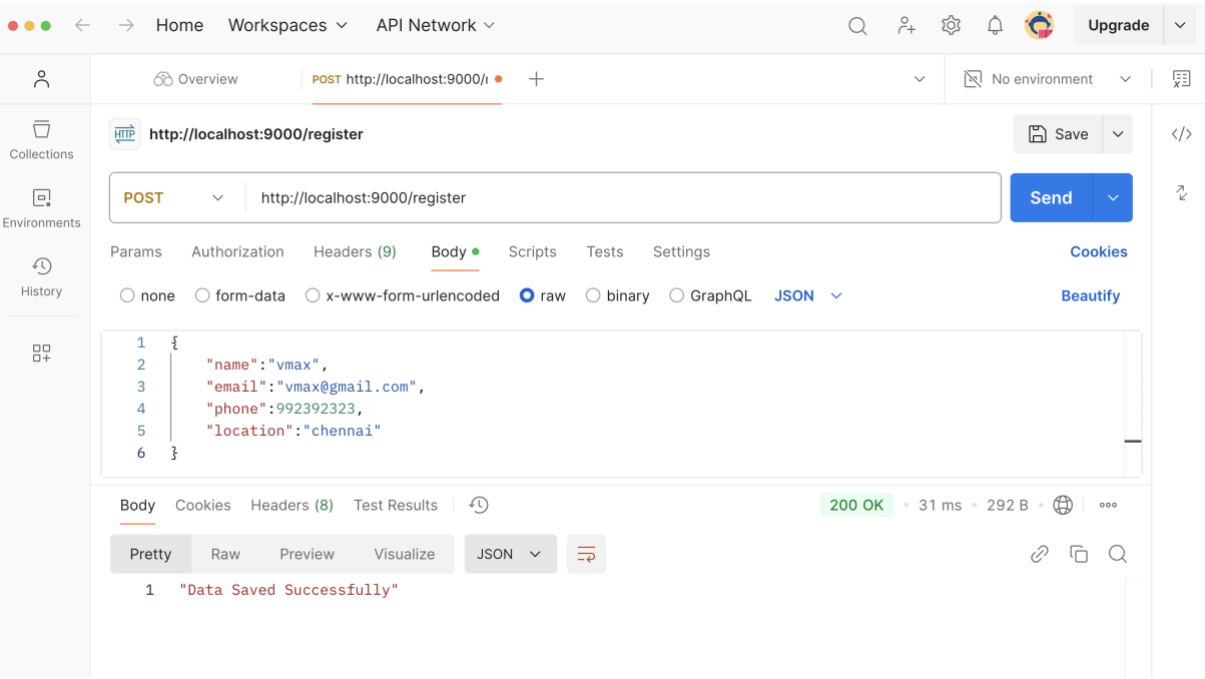
   https://www.postman.com/downloads/ **[Install from here]**

Test the Server API Route:

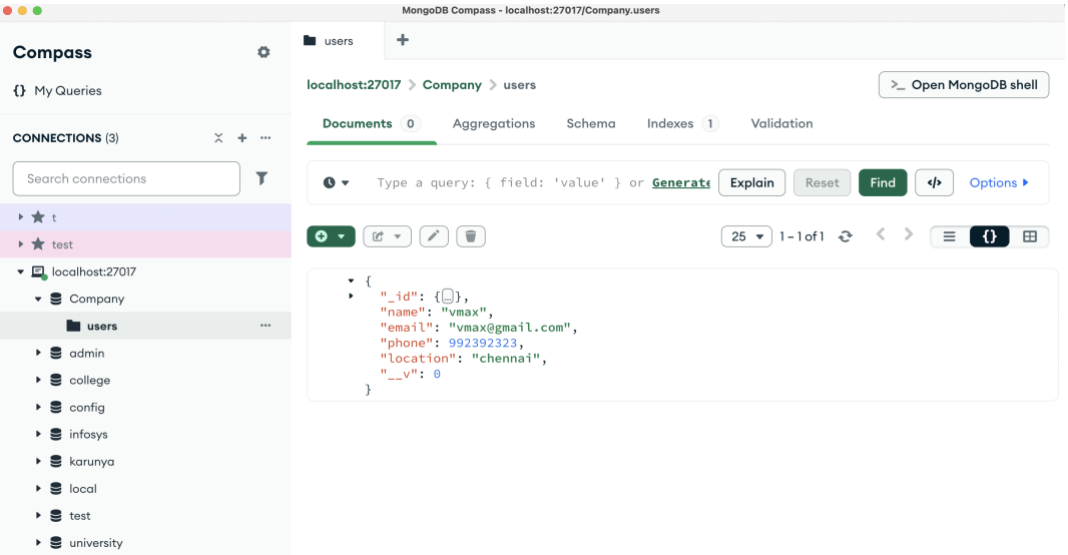
- **Open Postman** and click on the `New` button or just enter the API URL in the request bar.
- Select `POST` as the HTTP method.
- Enter the URL for the API endpoint (e.g., `http://localhost:9000/register`).
- In the `Body` tab, you can enter the data for the new user in `JSON` format:

```
{
  "name":"vmax",
  "email":"vmax@gmail.com",
  "phone":992392323,
  "location":"chennai"
}
```

- Click the `Send` button.
- Postman will display the response status code and the data returned by the server



Data Saved Successfully...



8. Customize the React client app [RegisterForm.js] to collect User form data and Store in a mongodb server via Server API Route[already tested via postman] using axios module functionalities.

RegisterForm.js

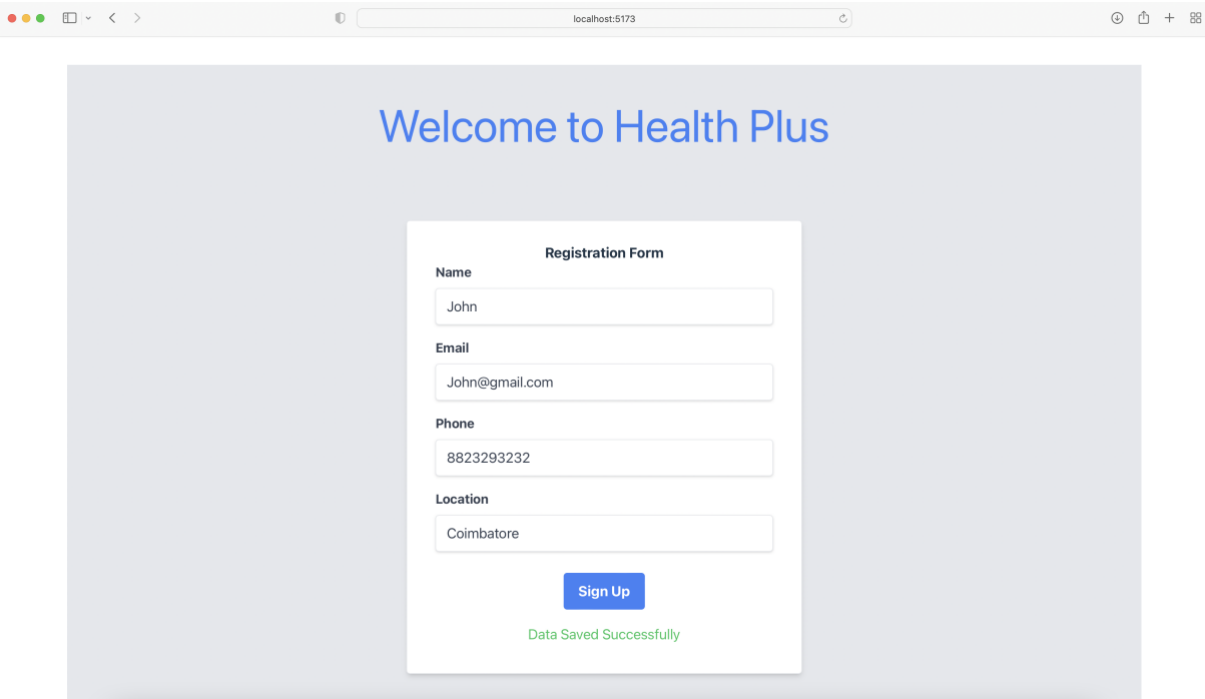
```
import axios from "axios";
import React, { useState } from "react";


function RegisterForm() {
  const [name,setName] = useState()
  const [email,setEmail] = useState()
  const [phone, setPhone] = useState()
  const [location,setLocation] = useState()
  const [status,setStatus] = useState()

  const handleSubmit = (e) => {
    e.preventDefault()
axios.post('http://localhost:9000/register',
         {name,email,phone,location})
    .then(result=>setStatus(result.data))
    .catch(err=>setStatus(err))
  }

  return (
    <div>
    <form onSubmit={handleSubmit}>
      <h3>Registration Form</h3>
      <div>
        <label> Name </label>
```

```
        <input onChange={(e)=>setName(e.target.value)}></input>
      </div>
      <div>
        <label> Email </label>
        <input onChange={(e)=>setEmail(e.target.value)}></input>
      </div>
      <div>
        <label> Phone </label>
        <input onChange={(e)=>setPhone(e.target.value)}></input>
      </div>
      <div>
        <label> Location </label>
        <input onChange={(e)=>setLocation(e.target.value)}></input>
      </div>
      <div>
        <button type="submit"> Sign Up </button>
      </div>
      <div> {status} </div>
    </form>
  </div>
  );
}
export default RegisterForm
```

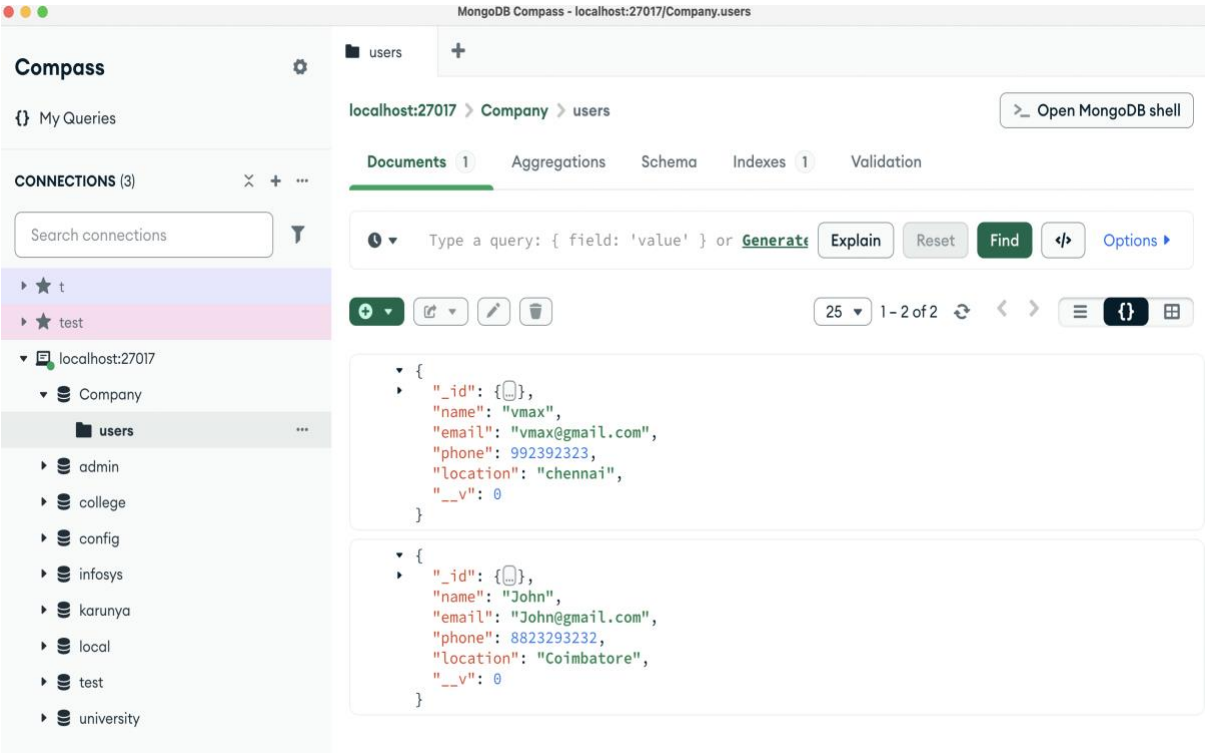
9. Run the React client App and Test the form submission

```
client> npm run dev

> client@0.0.0 dev
> vite


  VITE v6.0.3  ready in 281 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

10. Open client URL in Browser



11. Verify the MongoDB Database



Setting up Git and pushing a new project to GitHub:

Step 1: Install Git Software:

- **Download Git** from the official Git website: https://git-scm.com/downloads
- **Verify Installation:** After installing Git, you can verify that Git is installed correctly by opening a **terminal (command prompt or Git Bash)** and typing:

  git –version

  If Git is installed correctly, it will display the version number.

Step 2: Set Up Git Global Username:

- Run the following command in your terminal (replace "Your Name" with your desired username of Github account):

  git config --global user.name "Your Name"

Step 3: Set Up Git Global Email:

- **Set your Git email** that will be associated with your Git commits. This email will appear in your commit logs.
- Run the following command in your terminal (replace "your.email@example.com" with your actual email address):

  git config --global user.email "your.email@example.com"

Step 4: Initialize Git in Your Project Directory:

- You can do this by navigating to your MERN project folder in the terminal

  cd project_folder_name

  Example:

  cd DemoApp
- **Create a** README.md **file** in the root of your project directory:

  echo "# MERN-APP1" >> README.md
- **Initialize Git** in your project directory:

  git init

Step 5: Add Files to Git:

- **Stage all files** in your project directory (this includes the README.md and any other files you may have created):

  git add .

  This command stages all the files in the project directory for the next commit.

Step 6: Commit the Changes:

- **Commit the staged files** with a descriptive message:

  git commit -m "first commit"

  This creates the first commit with the message "first commit".

Step 7: Create a New GitHub Repository:

- **Go to GitHub** in your browser and log into your account.
- **Create a new repository**:
  - Click on the "+" icon in the top-right corner and select **"New repository"**.
  - Enter the repository name (e.g., MERN-APP1).
  - Optionally, add a description and choose visibility (public or private).
  - Click **Create repository**.

Step 8: Add GitHub Repository as Remote

- **Copy the URL** of the GitHub repository you just created (it will look like https://github.com/your-username/MERN-APP1.git).

- **Add the GitHub repository as a remote** to your local Git repository:
  git remote add origin https://github.com/arulxavier857/MERN-APP1.git

Step 9: Push Your Code to GitHub:

git push -u origin main

- The -u flag sets the upstream reference, meaning that you can just run git push in the future without specifying the remote and branch.
- The origin is the name of the remote (GitHub repository).
- main is the branch you're pushing to.

If this is your first push, Git may ask for your **GitHub username** and **personal access token** (instead of your password).

Verify your Github Account: