Full Stack Application:

**Frontend**

Index.js:

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

const root = ReactDOM.**createRoot**(document.**getElementById**('root'));

root.**render**(

  <**React.StrictMode**>

    <**App** />

  </**React.StrictMode**>

);

**reportWebVitals**();

App.js:

import React from "react";

import CreateUser from "./components/CreateUser";

import RetrieveCourses from "./components/RetrieveCourses";

const **App** = () => {

  return (

    <div>

      <h1>CRUD Application</h1>

      <**CreateUser** />

      <**RetrieveCourses** />

    </div>

  );

};

export default **App**;

CreateUser.js:

import React, { useState } from "react";

import axios from "axios";

const **CreateUser** = () => {

  const [user, **setUser**] = **useState**({

    name: "",

    email: "",

    address: "",

  });

  const **handleChange** = (e) => {

**setUser**({ ...user, [e.target.name]: e.target.value });

  };

  const **handleSubmit** = async (e) => {

    e.**preventDefault**();

    try {

      const response = await axios.**post**("http://localhost:8600/api/user/create", user);

      console.**log**(response);

**alert**("User created successfully!");

**setUser**({ name: "", email: "", address: "" });

    } catch (error) {

**alert**(error.response?.data?.message || "Error creating user");

    }

  };

  return (

    <div>

      <h2>Create User</h2>

      <form *onSubmit*={**handleSubmit**}>

        <input *type*="text" *name*="name" *placeholder*="Name" *value*={user.name} *onChange*={**handleChange**} *required* />

        <input *type*="email" *name*="email" *placeholder*="Email" *value*={user.email} *onChange*={**handleChange**} *required* />

        <input *type*="text" *name*="address" *placeholder*="Address" *value*={user.address} *onChange*={**handleChange**} *required* />

        <button *type*="submit">Create User</button>

      </form>

    </div>

  );

};

export default **CreateUser**;

RetrieveCourses.js:

import React, { useEffect, useState } from "react";

import axios from "axios";

const **RetrieveCourses** = () => {

  const [courses, **setCourses**] = **useState**([]);

**useEffect**(() => {

    const **fetchCourses** = async () => {

      try {

        const response = await axios.**get**("http://localhost:8600/api/user/retrieve");

        console.**log**(response.data);

**setCourses**(response.data);

      } catch (error) {

        console.**error**("Error fetching courses", error);

      }

    };

**fetchCourses**();

  }, []);

  return (

    <div>

      <h2>Courses</h2>

      <ul>

        {courses.**map**((course, index) => (

          <li *key*={index}>{JSON.**stringify**(course)}</li>

        ))}

      </ul>

    </div>

  );

};

export default **RetrieveCourses**;

**Backend:**

Index.js:

import express from "express";

import mongoose from "mongoose";

import dotenv from "dotenv";

import bodyParser from "body-parser";

import cors from "cors"; *// Import CORS*

import route from "./routes/userRoutes.js";

dotenv.**config**();

const app = **express**();

*// Enable CORS*

app.**use**(**cors**({ origin: "http://localhost:3001" })); *// Allow requests from React frontend*

app.**use**(bodyParser.**json**());

app.**use**("/api/user", route);

const PORT = process.env.PORT || 5000;

const MONGOURL = process.env.MONGO\_URL;

**mongoose**

  .**connect**(MONGOURL)

  .**then**(() => {

    console.**log**("Database Connection Successful");

    app.**listen**(PORT, () => {

      console.**log**("Server is listening on PORT " + PORT);

    });

  })

  .**catch**((error) => {

    console.**log**(error);

  });

userRoutes.js:

import express from "express"

import {fetch, create, retrieve} from"../controller/userController.js"

const route=express.**Router**();

route.**get**("/fetch",**fetch**);

route.**post**("/create",**create**);

route.**get**("/retrieve",**retrieve**);

export default route;

userModel.js:

import mongoose from "mongoose";

*// Define the User Schema*

const userSchema = new **mongoose**.**Schema**({

    name: {

**type**: **String**,

        required: true

    },

    email: {

**type**: **String**,

        required: true

    },

    address: {

**type**: **String**,

        required: true

    }

});

*// Export the User model*

export const **User** = **mongoose**.**model**("users", userSchema);

*// Create a model for "courses" collection without specifying a schema*

export const **Course** = **mongoose**.**model**("courses", new **mongoose**.**Schema**({}, { collection: "courses", strict: false }));

userController.js:

import {User, Course} from "../model/userModel.js"

export const **create** = async(req,res)=>{

    try{

        const userData=new **User**(req.body);

        const {name}=userData;

        const userExist=await **User**.**findOne**({name});

        if(userExist){

            return res.**status**(400).**json**({message:"User already Exists."});

        }

        const saveUser=await userData.**save**();

        res.**status**(200).**json**({saveUser});

    }

    catch(error){

        res.**status**(500).**json**({error:"Iam From Create Controller"});

    }

}

export const **retrieve** = async (req, res) => {

    try {

      const courses = await **Course**.**find**();

      if (courses.length === 0) {

        return res.**status**(200).**json**({ message: "No courses found", data: [] });

      }

      res.**status**(200).**json**(courses);

    } catch (error) {

      res.**status**(500).**json**({ error: "Internal Server Error" });

    }

  };

export const **fetch** = async(req,res)=>{

    try{   return res.**json**("Hello Praba");    }

    catch(error){

        res.**status**(500).**json**({error:"Iam From Fetch Controller"});

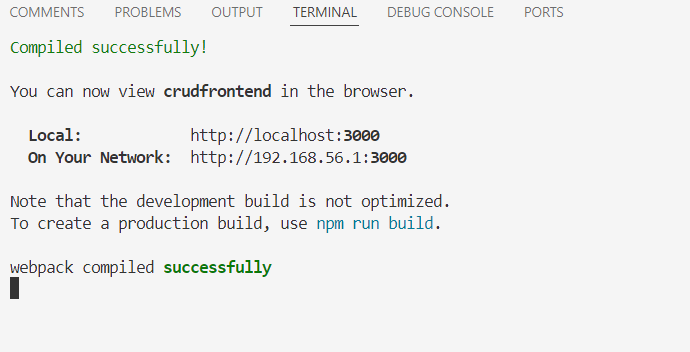
    }

}

Backend and Frontend started

A close-up of a computer screen

Description automatically generated



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer code

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated