**Aim:**

**"To provide an intuitive, lightweight, and efficient web-based Integrated Development Environment (IDE) for developers to seamlessly code, test, and debug HTML, CSS, and JavaScript projects, fostering creativity and productivity in web development."**

**Tech Used:** (MERN Stack)

**MERN Stack Overview:**

* **MERN Definition**: MERN stands for MongoDB, Express.js, React.js, and Node.js, forming a full-stack JavaScript framework.
* **Database (MongoDB)**: A NoSQL database for storing data as JSON-like documents, ensuring scalability and flexibility.
* **Backend Framework (Express.js)**: Simplifies building robust APIs and server-side functionality.
* **Frontend Library (React.js)**: Used for creating dynamic, user-friendly interfaces with reusable components.
* **Server Runtime (Node.js)**: Executes JavaScript code on the server, enabling fast and efficient backend operations.
* **End-to-End JS**: Allows developers to use JavaScript for both frontend and backend, streamlining development.
* **Scalable**: Perfect for building scalable applications with modular architecture.
* **Popularity**: Widely adopted due to ease of learning, extensive community support, and open-source resources.
* **Flexibility**: Supports both small and large-scale application development.
* **Tool Ecosystem**: Compatible with modern tools like Vite, enhancing productivity.

Important Installation:

**Frontend (React with Vite)**

**Libraries Used:**

1. **React**: Core library for building UI.
   * Installation: npm install react react-dom
2. **React Router**: For managing routes.
   * Installation: npm install react-router-dom
3. **Axios**: For making HTTP requests to the backend.
   * Installation: npm install axios
4. **Tailwind CSS**: For styling (optional, can replace with another CSS framework).
   * Installation: npm install -D tailwindcss postcss autoprefixer

npx tailwindcss init

1. **Vite**: Development tool for building React apps (already included with Vite setup).

**Vite Project Setup:**

npm create vite@latest my-project --template react

cd my-project

npm install

**Backend (Node.js with Express)**

**Libraries Used:**

1. **Express.js**: Backend framework.
   * Installation: npm install express
2. **Cors**: To handle cross-origin resource sharing.
   * Installation: npm install cors
3. **Body-Parser**: For parsing JSON and URL-encoded data.
   * Installation: npm install body-parser
4. **Dotenv**: For environment variable management.
   * Installation: npm install dotenv
5. **Mongoose**: For MongoDB interaction.
   * Installation: npm install mongoose
6. **Nodemon**: For automatic server restarting during development.
   * Installation (Development Dependency): npm install -D nodemon

**Database (MongoDB)**

**Libraries Used:**

1. **Mongoose**: Object Data Modeling (ODM) library for MongoDB.
   * Installation: npm install mongoose
2. **MongoDB Driver** (optional, if not using Mongoose): Official MongoDB client.
   * Installation: npm install mongodb

Implementation:

Frontend (React file)

App.js

import React, {useState} from 'react'

import { BrowserRouter, Routes, Route, Navigate } from "react-router-dom";

import "./App.css"

import Home from './pages/Home';

import NoPage from './pages/NoPage';

import SignUp from './pages/SignUp';

import Login from './pages/Login';

import Editior from './pages/Editior';

const App = () => {

  let isLoggedIn = localStorage.getItem("isLoggedIn");

  return (

    <>

      <BrowserRouter>

        <Routes>

          <Route path='/' element={isLoggedIn ? <Home /> : <Navigate to="/login"/>} />

          <Route path='/signUp' element={<SignUp />} />

          <Route path='/login' element={<Login />} />

          <Route path='/editior/:projectID' element={isLoggedIn ? <Editior /> : <Navigate to="/login"/>} />

          <Route path="\*" element={isLoggedIn ? <NoPage />: <Navigate to="/login"/>} />

        </Routes>

      </BrowserRouter>

    </>

  )

}

export default App

App.css

\*{

  margin: 0;

  padding: 0;

}

body{

  margin: 0;

  padding: 0;

  box-sizing: border-box;

  background-color: #0D0C0C;

  color: #fff;

  overflow-x: hidden;

}

html{

  overflow-x: hidden;

}

.inputBox{

  width: 100%;

  background-color: #141414;

  border-radius: 5px;

  display: flex;

  align-items: center;

  margin-bottom: 15px;

}

.inputBox input{

  flex: 1;

  padding: 10px;

  border: none;

  outline: none;

  background-color: transparent;

  color: #fff;

  font-size: 16px;

}

.btnBlue{

  background-color: #00AEEF;

  color: #fff;

  padding: 10px 20px;

  border: none;

  border-radius: 5px;

  text-align: center;

  cursor: pointer;

  font-size: 16px;

}

.btnBlue:hover{

  background-color: #0086b3;

}

.grid{

  display: flex !important;

  flex-wrap: wrap !important;

  gap: 10px;

}

.lightMode{

  background-color: #fff !important;

  color: #000 !important;

};

.lightMode .navbar{

  background-color: #fff;

}

.lightMode .tabs{

  background-color: #f7f7f7;

}

.lightMode .tabs .tab{

  background-color: #c2c1c1;

  color: #000;

}

.lightMode .btn{

  background-color: #C2C1C1;

  color: #000;

}

Src/compoents/NavbarEditor.jsx

import React from 'react'

import logo from "../images/logo.png"

import { FiDownload } from "react-icons/fi";

import { FaRegFileCode } from "react-icons/fa";

import { FaFileCode } from "react-icons/fa";

import { Link } from 'react-router-dom';

import ListCard from './ListCard';

const EditiorNavbar = () => {

  return (

    <>

     <div className="EditorNavbar flex items-center justify-between px-[100px] h-[80px] bg-[#141414]">

        {/\* Logo Section \*/}

        <div className="logo">

          <Link to="/">

            <img className="w-[60px] cursor-pointer" src={logo} alt="Logo" />

          </Link>

        </div>

        <p>File / <span className='text-[gray]'>Project</span></p>

        <i className='p-[8px] btn bg-black rounded-[5px] cursor-pointer text-[20px]'><FaFileCode /></i>

      </div>

    </>

  )

}

export default EditiorNavbar

src/components/Navbar.jsx

import React, { useEffect, useState } from 'react'

import logo from "../images/logo.png"

import { Link, useNavigate } from 'react-router-dom'

import Avatar from 'react-avatar';

import { MdLightMode } from "react-icons/md";

import { BsGridFill } from "react-icons/bs";

import { api\_base\_url, toggleClass } from '../helper';

const Navbar = ({ isGridLayout, setIsGridLayout }) => {

  const navigate = useNavigate();

  const [data, setData] = useState(null);

  const [error, setError] = useState("");;

  useEffect(() => {

    fetch(api\_base\_url + "/getUserDetails", {

      mode: "cors",

      method: "POST",

      headers: {

        "Content-Type": "application/json",

      },

      body: JSON.stringify({

        userId: localStorage.getItem("userId")

      })

    }).then(res => res.json()).then(data => {

      if (data.success) {

        setData(data.user);

      }

      else {

        setError(data.message);

      }

    })

  }, [])

  const logout = () => {

    localStorage.removeItem("userId");

    localStorage.removeItem("token");

    localStorage.removeItem("isLoggedIn");

    window.location.reload();

  }

  return (

    <>

      <div className="navbar flex items-center justify-between px-[100px] h-[80px] bg-[#141414]">

        <div className="logo">

          <img className='w-[70px] cursor-pointer' src={logo} alt="" />

        </div>

        <div className="links flex items-center gap-2">

          {/\* <Link>Home</Link>

          <Link>About</Link>

          <Link>Contact</Link>

          <Link>Services</Link> \*/}

          <button onClick={logout} className='btnBlue !bg-red-500 min-w-[120px] ml-2 hover:!bg-red-600'>Logout</button>

          <Avatar onClick={() => { toggleClass(".dropDownNavbar", "hidden") }} name={data ? data.name : ""} size="40" round="50%" className=' cursor-pointer ml-2' />

        </div>

        <div className='dropDownNavbar hidden absolute right-[60px] top-[80px] shadow-lg shadow-black/50 p-[10px] rounded-lg bg-[#1A1919] w-[150px] h-[160px]'>

          <div className='py-[10px] border-b-[1px] border-b-[#fff]'>

            <h3 className='text-[17px]' style={{ lineHeight: 1 }}>{data ? data.name : ""}</h3>

          </div>

          <i className='flex items-center gap-2 mt-3 mb-2 cursor-pointer' style={{ fontStyle: "normal" }}><MdLightMode className='text-[20px]' /> Light mode</i>

          <i onClick={() => setIsGridLayout(!isGridLayout)} className='flex items-center gap-2 mt-3 mb-2 cursor-pointer' style={{ fontStyle: "normal" }}><BsGridFill className='text-[20px]' /> {isGridLayout ? "List" : "Grid"} layout</i>

        </div>

      </div>

    </>

  )

}

export default Navbar

src/components/ListCard.jsx

import React, { useState } from 'react'

import img from "../images/code.png"

import deleteImg from "../images/delete.png"

import { api\_base\_url } from '../helper';

import { useNavigate } from 'react-router-dom';

const ListCard = ({item}) => {

  const navigate = useNavigate();

  const [isDeleteModelShow, setIsDeleteModelShow] = useState(false);

  const deleteProj = (id) => {

    fetch(api\_base\_url + "/deleteProject",{

      mode: "cors",

      method: "POST",

      headers: {

        "Content-Type": "application/json",

      },

      body: JSON.stringify({

        progId: id,

        userId: localStorage.getItem("userId")

      })

    }).then(res=>res.json()).then(data=>{

      if(data.success){

        setIsDeleteModelShow(false)

        window.location.reload()

      }else{

        alert(data.message)

        setIsDeleteModelShow(false)

      }

    })

  }

  return (

    <>

      <div className="listCard mb-2 w-[full] flex items-center justify-between p-[10px] bg-[#141414] cursor-pointer rounded-lg hover:bg-[#202020]">

        <div onClick={()=>{navigate(`/editior/${item.\_id}`)}} className='flex items-center gap-2'>

          <img className='w-[80px]' src={img} alt="" />

          <div>

            <h3 className='text-[20px]'>{item.title}</h3>

            <p className='text-[gray] text-[14px]'>Created in {new Date(item.date).toDateString()}</p>

          </div>

        </div>

        <div>

          <img onClick={()=>{setIsDeleteModelShow(true)}} className='w-[30px] cursor-pointer mr-4' src={deleteImg} alt="" />

        </div>

      </div>

      {

        isDeleteModelShow ? <div className="model fixed top-0 left-0 w-screen h-screen bg-[rgba(0,0,0,0.1)] flex justify-center items-center flex-col">

          <div className="mainModel w-[25vw] h-[25vh] bg-[#141414] rounded-lg p-[20px]">

            <h3 className='text-3xl'>Do you want to delete <br />

              this project</h3>

            <div className='flex w-full mt-5 items-center gap-[10px]'>

              <button onClick={()=>{deleteProj(item.\_id)}} className='p-[10px] rounded-lg bg-[#FF4343] text-white cursor-pointer min-w-[49%]'>Delete</button>

              <button onClick={()=>{setIsDeleteModelShow(false)}} className='p-[10px] rounded-lg bg-[#1A1919] text-white cursor-pointer min-w-[49%]'>Cancel</button>

            </div>

          </div>

        </div> : ""

      }

    </>

  )

}

export default ListCard

src/components/GridList.jsx

import React, { useState } from 'react'

import deleteImg from "../images/delete.png"

import codeImg from "../images/code.png"

import { useNavigate } from 'react-router-dom';

const GridCard = ({item}) => {

  const [isDeleteModelShow, setIsDeleteModelShow] = useState(false);

  const navigate = useNavigate();

  return (

    <>

      <div className="gridCard bg-[#141414] w-[270px] p-[10px] h-[180px] cursor-pointer hover:bg-[#202020] rounded-lg shadow-lg shadow-black/50">

       <div onClick={()=>{navigate(`/editior/${item.\_id}`)}}>

        <img className="w-[90px]" src={codeImg} alt="" />

        <h3 className='text-[20px] w-[90%] line-clamp-1'>{item.title}</h3>

       </div>

        <div className='flex items-center justify-between'>

          <p className='text-[14px] text-[gray]'>Created in {new Date(item.date).toDateString()}</p>

          <img onClick={()=>{setIsDeleteModelShow(true)}} className='w-[30px] cursor-pointer' src={deleteImg} alt="" />

        </div>

      </div>

      {

        isDeleteModelShow ? <div className="model fixed top-0 left-0 w-screen h-screen bg-[rgba(0,0,0,0.1)] flex justify-center items-center flex-col">

          <div className="mainModel w-[25vw] h-[25vh] bg-[#141414] rounded-lg p-[20px]">

            <h3 className='text-3xl'>Do you want to delete <br />

              this project</h3>

            <div className='flex w-full mt-5 items-center gap-[10px]'>

              <button className='p-[10px] rounded-lg bg-[#FF4343] text-white cursor-pointer min-w-[49%]'>Delete</button>

              <button onClick={()=>{setIsDeleteModelShow(false)}} className='p-[10px] rounded-lg bg-[#1A1919] text-white cursor-pointer min-w-[49%]'>Cancel</button>

            </div>

          </div>

        </div> : ""

      }

    </>

  )

}

export default GridCard

src/pages/Editor.jsx

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

import EditiorNavbar from '../components/EditiorNavbar';

import Editor from '@monaco-editor/react';

import { MdLightMode } from 'react-icons/md';

import { AiOutlineExpandAlt } from "react-icons/ai";

import { api\_base\_url } from '../helper';

import { useParams } from 'react-router-dom';

const Editior = () => {

  const [tab, setTab] = useState("html");

  const [isLightMode, setIsLightMode] = useState(false);

  const [isExpanded, setIsExpanded] = useState(false);

  const [htmlCode, setHtmlCode] = useState("<h1>Hello world</h1>");

  const [cssCode, setCssCode] = useState("body { background-color: #f4f4f4; }");

  const [jsCode, setJsCode] = useState("// some comment");

  // Extract projectID from URL using useParams

  const { projectID } = useParams();

  const changeTheme = () => {

    const editorNavbar = document.querySelector(".EditiorNavbar");

    if (isLightMode) {

      editorNavbar.style.background = "#141414";

      document.body.classList.remove("lightMode");

      setIsLightMode(false);

    } else {

      editorNavbar.style.background = "#fff";

      document.body.classList.add("lightMode");

      setIsLightMode(true);

    }

  };

  const run = () => {

    const html = htmlCode;

    const css = `<style>${cssCode}</style>`;

    const js = `<script>${jsCode}</script>`;

    const iframe = document.getElementById("iframe");

    if (iframe) {

      iframe.srcdoc = html + css + js;

    }

  };

  useEffect(() => {

    setTimeout(() => {

      run();

    }, 200);

  }, [htmlCode, cssCode, jsCode]);

  useEffect(() => {

    fetch(api\_base\_url + "/getProject", {

      mode: "cors",

      method: "POST",

      headers: {

        "Content-Type": "application/json"

      },

      body: JSON.stringify({

        userId: localStorage.getItem("userId"),

        projId: projectID // Use projectID here

      })

    })

      .then(res => res.json())

      .then(data => {

        setHtmlCode(data.project.htmlCode);

        setCssCode(data.project.cssCode);

        setJsCode(data.project.jsCode);

      });

  }, [projectID]);

  useEffect(() => {

    const handleKeyDown = (event) => {

      if (event.ctrlKey && event.key === 's') {

        event.preventDefault(); // Prevent the default save file dialog

        // Ensure that projectID and code states are updated and passed to the fetch request

        fetch(api\_base\_url + "/updateProject", {

          mode: "cors",

          method: "POST",

          headers: {

            "Content-Type": "application/json"

          },

          body: JSON.stringify({

            userId: localStorage.getItem("userId"),

            projId: projectID,  // Make sure projectID is correct

            htmlCode: htmlCode,  // Passing the current HTML code

            cssCode: cssCode,    // Passing the current CSS code

            jsCode: jsCode       // Passing the current JS code

          })

        })

        .then(res => res.json())

        .then(data => {

          if (data.success) {

            alert("Project saved successfully");

          } else {

            alert("Something went wrong");

          }

        })

        .catch((err) => {

          console.error("Error saving project:", err);

          alert("Failed to save project. Please try again.");

        });

      }

    };

    window.addEventListener('keydown', handleKeyDown);

    // Clean up the event listener on component unmount

    return () => {

      window.removeEventListener('keydown', handleKeyDown);

    };

  }, [projectID, htmlCode, cssCode, jsCode]);

  return (

    <>

      <EditiorNavbar />

      <div className="flex">

        <div className={`left w-[${isExpanded ? "100%" : "50%"}]`}>

          <div className="tabs flex items-center justify-between gap-2 w-full bg-[#1A1919] h-[50px] px-[40px]">

            <div className="tabs flex items-center gap-2">

              <div onClick={() => { setTab("html"); }} className="tab cursor-pointer p-[6px] bg-[#1E1E1E] px-[10px] text-[15px]">HTML</div>

              <div onClick={() => { setTab("css"); }} className="tab cursor-pointer p-[6px] bg-[#1E1E1E] px-[10px] text-[15px]">CSS</div>

              <div onClick={() => { setTab("js"); }} className="tab cursor-pointer p-[6px] bg-[#1E1E1E] px-[10px] text-[15px]">JavaScript</div>

            </div>

            <div className="flex items-center gap-2">

              {/\* <i className="text-[20px] cursor-pointer" onClick={changeTheme}><MdLightMode /></i> \*/}

              <i className="text-[20px] cursor-pointer" onClick={() => { setIsExpanded(!isExpanded); }}><AiOutlineExpandAlt /></i>

            </div>

          </div>

          {tab === "html" ? (

            <Editor

              onChange={(value) => {

                setHtmlCode(value || "");

                run();

              }}

              height="82vh"

              theme={isLightMode ? "vs-light" : "vs-dark"}

              language="html"

              value={htmlCode}

            />

          ) : tab === "css" ? (

            <Editor

              onChange={(value) => {

                setCssCode(value || "");

                run();

              }}

              height="82vh"

              theme={isLightMode ? "vs-light" : "vs-dark"}

              language="css"

              value={cssCode}

            />

          ) : (

            <Editor

              onChange={(value) => {

                setJsCode(value || "");

                run();

              }}

              height="82vh"

              theme={isLightMode ? "vs-light" : "vs-dark"}

              language="javascript"

              value={jsCode}

            />

          )}

        </div>

        {!isExpanded && (

          <iframe

            id="iframe"

            className="w-[50%] min-h-[82vh] bg-[#fff] text-black"

            title="output"

          />

        )}

      </div>

    </>

  );

};

export default Editior;

src/pages/Home.jsx

import React, { useEffect, useState } from 'react'

import Navbar from '../components/Navbar'

import ListCard from '../components/ListCard';

import GridCard from '../components/GridCard';

import { api\_base\_url } from '../helper';

import { useNavigate } from 'react-router-dom';

const Home = () => {

  const [data, setData] = useState(null);

  const [error, setError] = useState("");

  const [searchQuery, setSearchQuery] = useState(''); // State for search query

  const [projTitle, setProjTitle] = useState("");

  const navigate = useNavigate();

  const [isCreateModelShow, setIsCreateModelShow] = useState(false);

  // Filter data based on search query

  const filteredData = data ? data.filter(item =>

    item.title.toLowerCase().includes(searchQuery.toLowerCase()) // Case insensitive filtering

  ) : [];

  const createProj = (e) => {

    if (projTitle === "") {

      alert("Please Enter Project Title");

    } else {

      fetch(api\_base\_url + "/createProject", {

        mode: "cors",

        method: "POST",

        headers: {

          "Content-Type": "application/json",

        },

        body: JSON.stringify({

          title: projTitle,

          userId: localStorage.getItem("userId")

        })

      }).then(res => res.json()).then(data => {

        if (data.success) {

          setIsCreateModelShow(false);

          setProjTitle("");

          alert("Project Created Successfully");

          navigate(`/editior/${data.projectId}`);

        } else {

          alert("Something Went Wrong");

        }

      });

    }

  };

  const getProj = () => {

    fetch(api\_base\_url + "/getProjects", {

      mode: "cors",

      method: "POST",

      headers: {

        "Content-Type": "application/json",

      },

      body: JSON.stringify({

        userId: localStorage.getItem("userId")

      })

    }).then(res => res.json()).then(data => {

      if (data.success) {

        setData(data.projects);

      } else {

        setError(data.message);

      }

    });

  };

  useEffect(() => {

    getProj();

  }, []);

  const [userData, setUserData] = useState(null);

  const [userError, setUserError] = useState("");;

  useEffect(() => {

    fetch(api\_base\_url + "/getUserDetails", {

      mode: "cors",

      method: "POST",

      headers: {

        "Content-Type": "application/json",

      },

      body: JSON.stringify({

        userId: localStorage.getItem("userId")

      })

    }).then(res => res.json()).then(data => {

      if (data.success) {

        setUserData(data.user);

      }

      else {

        setUserError(data.message);

      }

    })

  }, [])

  const [isGridLayout, setIsGridLayout] = useState(false);

  return (

    <>

      <Navbar isGridLayout={isGridLayout} setIsGridLayout={setIsGridLayout} />

      <div className='flex items-center justify-between px-[100px] my-[40px]'>

        <h2 className='text-2xl'>Hi, {userData ? userData.username : ""} 👋</h2>

        <div className='flex items-center gap-1'>

          {/\* Search Bar \*/}

          <div className="inputBox !w-[350px]">

            <input

              type="text"

              placeholder='Search Here... !'

              value={searchQuery} // Bind search input to searchQuery state

              onChange={(e) => setSearchQuery(e.target.value)} // Update searchQuery on input change

            />

          </div>

          <button onClick={() => { setIsCreateModelShow(true) }} className='btnBlue !bg-[#EE4323] rounded-[5px] mb-4 text-[20px] !p-[5px] !px-[10px]'>+</button>

        </div>

      </div>

      {/\* Project Display \*/}

      <div className="cards">

        {

          isGridLayout ?

            <div className='grid px-[100px]'>

              {

                filteredData.length > 0 ? filteredData.map((item, index) => (

                  <GridCard key={index} item={item} />

                )) : <p>No projects found</p>

              }

            </div>

            : <div className='list px-[100px]'>

              {

                filteredData.length > 0 ? filteredData.map((item, index) => (

                  <ListCard key={index} item={item} />

                )) : <p>No projects found</p>

              }

            </div>

        }

      </div>

      {/\* Modal for Creating a New Project \*/}

      {isCreateModelShow &&

        <div className="createModelCon fixed top-0 left-0 right-0 bottom-0 w-screen h-screen bg-[rgb(0,0,0,0.1)] flex items-center justify-center">

          <div className="createModel w-[25vw] h-[27vh] shadow-lg shadow-black/50 bg-[#141414] rounded-[10px] p-[20px]">

            <h3 className='text-2xl'>Create New Project</h3>

            <div className="inputBox !bg-[#202020] mt-4">

              <input

                onChange={(e) => { setProjTitle(e.target.value) }}

                value={projTitle}

                type="text"

                placeholder='Project Title'

              />

            </div>

            <div className='flex items-center gap-[10px] w-full mt-2'>

              <button onClick={createProj} className='btnBlue !bg-[#EE4323] rounded-[5px] w-[49%] mb-4 !p-[5px] !px-[10px] !py-[10px]'>Create</button>

              <button onClick={() => { setIsCreateModelShow(false) }} className='btnBlue !bg-[#1A1919] rounded-[5px] mb-4 w-[49%] !p-[5px] !px-[10px] !py-[10px]'>Cancel</button>

            </div>

          </div>

        </div>

      }

    </>

  );

}

export default Home;

src/pages/Login.jsx

import React, { useState } from 'react'

import logo from "../images/logo.png"

import { Link, useNavigate } from 'react-router-dom';

import image from "../images/authP.jpg";

import { api\_base\_url } from '../helper';

const Login = () => {

  const [email, setEmail] = useState("");

  const [pwd, setPwd] = useState("");

  const [error, setError] = useState("");

  const navigate = useNavigate();

  const submitForm = (e) => {

    e.preventDefault();

    fetch(api\_base\_url + "/login",{

      mode: "cors",

      method: "POST",

      headers: {

        "Content-Type": "application/json"

      },

      body: JSON.stringify({

        email: email,

        password: pwd

      })

    }).then(res => res.json()).then(data => {

      if(data.success === true){

        localStorage.setItem("token", data.token);

        localStorage.setItem("isLoggedIn", true);

        localStorage.setItem("userId", data.userId);

        setTimeout(() => {

          window.location.href = "/"

        }, 200);

      } else {

        setError(data.message);

      }

    })

  }

  return (

    <>

      <div className="container w-screen min-h-screen flex items-center justify-between pl-[100px]">

        <div className="left w-[35%]">

          <img className='w-[200px]' src={logo} alt="" />

          <form onSubmit={submitForm} className='w-full mt-[60px]' action="">

            <div className="inputBox">

              <input required onChange={(e)=>{setEmail(e.target.value)}} value={email} type="email" placeholder='Email'/>

            </div>

            <div className="inputBox">

              <input required onChange={(e)=>{setPwd(e.target.value)}} value={pwd} type="password" placeholder='Password'/>

            </div>

            <p className='text-[gray]'>Don't have an account <Link to="/signUp" className='text-[#EE4323]'>Sign Up</Link></p>

            <p className='text-red-500 text-[14px] my-2'>{error}</p>

            <button className="btnBlue !bg-[#EE4323] w-full mt-[20px]">Login</button>

          </form>

        </div>

        <div className="right w-[55%]">

          <img className='h-[100vh] w-[100%] object-cover' src={image} alt="" />

        </div>

      </div>

    </>

  )

}

export default Login

src/pages/Signup.jsx

import React, { useState } from 'react'

import logo from "../images/logo.png"

import { Link, useNavigate } from 'react-router-dom';

import image from "../images/authP.jpg";

import { api\_base\_url } from '../helper';

const SignUp = () => {

  const [username, setUsername] = useState("");

  const [name, setName] = useState("");

  const [email, setEmail] = useState("");

  const [pwd, setPwd] = useState("");

  const [error, setError] = useState("");

  const navigate = useNavigate();

  const submitForm = (e) => {

    e.preventDefault();

    fetch(api\_base\_url + "/signUp",{

      mode: "cors",

      method: "POST",

      headers: {

        "Content-Type": "application/json"

      },

      body: JSON.stringify({

        username: username,

        name: name,

        email: email,

        password: pwd

      })

    }).then((res)=>res.json()).then((data)=>{

      if(data.success === true){

        alert("Account created successfully");

        navigate("/login");

      }

      else{

        setError(data.message);

      }

    })

  }

  return (

    <>

      <div className="container w-screen min-h-screen flex items-center justify-between pl-[100px]">

        <div className="left w-[35%]">

          <img className='w-[200px]' src={logo} alt="" />

          <form onSubmit={submitForm} className='w-full mt-[60px]' action="">

            <div className="inputBox">

              <input required onChange={(e)=>{setUsername(e.target.value)}} value={username} type="text" placeholder='Username'/>

            </div>

            <div className="inputBox">

              <input required onChange={(e)=>{setName(e.target.value)}} value={name} type="text" placeholder='Name'/>

            </div>

            <div className="inputBox">

              <input required onChange={(e)=>{setEmail(e.target.value)}} value={email} type="email" placeholder='Email'/>

            </div>

            <div className="inputBox">

              <input required onChange={(e)=>{setPwd(e.target.value)}} value={pwd} type="password" placeholder='Password'/>

            </div>

            <p className='text-[gray]'>Already have an account <Link to="/login" className='text-[#EE4323]'>login</Link></p>

            <p className='text-red-500 text-[14px] my-2'>{error}</p>

            <button className="btnBlue !bg-[#EE4323] w-full mt-[20px]">Sign Up</button>

          </form>

        </div>

        <div className="right w-[55%]">

          <img className='h-[100vh] w-[100%] object-cover' src={image} alt="" />

        </div>

      </div>

    </>

  )

}

export default SignUp

src/pages/Nopage.jsx

import React from 'react'

const NoPage = () => {

  return (

    <div>NoPage</div>

  )

}

export default NoPage

BackEnd (Node.js and express)

App.js

var createError = require('http-errors');

var express = require('express');

var path = require('path');

var cookieParser = require('cookie-parser');

var logger = require('morgan');

var cors = require("cors");

var indexRouter = require('./routes/index');

var usersRouter = require('./routes/users');

var app = express();

// view engine setup

app.set('views', path.join(\_\_dirname, 'views'));

app.set('view engine', 'ejs');

app.use(logger('dev'));

app.use(express.json());

app.use(express.urlencoded({ extended: false }));

app.use(cookieParser());

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use(cors());

app.use('/', indexRouter);

app.use('/users', usersRouter);

// catch 404 and forward to error handler

app.use(function(req, res, next) {

  next(createError(404));

});

// error handler

app.use(function(err, req, res, next) {

  // set locals, only providing error in development

  res.locals.message = err.message;

  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page

  res.status(err.status || 500);

  res.render('error');

});

module.exports = app;

projectModel.js

const mongoose = require("mongoose");

mongoose.connect('mongodb://127.0.0.1:27017/codeIDE');

const projectSchema = new mongoose.Schema({

  title: String,

  createdBy: String,

  date: {

    type: Date,

    default: Date.now

  },

  htmlCode: {

    type: String,

    default: `

    <!DOCTYPE html>

    <html lang="en">

    <head>

      <meta charset="UTF-8">

      <meta http-equiv="X-UA-Compatible" content="IE=edge">

      <meta name="viewport" content="width=device-width, initial-scale=1.0">

      <title>Document</title>

    </head>

    <body>

    </body>

    </html>`

  },

  cssCode: {

    type: String,

    default: `

    body{

      margin: 0;

      padding: 0;

      box-sizing: border-box;

    }`

  },

  jsCode: {

    type: String,

    default: 'console.log("Hello World")'

  }

});

module.exports = mongoose.model("Project", projectSchema);

userModel.jsx

let mongoose = require('mongoose');

mongoose.connect('mongodb://127.0.0.1:27017/codeIDE');

let userSchema = new mongoose.Schema({

  name: String,

  username: String,

  email: String,

  password: String,

  date:{

    type: Date,

    default: Date.now

  },

  isBlocked: {

    type: Boolean,

    default: false

  },

  isAdmin: {

    type: Boolean,

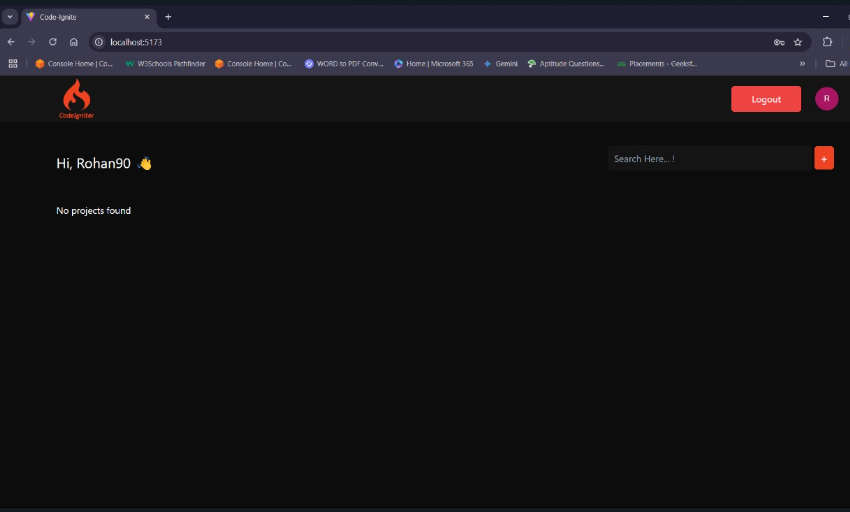
    default: false

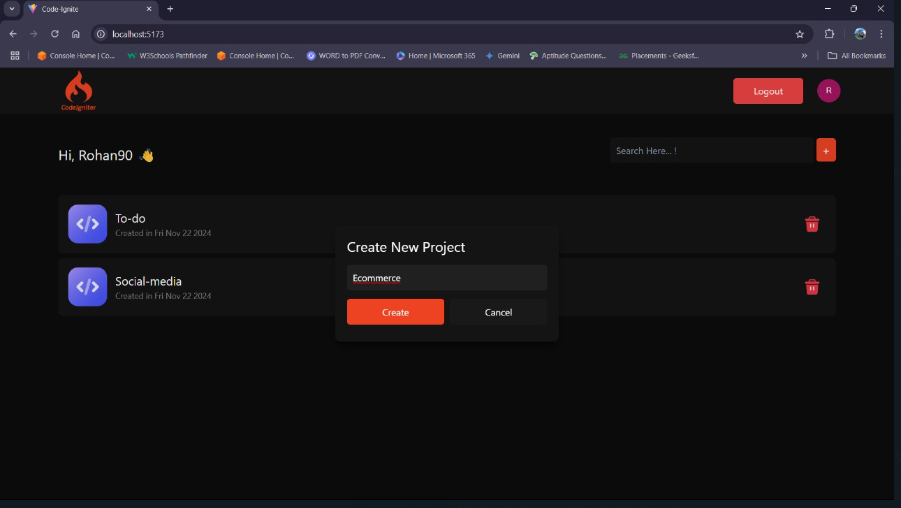
  }

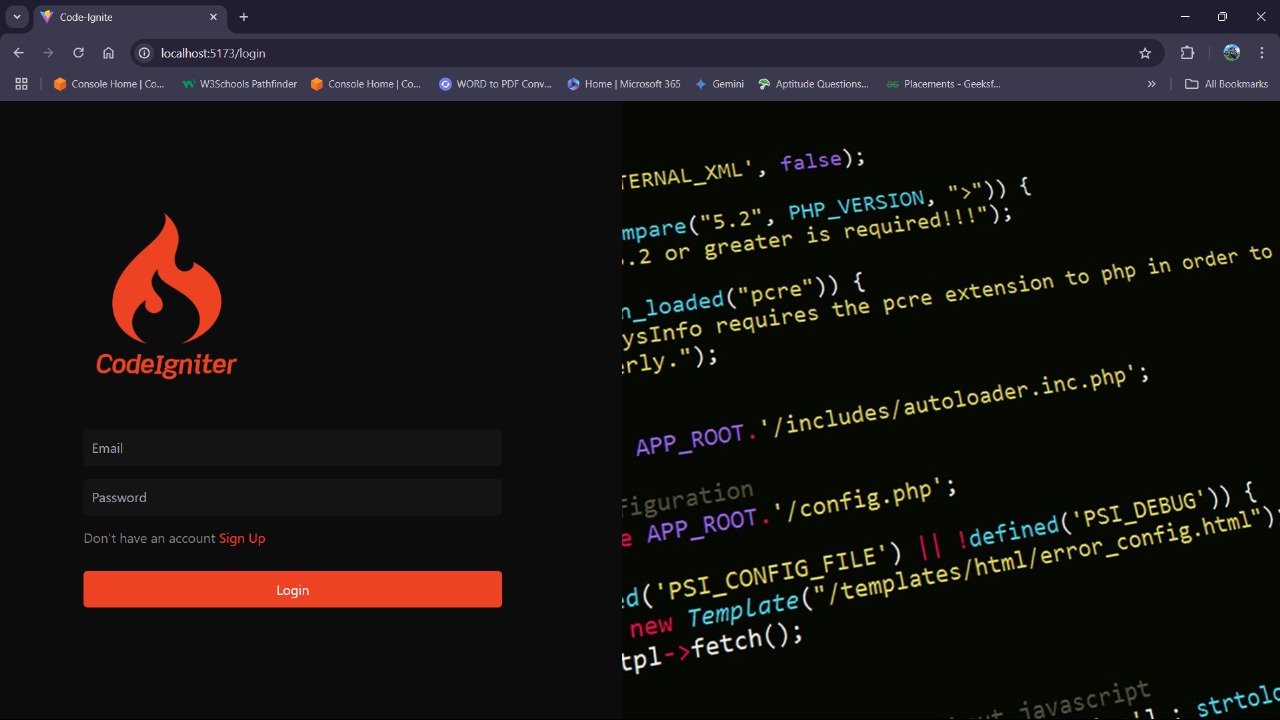
});

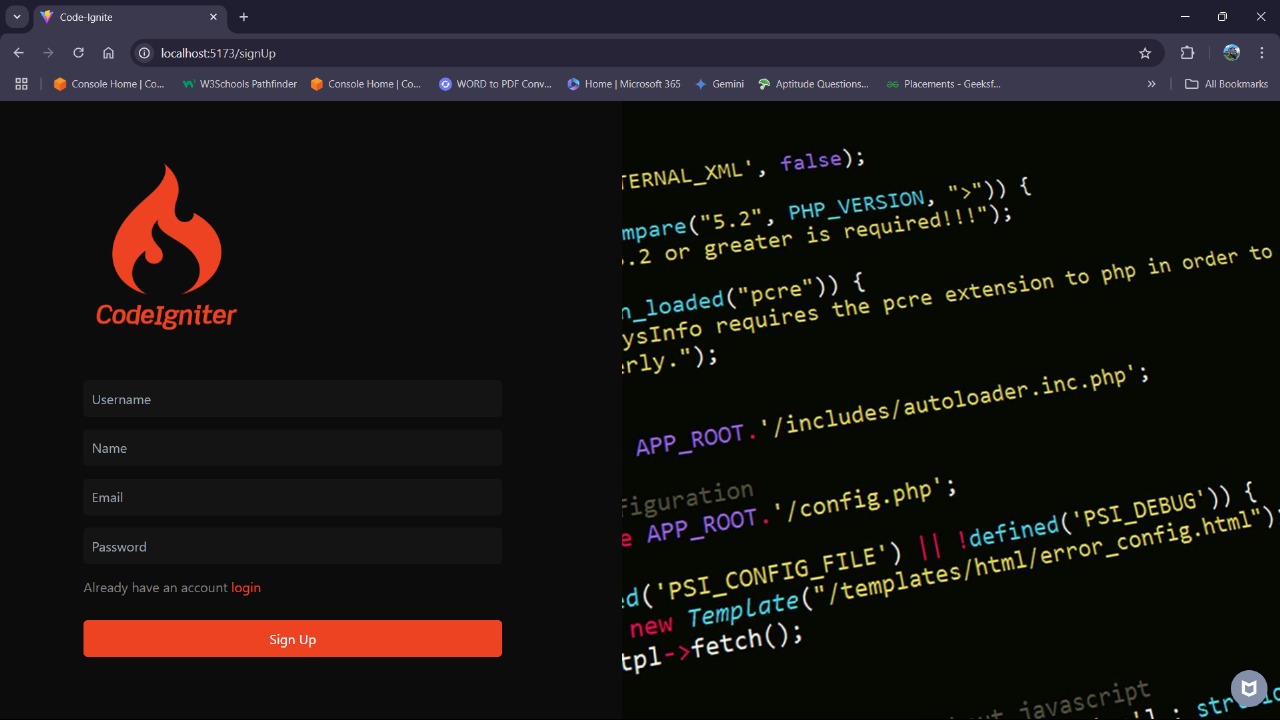
module.exports = mongoose.model('User', userSchema); // 'User' is the name of the collection

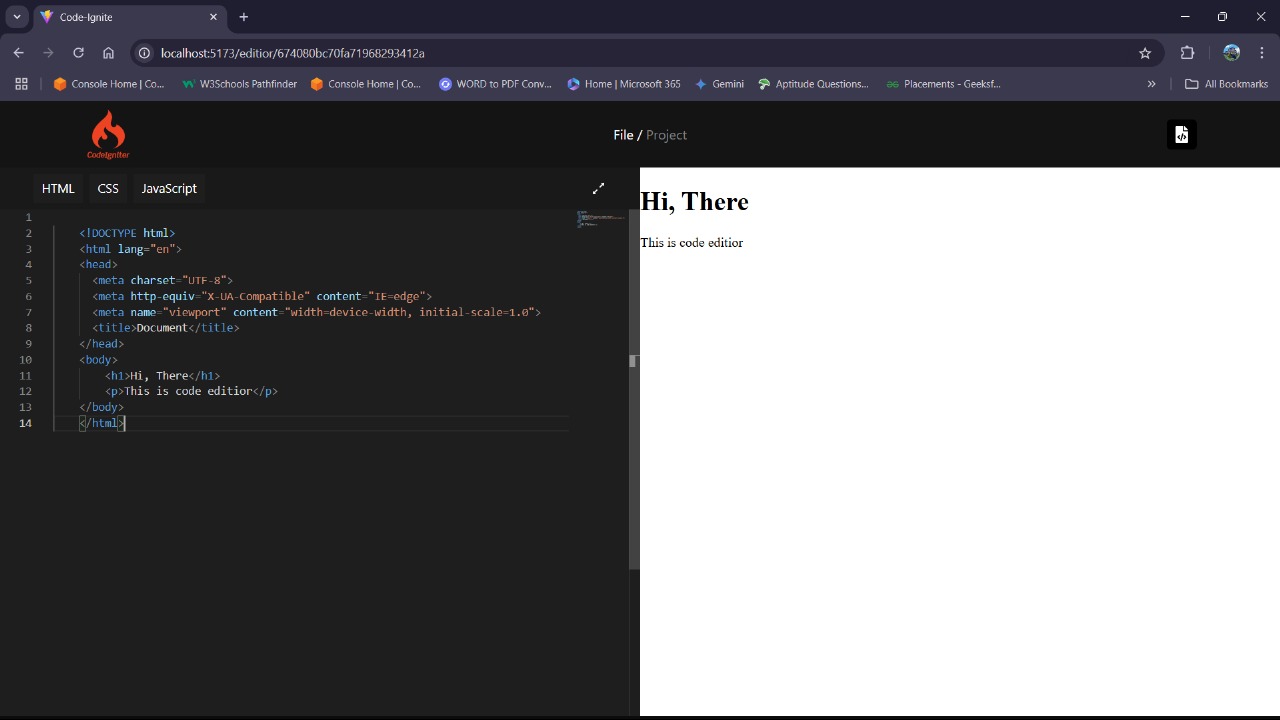
output:

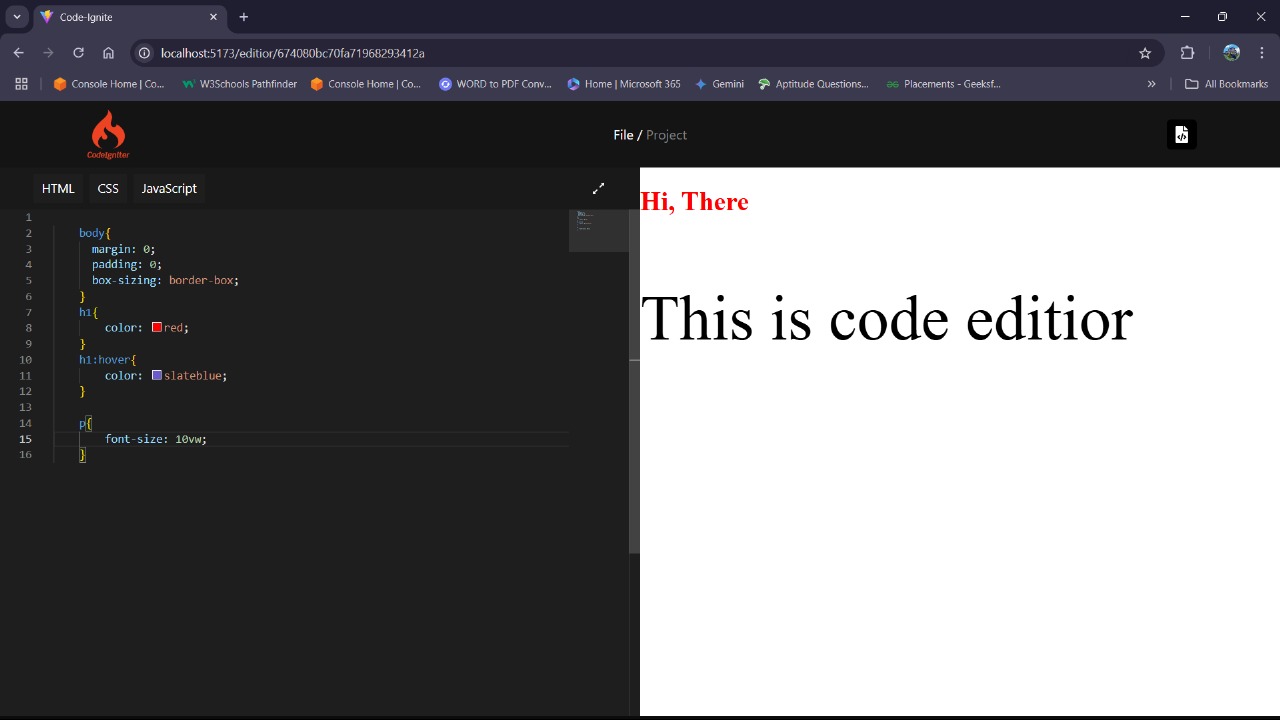


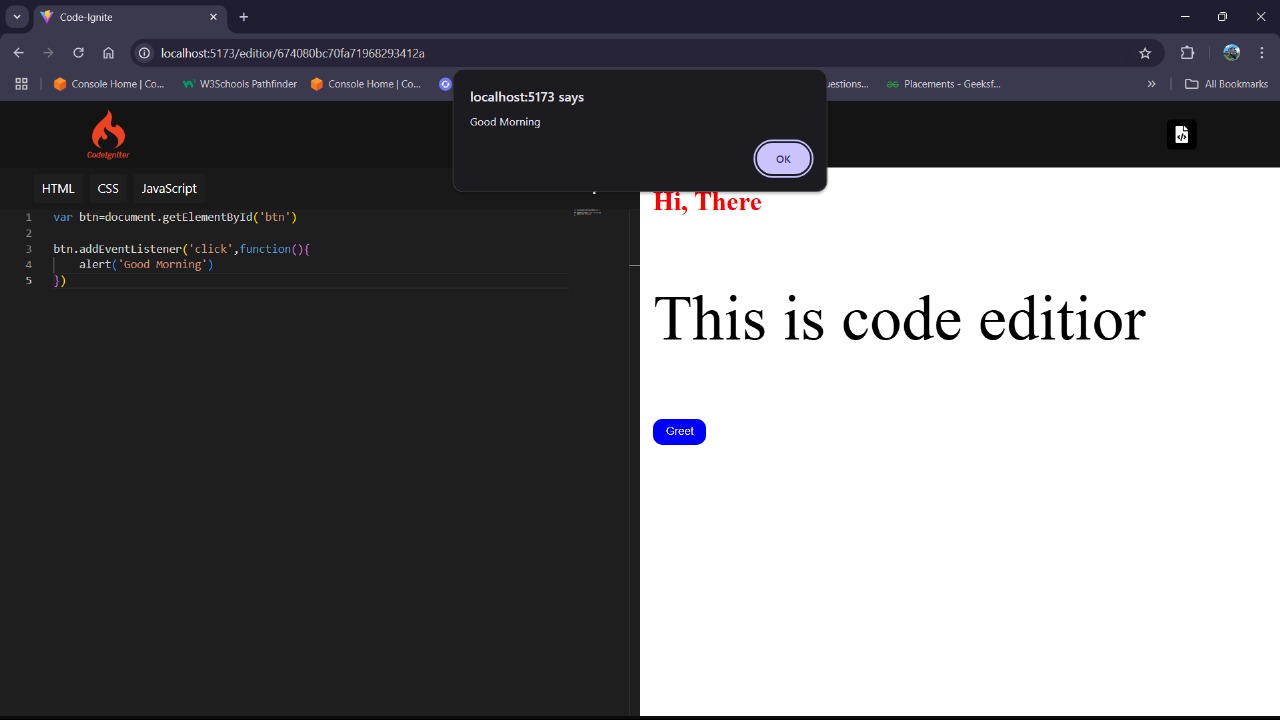




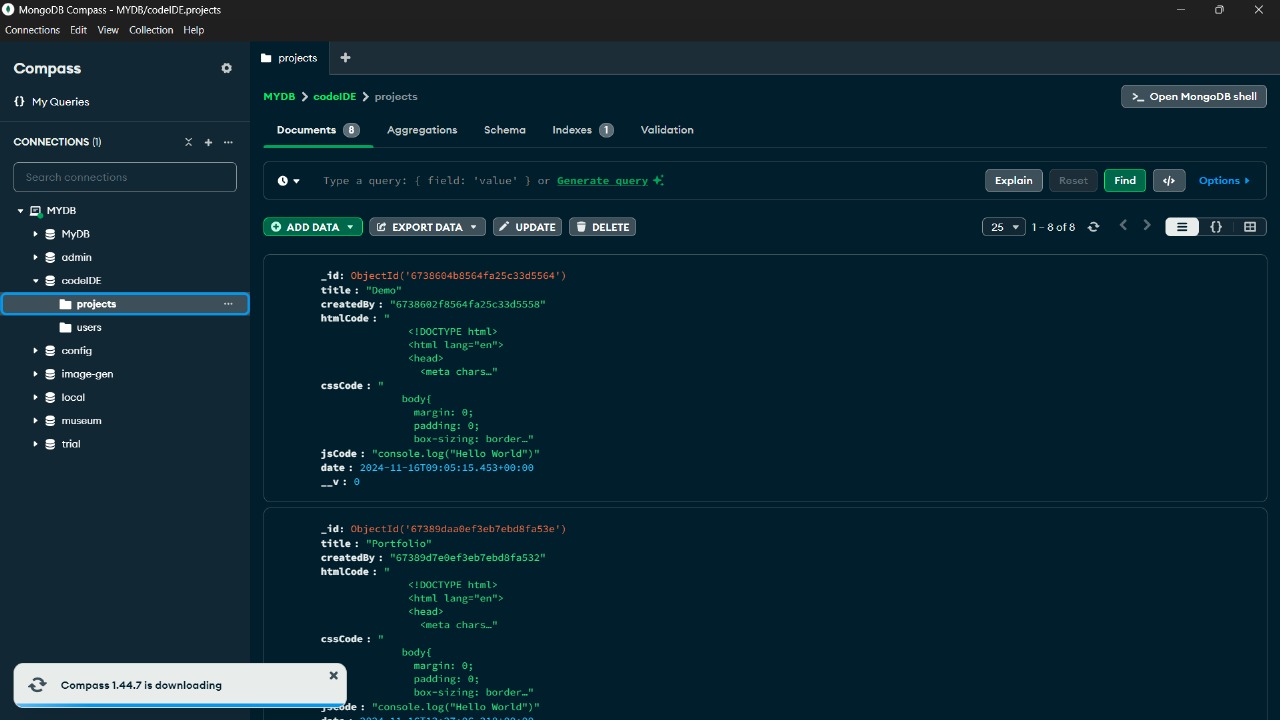








MongoDB Compass: MYDB -> codeIDE -> projects



MongoDB Compass: MYDB -> codeIDE -> users

