**COMSATS University Islamabad**

**Department of Computer Science**



**Assignment 4**

**Subject: Topics in Software Engineering 1**

**Submitted to:**

**Ma’am Humaira Waqas**

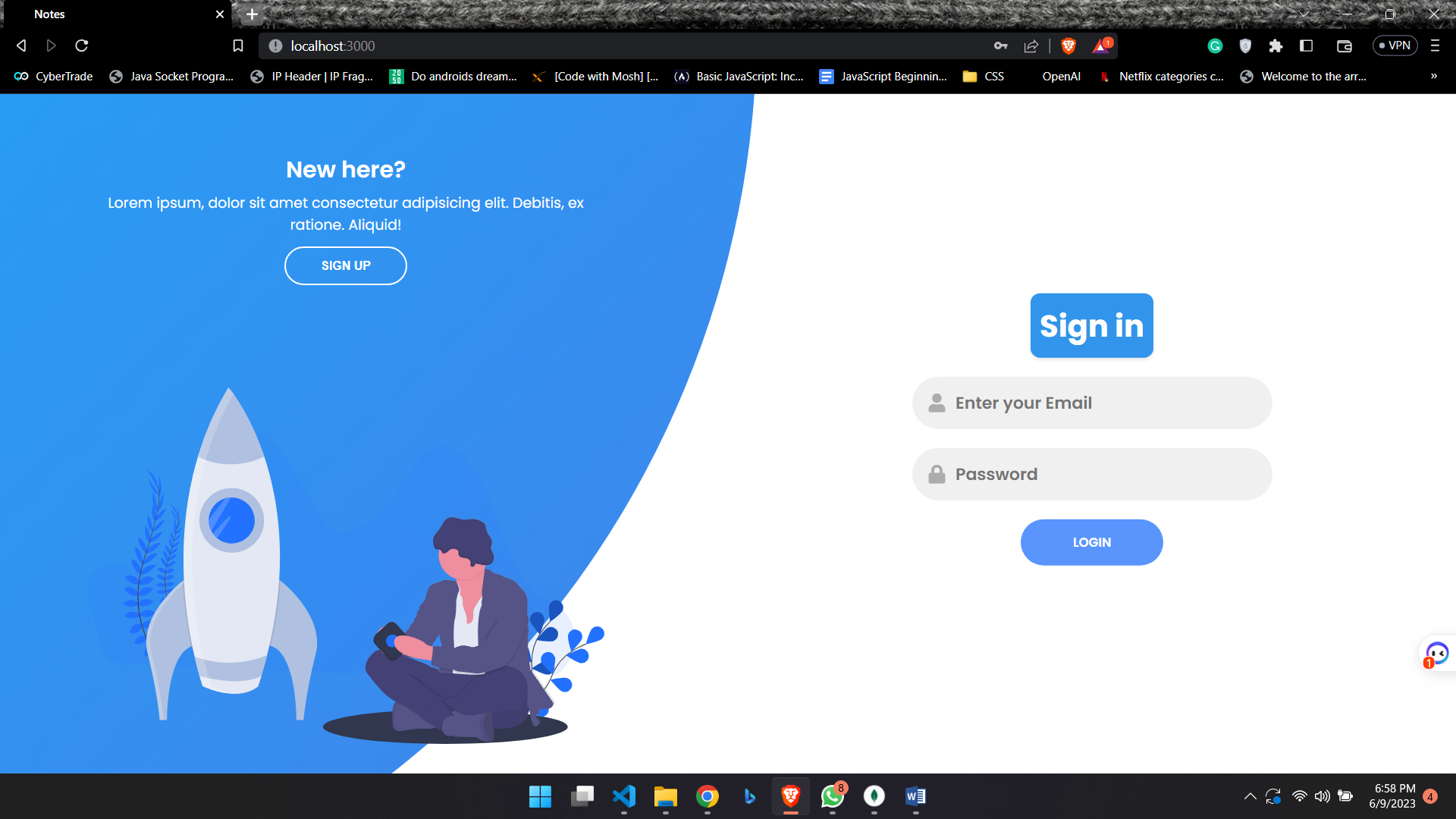
**Submitted by:**

**Abdul Wahab (SP19-BSE-004-8A)**

**Date: Jun 7, 2023**

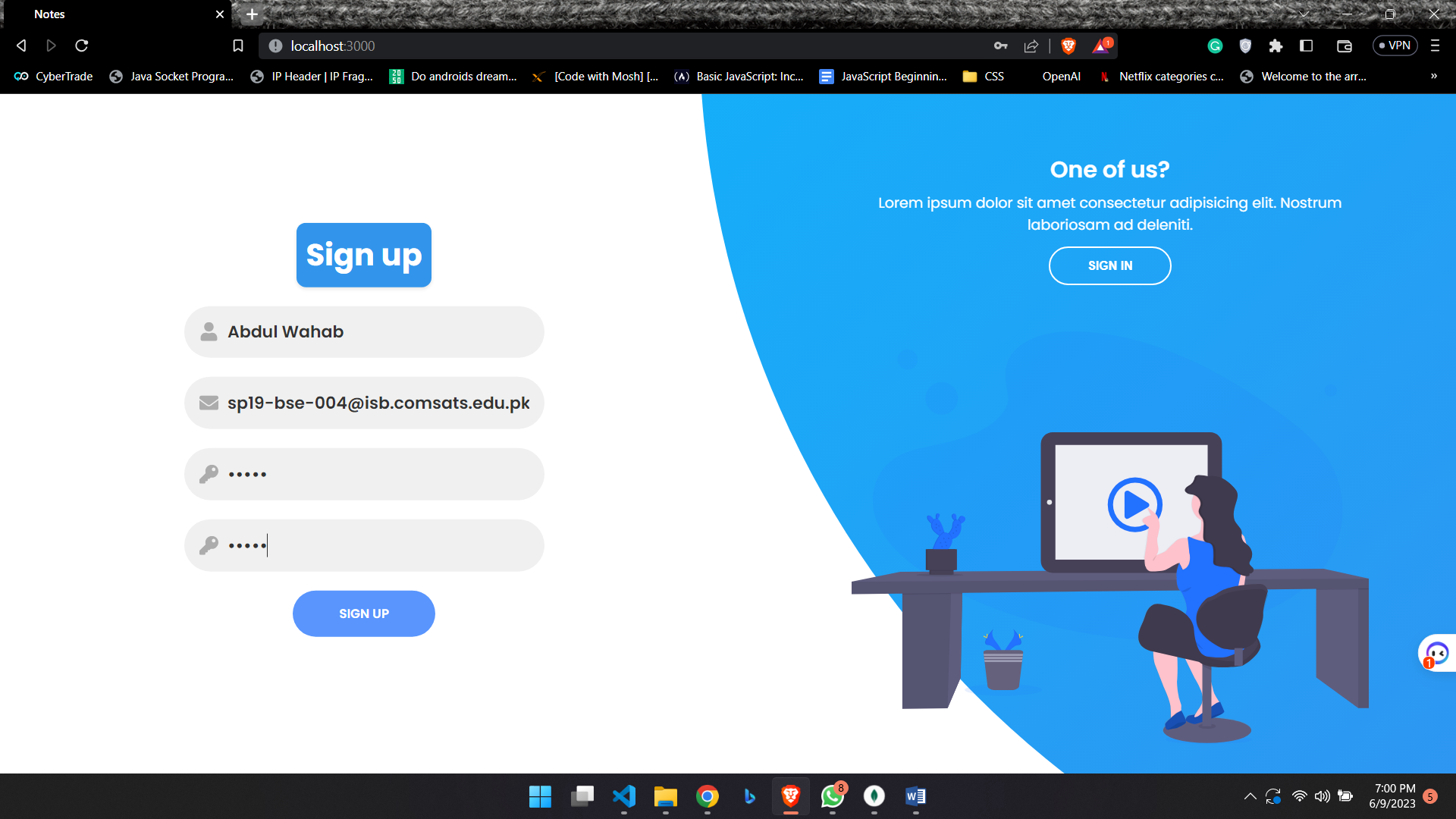
**Outputs**

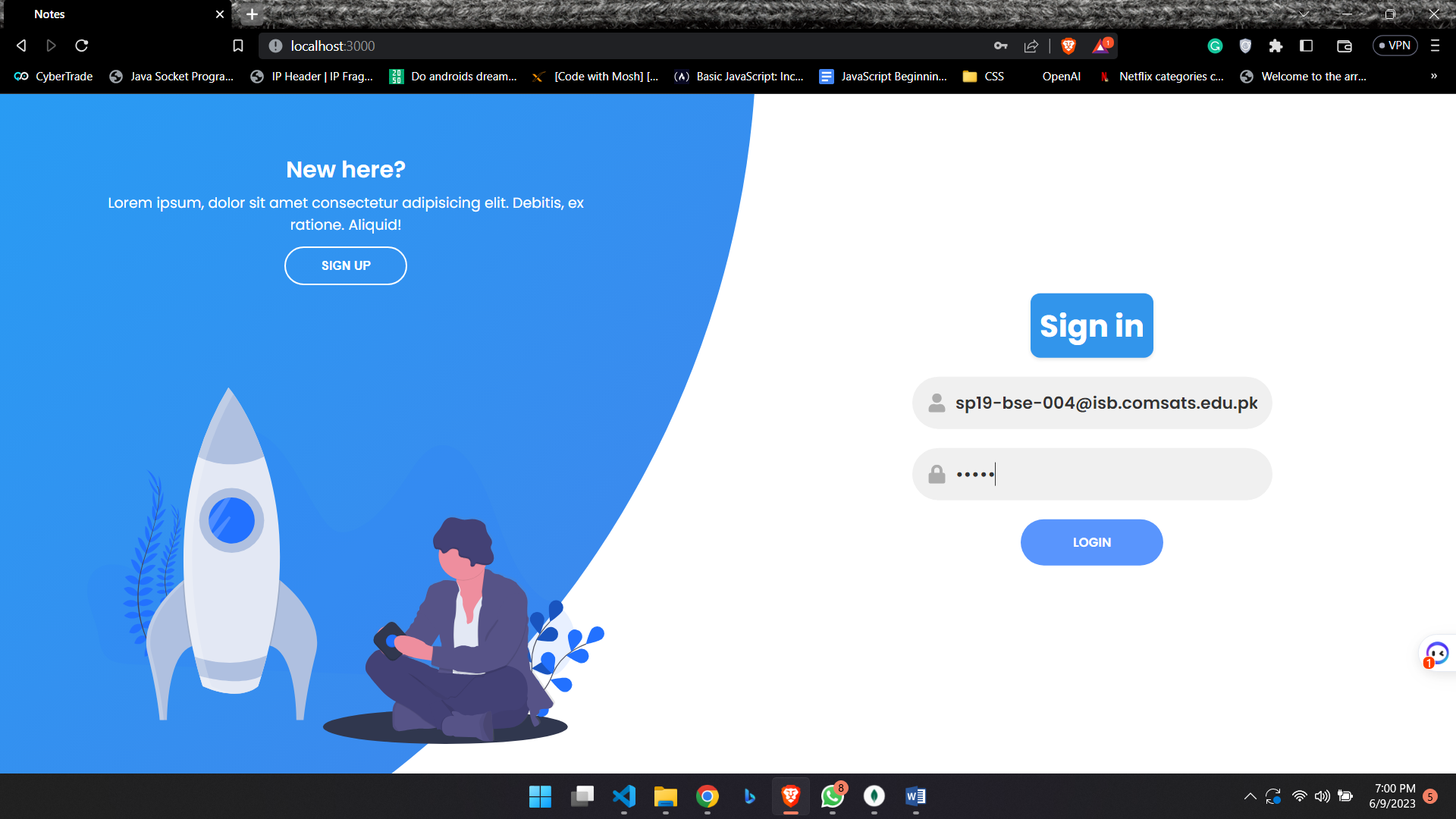
**Sign In**

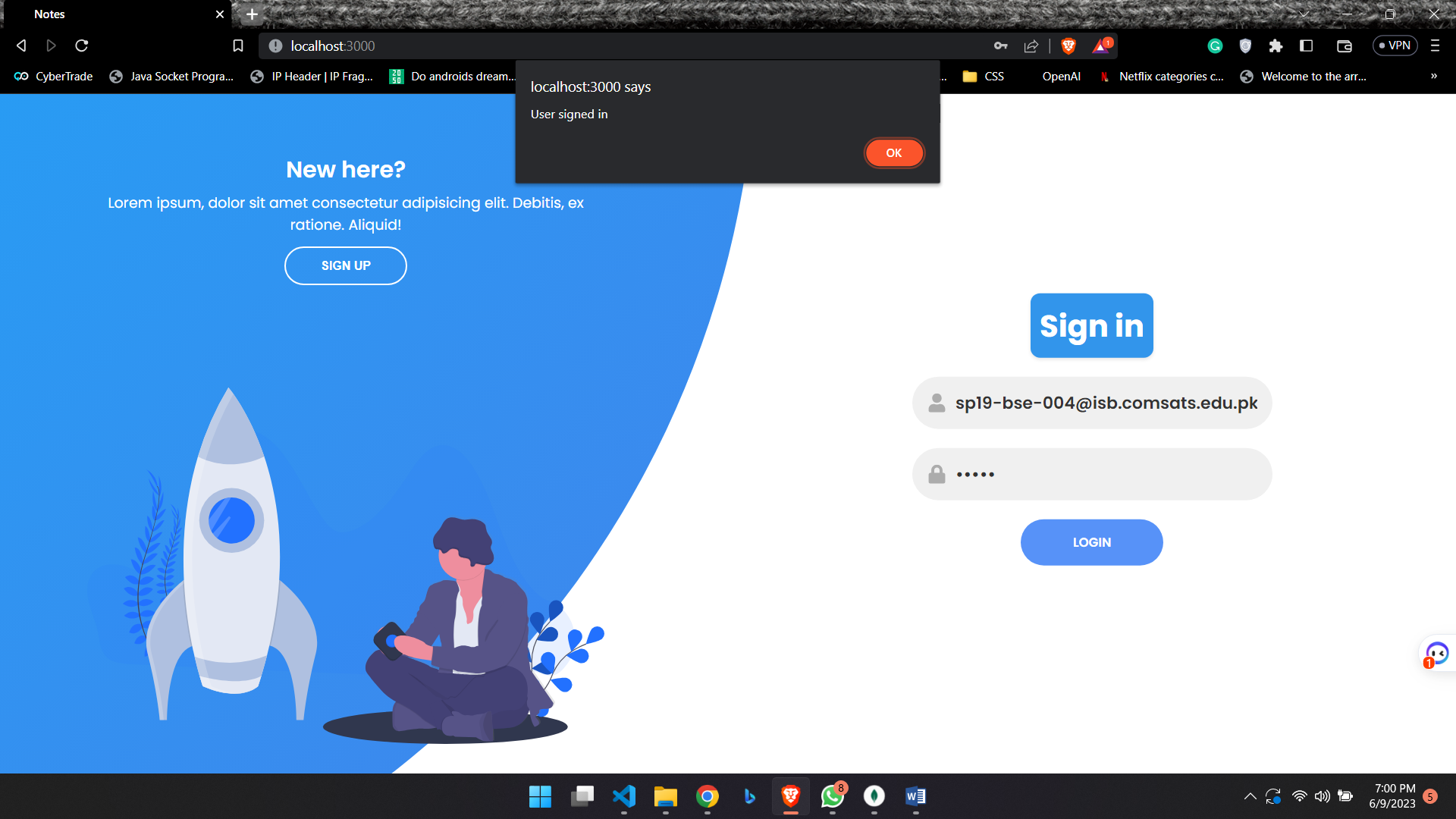
****

**SignUp**

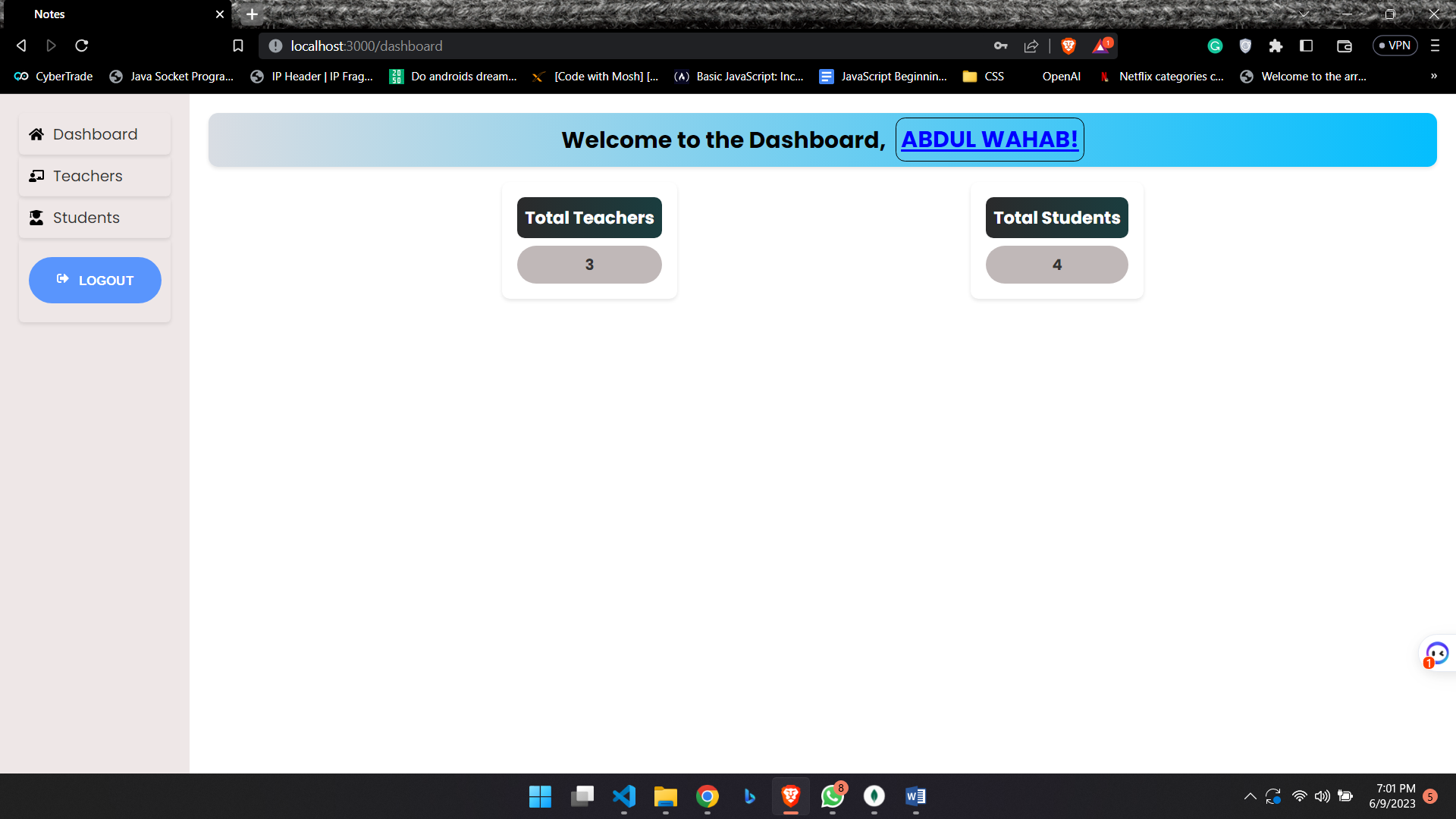
****

****

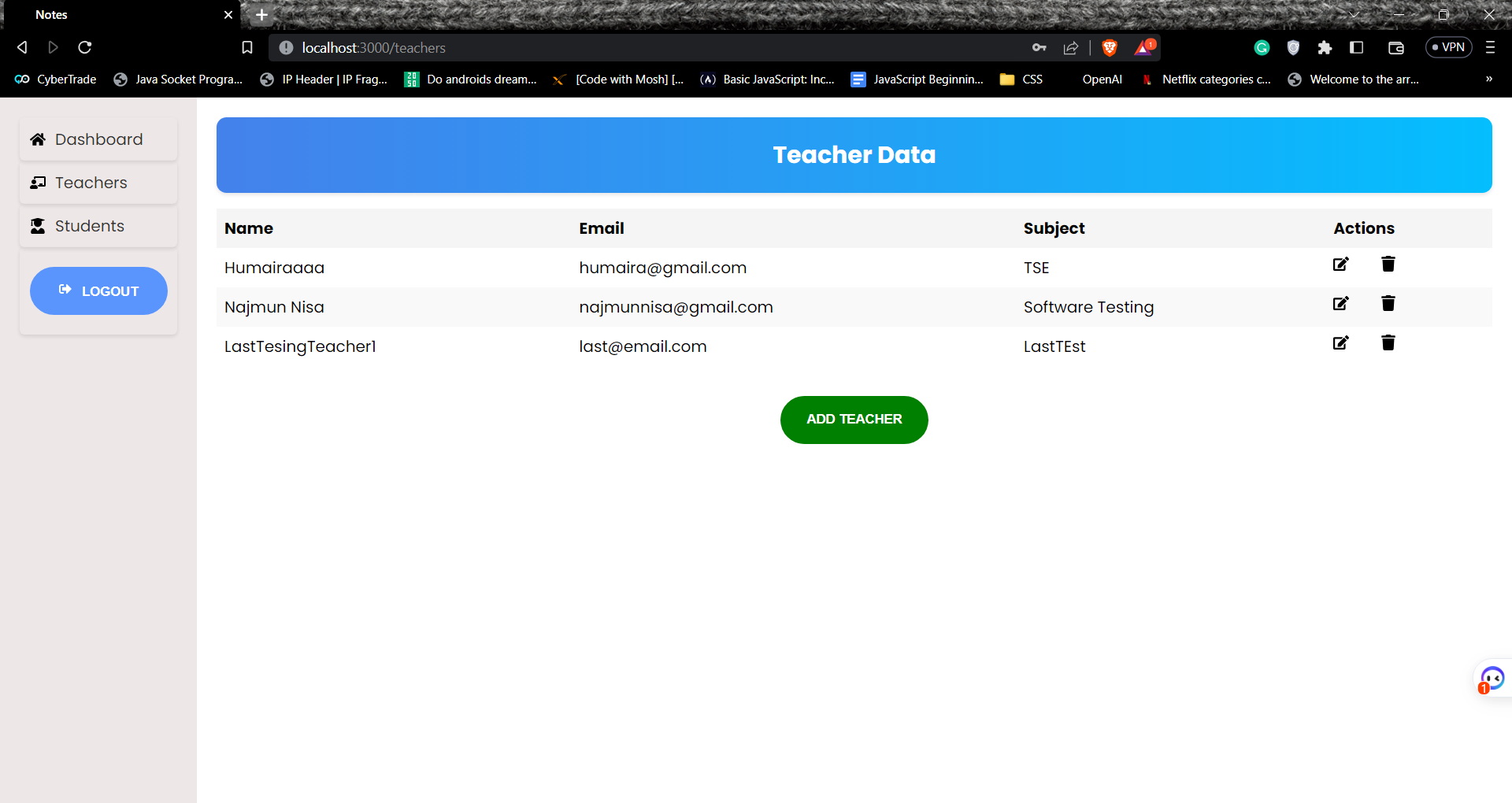
****

****

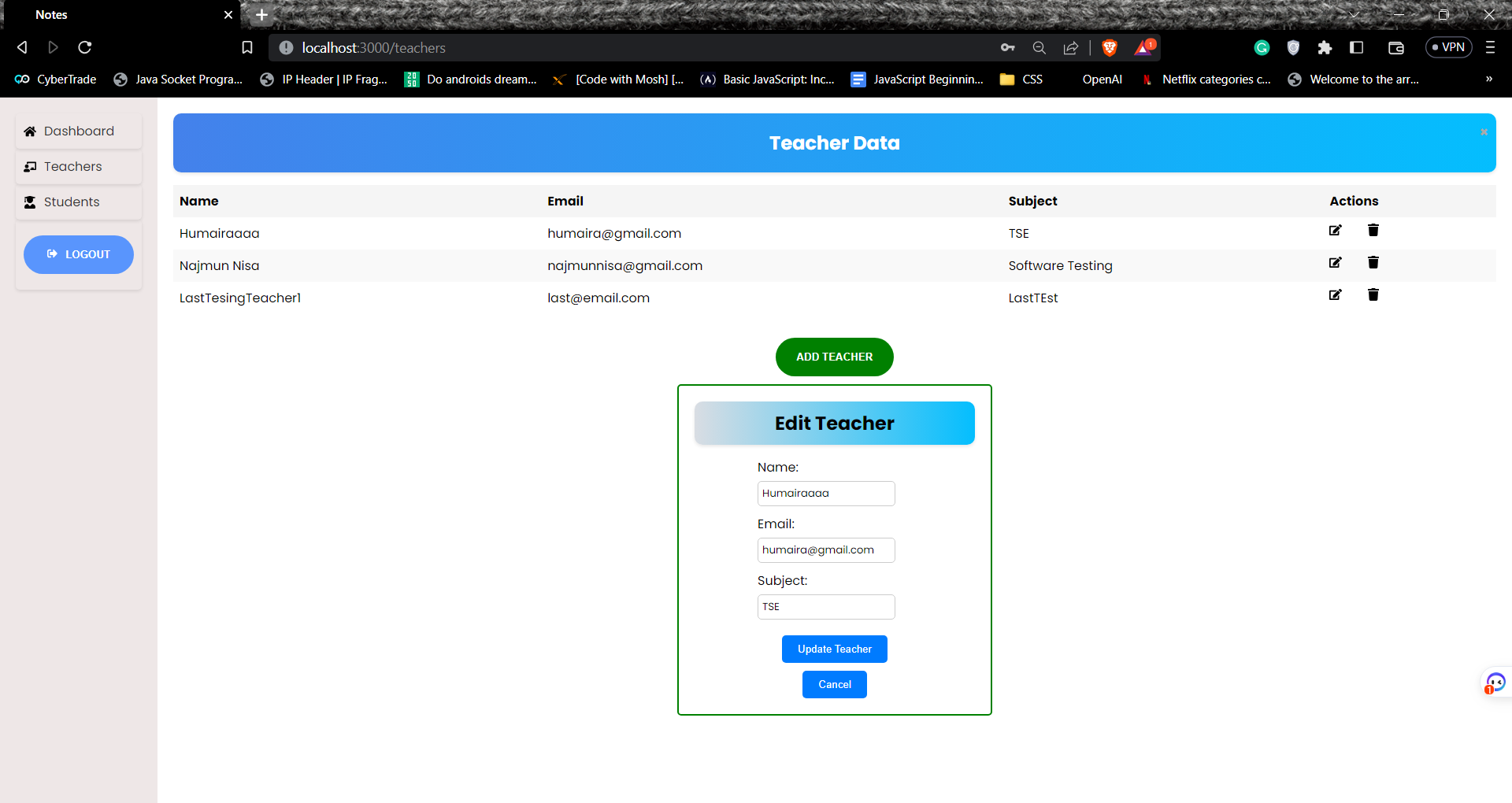
**Dashboard**

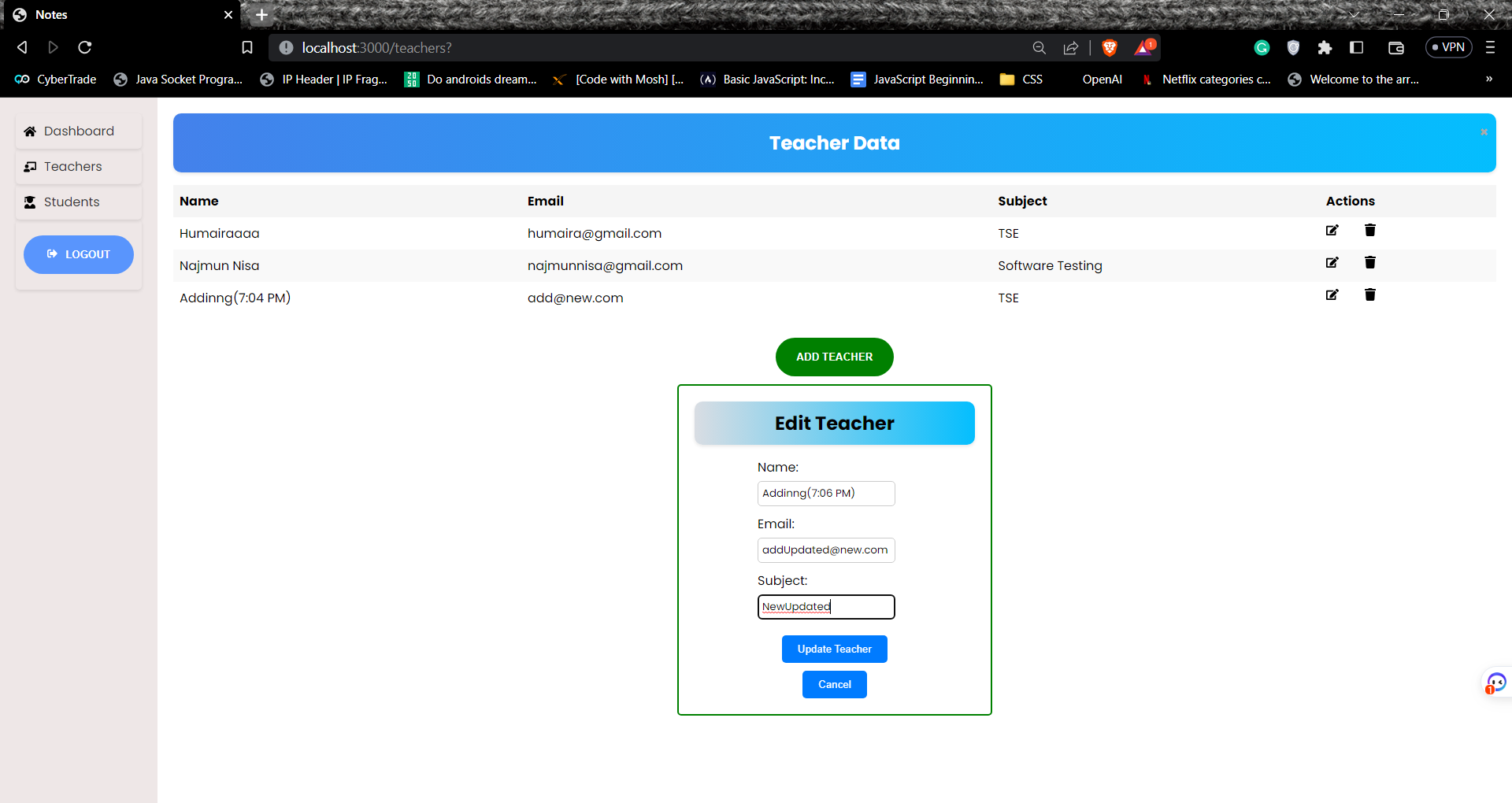
****

**Teachers**

****

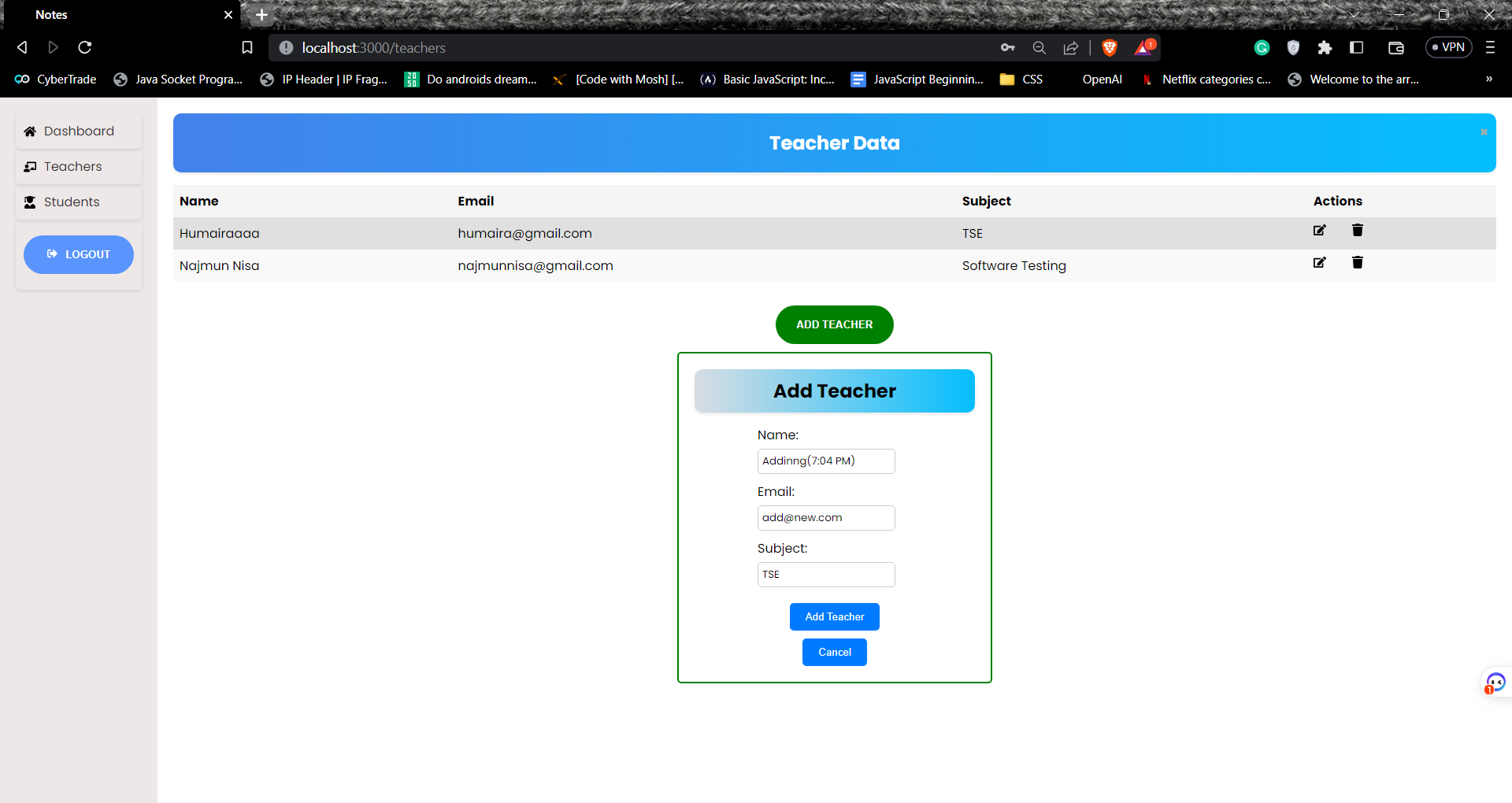
**Update Teacher**

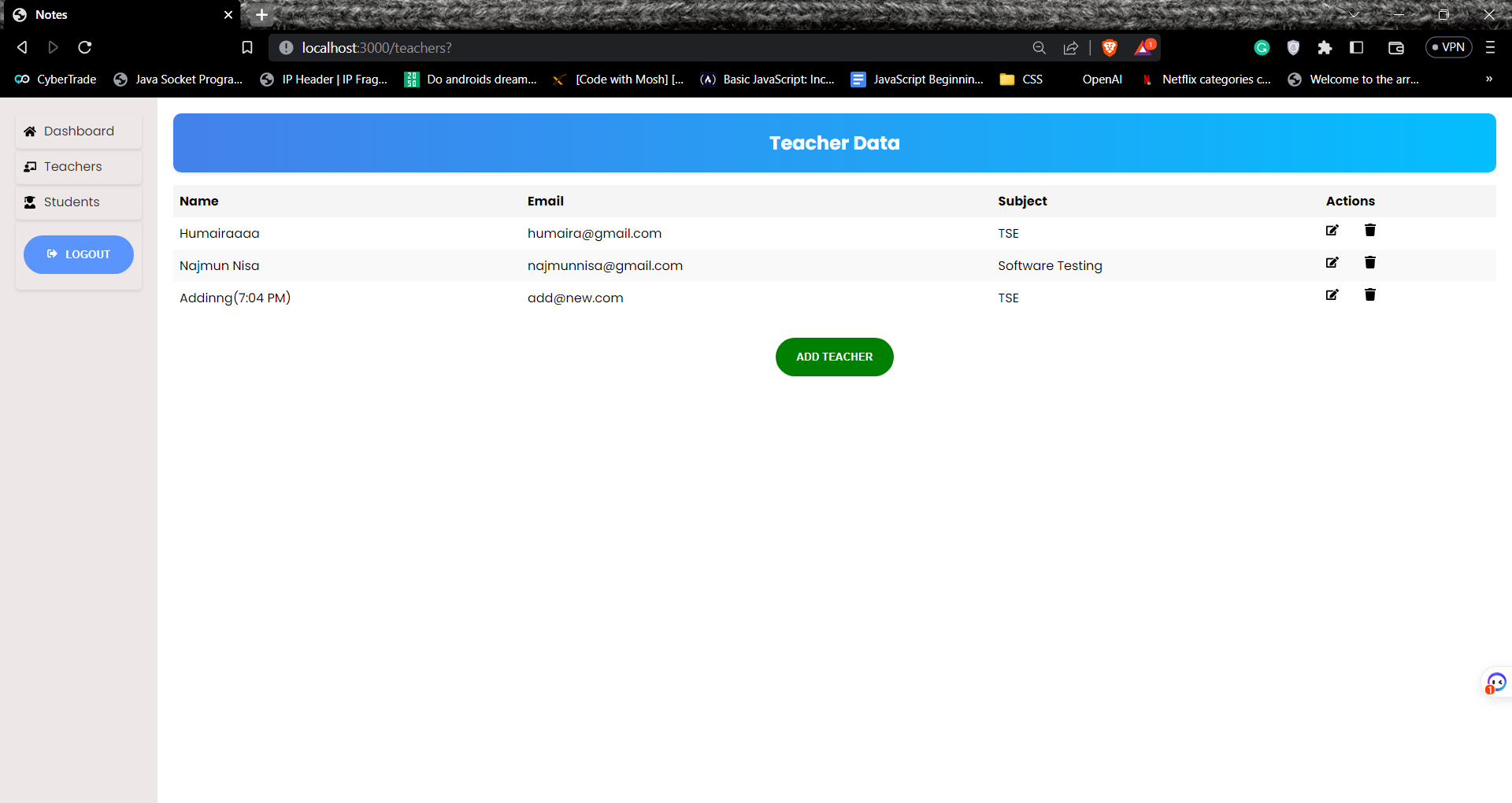
****

****

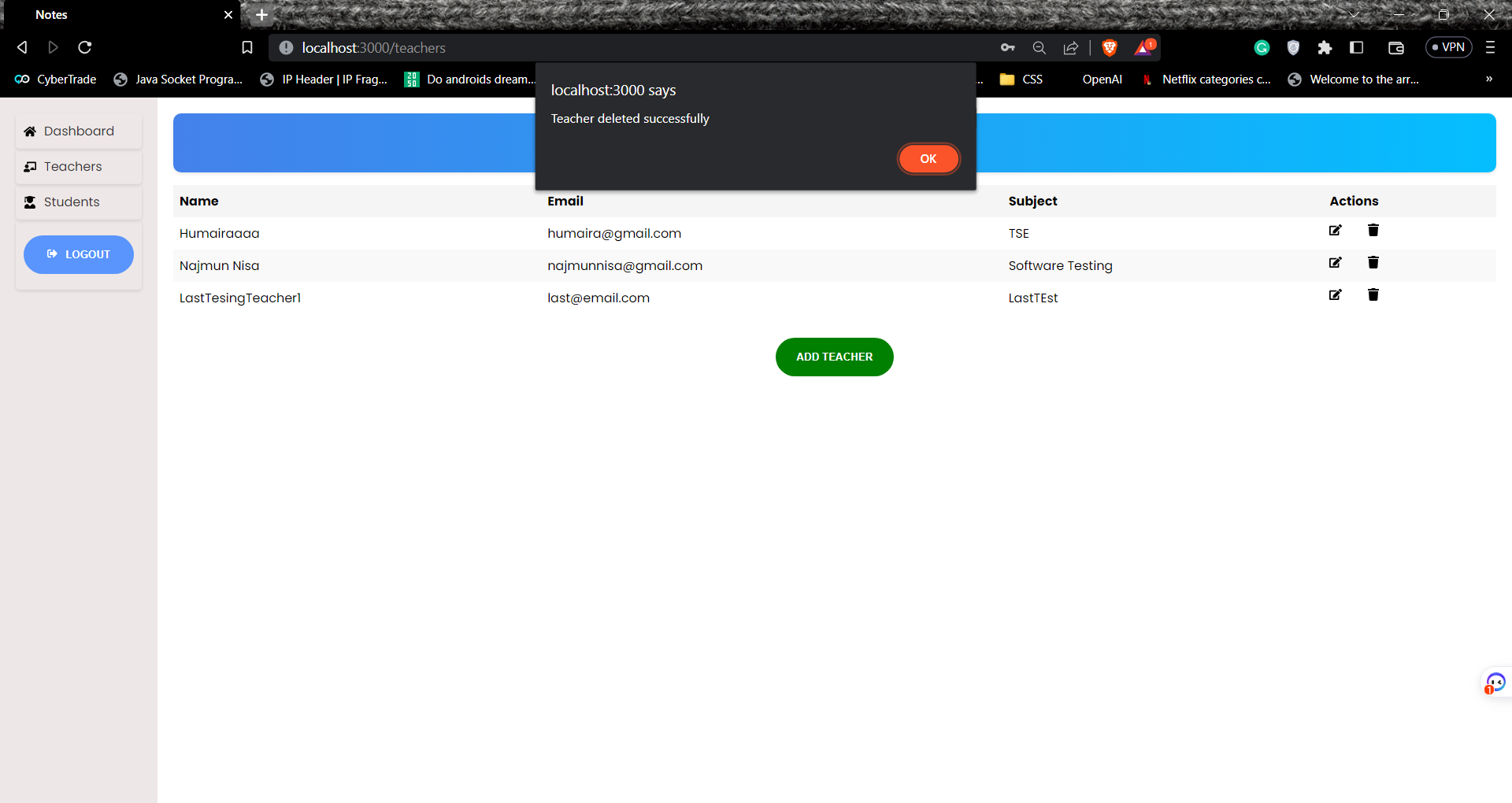
**Add Teacher**

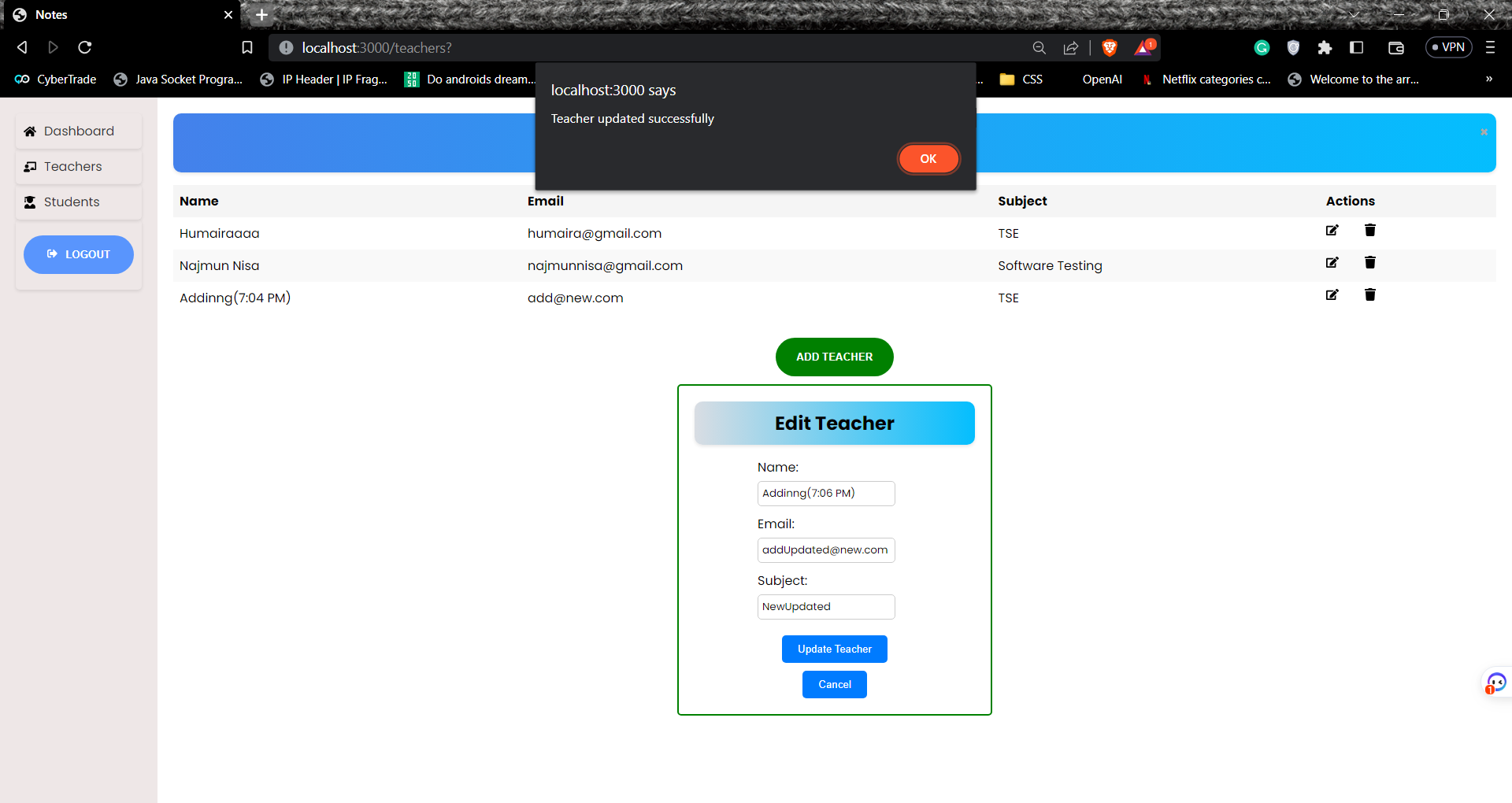
****

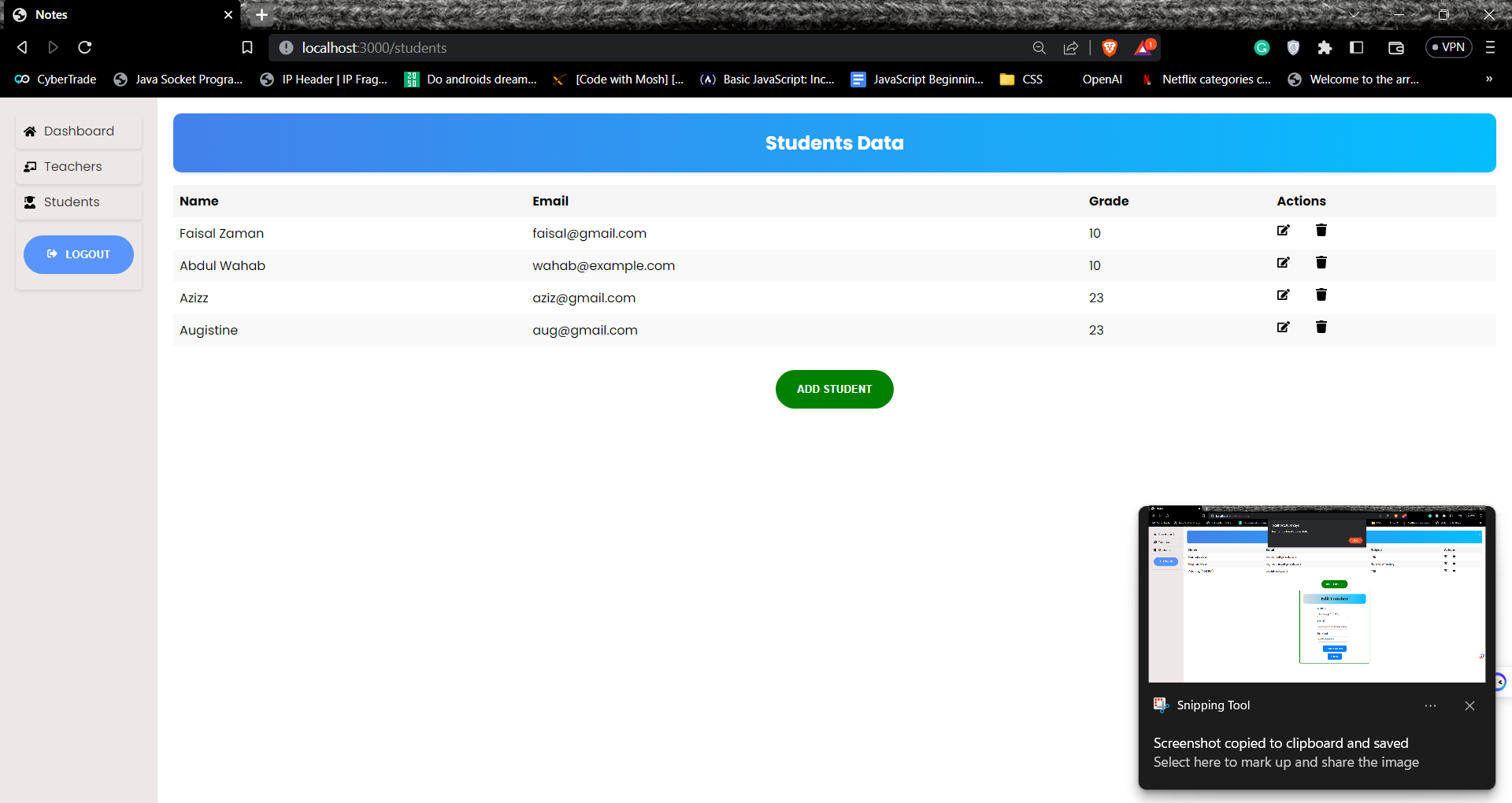
****

****

**Deleting Teacher**

****

****

**Students   
**

**Backend Code**

**App.js:**

*// app.js*

const express = require("express");

const session = require("express-session");

const FileStore = require("session-file-store")(session);

const User = require("./models/user");

const jwt = require("jsonwebtoken");

const passport = require("passport");

const LocalStrategy = require("passport-local").Strategy;

const cors = require("cors");

const helmet = require("helmet");

const mongoose = require("mongoose");

const teacherRoutes = require("./routes/teacherRoutes");

const studentRoutes = require("./routes/studentRoutes");

const shoppingCartRoutes = require("./routes/shoppingCartRoutes");

const authRoutes = require("./routes/auth");

const app = express();

app.use(express.json());

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

app.use(cors());

app.use(

  session({

    store: new FileStore(),

    secret: "keyboard-cat",

    resave: false,

    saveUninitialized: true,

    cookie: {

      maxAge: 36000,

    },

  })

);

app.use(passport.initialize());

app.use(passport.session());

passport.use(new LocalStrategy(User.authenticate()));

passport.serializeUser(User.serializeUser());

passport.deserializeUser((jwtPayload, done) => {

  if (jwtPayload && jwtPayload.userId) {

    User.findById(jwtPayload.userId, (err, user) => {

      done(err, user);

    });

  } else {

    done(null, false);

  }

});

const verifyToken = (req, res, next) => {

  const token = req.headers.authorization;

  console.log(token);

  if (!token) {

    return res.status(401).json({ message: "No token provided" });

  }

  jwt.verify(token, "your-secret-key", (err, decoded) => {

    if (err) {

      return res.status(401).json({ message: "Invalid token" });

    }

    req.userId = decoded.userId;

    next();

  });

};

const connection = mongoose.connect("mongodb://0.0.0.0:27017/tuitionAcademy", {

  useNewUrlParser: true,

  useUnifiedTopology: true,

});

connection.then(

  (db) => {

    console.log("Connected correctly to MongoDB");

  },

  (err) => {

    console.log(err);

  }

);

*// Teacher Route*

app.use("/teachers", verifyToken, teacherRoutes);

*// Teacher Endpoints (CRUD)*

*// POST=>  /teachers*

*// {*

*//   "name": "Humaira Waqas",*

*//   "email": "humairawaqas@example",*

*//   "subject": "TSE"*

*// }*

*//get=> /teachers*

*//get=> /teachers/:id*

*//put=> /teachers/:id*

*//delete=> /teachers/:id*

*// Student Route*

app.use("/students", verifyToken,studentRoutes);

*// Student Endpoints (CRUD)*

*// POST=>  /students*

*// {*

*//   "name": "Abdul Wahab",*

*//   "email": "abdulwahab@example",*

*//   "grade": 10,*

*//   "teacherId": "12345"*

*// }*

*//get=> /students*

*//get=> /students/:id*

*//put=> /students/:id*

*//delete=> /students/:id*

*// ShoppingCart Route*

app.use("/shoppingCart", shoppingCartRoutes);

*// ShoppingCart Endpoints*

*// POST=>  /shoppingCart/setItemdetails*

*// {*

*//   "itemNumber": "ABC123",*

*//   "quantity": 2*

*// }*

app.use("/auth", authRoutes);

app.listen(4500, () => {

  console.log("Server started on port 4500");

});

***Models***

**studentModel.js**

const mongoose = require("mongoose");

const studentSchema = new mongoose.Schema({

  name: {

    type: String,

    required: true,

  },

  email: {

    type: String,

    required: true,

  },

  grade: {

    type: String,

    required: true,

  },

  teachers: [

    {

      type: mongoose.Schema.Types.ObjectId,

      ref: "Teacher",

    },

  ],

});

const Student = mongoose.model("Student", studentSchema);

module.exports = Student;

**teacherModel.js**

const mongoose = require("mongoose");

const teacherSchema = new mongoose.Schema({

  name: {

    type: String,

    required: true,

  },

  email: {

    type: String,

    require: true,

    unique: true,

  },

  subject: {

    type: String,

  },

  students: [

    {

      type: mongoose.Schema.Types.ObjectId,

      ref: "Student",

    },

  ],

});

const Teacher = mongoose.model("Teacher", teacherSchema);

module.exports = Teacher;

**userModel.js**

const mongoose = require("mongoose");

const passportLocalMongoose = require("passport-local-mongoose");

const userSchema = new mongoose.Schema({

  name: {

    type: String,

    required: true,

  },

  email: {

      type: String,

      required: true,

      unique: true,

  },

});

userSchema.plugin(passportLocalMongoose);

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

module.exports = User;

***Routes***

**studentsRoutes.js**

const express = require("express");

const router = express.Router();

const Student = require("../models/student");

const Teacher = require("../models/teacher");

*// Get all students*

router.get("/", (req, res) => {

  console.log("Student Get ALl Student Rout Called")

  Student.find()

    .populate("teachers")

    .then((students) => res.json(students))

    .catch((err) => res.status(500).json({ error: err.message }));

});

*//Count all students*

router.get("/count", (req, res) => {

  Student.countDocuments()

    .then((count) => {

      res.json({ count });

    })

    .catch((err) => {

      res

        .status(500)

        .json({ error: "An error occurred while counting teachers" });

    });

});

*// Get a specific student*

router.get("/:id", (req, res) => {

  const { id } = req.params;

  Student.findById(id)

    .populate("teachers")

    .then((student) => res.json(student))

    .catch((err) => res.status(404).json({ error: "Student not found" }));

});

*// Create a new student*

router.post("/", (req, res) => {

  console.log("AddStudent In Route called")

  const { name, email, grade, teacherId } = req.body;

  const newStudent = new Student({ name, email, grade });

  Teacher.findById(teacherId)

    .then((teacher) => {

      if (!teacher) {

        throw new Error("Teacher not found");

      }

      newStudent.teachers.push(teacher);

      teacher.students.push(newStudent);

      return Promise.all([newStudent.save(), teacher.save()]);

    })

    .then(([student, teacher]) => res.status(201).json(student))

    .catch((err) => res.status(400).json({ error: err.message }));

});

*// Update a student's details*

router.put("/:id", (req, res) => {

  const { id } = req.params;

  const { name, email, grade, teacherId } = req.body;

  Promise.all([

    Student.findByIdAndUpdate(id, { name, email, grade }, { new: true }),

    Teacher.findById(teacherId),

  ])

    .then(([student, teacher]) => {

      if (!student) {

        throw new Error("Student not found");

      }

      if (!teacher) {

        throw new Error("Teacher not found");

      }

      student.teachers = [teacher];

      teacher.students.push(student);

      return Promise.all([student.save(), teacher.save()]);

    })

    .then(([student, teacher]) => res.json(student))

    .catch((err) => res.status(400).json({ error: err.message }));

});

*// Update a student's details partially*

router.patch("/:id", (req, res) => {

  const { id } = req.params;

  const { name, email, grade, teacherId } = req.body;

  Promise.all([

    Student.findByIdAndUpdate(

      id,

      { $set: { name, email, grade } },

      { new: true }

    ),

    Teacher.findById(teacherId),

  ])

    .then(([student, teacher]) => {

      if (!student) {

        throw new Error("Student not found");

      }

      if (!teacher) {

        throw new Error("Teacher not found");

      }

      student.teachers = [teacher];

      teacher.students.push(student);

      return Promise.all([student.save(), teacher.save()]);

    })

    .then(([student, teacher]) => res.json(student))

    .catch((err) => res.status(400).json({ error: err.message }));

});

*// Delete a student*

router.delete("/:id", (req, res) => {

  const { id } = req.params;

  Student.findByIdAndDelete(id)

    .then(() => res.sendStatus(204))

    .catch((err) => res.status(400).json({ error: err.message }));

});

module.exports = router;

**teacherRoutes.js**

const express = require("express");

const router = express.Router();

const Teacher = require("../models/teacher");

const Student = require("../models/student");

*// Get all teachers*

router.get("/", (req, res) => {

  Teacher.find()

    .populate("students")

    .then((teachers) => res.json(teachers))

    .catch((err) => res.status(500).json({ error: err.message }));

});

router.get("/count", (req, res) => {

  Teacher.countDocuments()

    .then((count) => {

      res.json({ count });

    })

    .catch((err) => {

      res

        .status(500)

        .json({ error: "An error occurred while counting teachers" });

    });

});

*// Get a specific teacher*

router.get("/:id", (req, res) => {

  console.log("GET TEACHER ID")

  const { id } = req.params;

  Teacher.findById(id)

    .populate("students")

    .then((teacher) => res.json(teacher))

    .catch((err) => res.status(404).json({ error: "Teacher not found" }));

});

*// Create a new teacher*

router.post("/", (req, res) => {

  console.log("POST TEACHER")

  console.log(req.body)

  const { name, email, subject } = req.body;

  const newTeacher = new Teacher({ name, email, subject });

  newTeacher

    .save()

    .then((teacher) => res.status(201).json(teacher))

    .catch((err) => res.status(400).json({ error: err.message }));

});

*// Update a teacher's details*

router.put("/:id", (req, res) => {

  const { id } = req.params;

  const { name, email, subject } = req.body;

  Teacher.findByIdAndUpdate(id, { name, email, subject }, { new: true })

    .then((teacher) => res.json(teacher))

    .catch((err) => res.status(400).json({ error: err.message }));

});

*// Update a teacher's details partially*

router.patch("/:id", (req, res) => {

  const { id } = req.params;

  const { name, email, subject } = req.body;

  Teacher.findByIdAndUpdate(

    id,

    { $set: { name, email, subject } },

    { new: true }

  )

    .then((teacher) => res.json(teacher))

    .catch((err) => res.status(400).json({ error: err.message }));

});

*// Delete a teacher*

router.delete("/:id", (req, res) => {

  const { id } = req.params;

  Teacher.findByIdAndDelete(id)

    .then((teacher) => res.json(teacher))

    .catch((err) => res.status(400).json({ error: err.message }));

});

module.exports = router;

**auth.js**

const jwt = require("jsonwebtoken");

const express = require("express");

const router = express.Router();

const passport = require("passport");

const User = require("../models/user");

*// POST /auth/signup*

router.post("/signup", (req, res) => {

  const { name, email, password } = req.body;

  console.log(req.body);

  User.register(new User({ username: email, name, email }), password, (err) => {

    if (err) {

      console.error(err);

      return res

        .status(500)

        .json({ message: "Error occurred while signing up" });

    }

    res.json({ message: "Signup successful" });

  });

});

*// POST /auth/login*

router.post("/login", (req, res, next) => {

  console.log("LOGIN")

  if (req.isAuthenticated()) {

    return res.json({ message: "You are already logged in" });

  }

  passport.authenticate("local", (err, user, info) => {

    if (err) {

      console.error(err);

      return res

        .status(500)

        .json({ message: "Error occurred while logging in" });

    }

    if (!user) {

      return res.status(401).json({ message: "Invalid email or password" });

    }

    req.logIn(user, (err) => {

      if (err) {

        console.error(err);

        return res

          .status(500)

          .json({ message: "Error occurred while logging in" });

      }

*// Generate JWT token*

      const token = jwt.sign({ userId: user.\_id }, "your-secret-key", {

        expiresIn: "1h",

      });

      res.json({

        message: "Login successful", token,

        user: {

          id: user.\_id,

          name: user.name,

          email: user.email,

        },});

    });

  })(req, res, next);

});

*// GET /auth/logout*

router.get("/logout", (req, res) => {

  req.logout((err) => {

    if (err) {

      router.post("/login", (req, res, next) => {

        if (req.isAuthenticated()) {

          return res.json({ message: "You are already logged in" });

        }

        passport.authenticate("local", (err, user, info) => {

          if (err) {

            console.error(err);

            return res

              .status(500)

              .json({ message: "Error occurred while logging in" });

          }

          if (!user) {

            return res

              .status(401)

              .json({ message: "Invalid email or password" });

          }

          req.logIn(user, (err) => {

            if (err) {

              console.error(err);

              return res

                .status(500)

                .json({ message: "Error occurred while logging in" });

            }

            res.json({ message: "Login successful" });

          });

        })(req, res, next);

      });

      console.error(err);

      return res

        .status(500)

        .json({ message: "Error occurred while logging out" });

    }

    res.json({ message: "Logout successful" });

  });

});

module.exports = router;

**Front End**

**App.jsx**

import React from "react";

import {

  BrowserRouter as Router,

  Routes,

  Route,

  Navigate,

} from "react-router-dom";

import LoginSignUpForm from "./pages/LoginSignupForm.js";

import Dashboard from "./pages/dashboard.js";

import Teachers from "./pages/Teachers.js";

import Layout from "./layout/layout.js";

import Students from "./pages/Students.js"

import "./index.css";

const App = () => {

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

  const PrivateRoute = ({ element: Element, ...rest }) => {

    return isLoggedIn ? (

      <Layout>

        <Element {...rest} />

      </Layout>

    ) : (

      <Navigate to="/" replace={true} />

    );

  };

  const PublicRoute = ({ element: Element, ...rest }) => {

    return isLoggedIn ? (

      <Navigate to="/dashboard" replace={true} />

    ) : (

      <Element {...rest} />

    );

  };

  return (

    <Router>

      <div className="page">

        <Routes>

          <Route path="/" element={<PublicRoute element={LoginSignUpForm} />} />

          <Route

            path="/dashboard"

            element={<PrivateRoute element={Dashboard} />}

          />

          <Route

            path="/teachers"

            element={<PrivateRoute element={Teachers} />}

          />

          <Route

            path="/students"

            element={<PrivateRoute element={Students} />}

          />

        </Routes>

      </div>

    </Router>

  );

};

export default App;

**Pages**

**LoginSigUpForm.js**

import React, { useState } from "react";

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

import axios from "axios";

import LoginImage from "../assets/images/log.svg";

import RegisterImage from "../assets/images/register.svg";

import { FaUser, FaLock, FaKey, FaEnvelope } from "react-icons/fa";

const LoginForm = () => {

  const [isSignUpMode, setIsSignUpMode] = useState(false);

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

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

  const [password, setPassword] = useState("");

  const [confirmPassword, setConfirmPassword] = useState("");

  const navigate = useNavigate();

  const handleSignUpClick = () => {

    setIsSignUpMode(true);

  };

  const handleSignInClick = () => {

    setIsSignUpMode(false);

  };

  const handleInputChange = (e) => {

    const { name, value } = e.target;

    if (name === "name") {

      setName(value);

    } else if (name === "email") {

      setEmail(value);

    } else if (name === "password") {

      setPassword(value);

    }

  };

  const handleSubmit = async (e) => {

      e.preventDefault();

    if (isSignUpMode) {

*// Handle sign up*

*//console log to check if the data is being sent to the backend*

        if (password !== confirmPassword) {

            alert("Passwords do not match");

            return;

        }

      console.log(name, email, password);

      try {

*// Make an API call to sign up the user*

        const response = await axios.post("http://localhost:4500/auth/signup", {

          name,

          email,

          password,

        });

*// Handle successful login*

        console.log(response.data);

*// Redirect the user to the desired page after successful sign up*

        navigate("/dashboard");

      } catch (error) {

*// Handle sign up error*

        console.log(error);

      }

    } else {

*// Handle sign in*

      try {

        const response = await axios.post("http://localhost:4500/auth/login", {

          username: name,

          password,

        });

*// Handle successful login*

        console.log(response.data);

        onSuccess(response.data);

*// Redirect the user to the desired page after successful sign in*

      } catch (error) {

*// Handle sign in error*

        if (error.response && error.response.status === 401) {

*// Unauthorized: Incorrect email or password*

          alert("Incorrect email or password");

        } else if (error.response && error.response.status === 400) {

*// Bad Request: Missing email or password*

          alert("Please enter email and password");

*// Other error*

          console.log(error);

        }

        console.log(error);

      }

    }

  };

  const onSuccess = (data) => {

*// Handle successful login*

    console.log("User",data);

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

    localStorage.setItem("user", JSON.stringify(data.user));

    localStorage.setItem("isLoggedIn", true);

    const user = localStorage.getItem("user");

    console.log("user", user);

 alert("User signed in");

*// Redirect to Home component*

    window.location.href = "/dashboard";

  };

  return (

    <div className={`container ${isSignUpMode ? "sign-up-mode" : ""}`}>

      <div className="forms-container">

        <div className="signin-signup">

          <form className="sign-in-form" onSubmit={handleSubmit}>

            <h2 className="title">Sign in</h2>

            <div className="input-field">

              <FaUser />

              <input

                type="text"

                placeholder="Enter your Email"

                name="name"

                value={name}

                onChange={handleInputChange}

                required

              />

            </div>

            <div className="input-field">

              <FaLock />

              <input

                type="password"

                placeholder="Password"

                name="password"

                value={password}

                required

                onChange={handleInputChange}

              />

            </div>

            <input type="submit" value="Login" className="btn solid" />

          </form>

          <form className="sign-up-form" onSubmit={handleSubmit}>

            <h2 className="title">Sign up</h2>

            <div className="input-field">

              <FaUser />

              <input

                type="text"

                placeholder="Username"

                name="name"

                required

                value={name}

                onChange={handleInputChange}

              />

            </div>

            <div className="input-field">

              <FaEnvelope />

              <input

                type="email"

                placeholder="Email"

                name="email"

                required

                value={email}

                onChange={handleInputChange}

              />

            </div>

            <div className="input-field">

              <FaKey />

              <input

                type="password"

                required

                placeholder="Password"

                name="password"

                value={password}

                onChange={handleInputChange}

              />

            </div>

            <div className="input-field">

              <FaKey />

              <input

                type="password"

                required

                placeholder="Confirm Password"

                name="password"

                value={confirmPassword}

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

              />

            </div>

            <input type="submit" className="btn" value="Sign up" />

          </form>

        </div>

      </div>

      <div className="panels-container">

        <div className="panel left-panel">

          <div className="content">

            <h3>New here?</h3>

            <p>

              Lorem ipsum, dolor sit amet consectetur adipisicing elit. Debitis,

              ex ratione. Aliquid!

            </p>

            <button className="btn transparent" onClick={handleSignUpClick}>

              Sign up

            </button>

          </div>

          <img src={LoginImage} className="image" alt="" />

        </div>

        <div className="panel right-panel">

          <div className="content">

            <h3>One of us?</h3>

            <p>

              Lorem ipsum dolor sit amet consectetur adipisicing elit. Nostrum

              laboriosam ad deleniti.

            </p>

            <button className="btn transparent" onClick={handleSignInClick}>

              Sign in

            </button>

          </div>

          <img src={RegisterImage} className="image" alt="" />

        </div>

      </div>

    </div>

  );

};

export default LoginForm;

**Dashboard.jsx**import React, { useState, useEffect } from "react";

import axios from "axios";

const Dashboard = () => {

  const [teacherCount, setTeacherCount] = useState(0);

  const [studentCount, setStudentCount] = useState(0);

  const [userName, setUserName] = useState("");

  const user = localStorage.getItem("user");

  const userObj = JSON.parse(user);

  const { name } = userObj;

  console.log("user", userObj);

  console.log("name", name);

  useEffect(() => {

    fetchTeachers();

  }, []);

  const fetchTeachers = async () => {

    try {

      const token = localStorage.getItem("token");

      const response = await axios.get("http://localhost:4500/teachers/count", {

        headers: {

          Authorization: `${token}`,

        },

      });

      console.log("token", token);

      setTeacherCount(response.data.count);

    } catch (error) {

      console.log(error);

    }

  };

  useEffect(() => {

    fetchStudents();

  }, []);

  const fetchStudents = async () => {

    try {

      const token = localStorage.getItem("token");

      const response = await axios.get("http://localhost:4500/students/count", {

        headers: {

          Authorization: `${token}`,

        },

      });

      console.log("token", token);

      setStudentCount(response.data.count);

    } catch (error) {

      console.log(error);

    }

  };

  return (

    <>

      <h2>Welcome to the Dashboard,

        <span

          style={{

            color: "blue",

            textTransform: "uppercase",

            textDecoration: "underline",

            border: "1px solid black",

            padding: "5px",

            borderRadius: "10px",

            marginLeft: "10px",

          }}

        >

          {name}!

      </span>

        </h2>

      <div className="card-container">

        <div className="card">

          <h3>Total Teachers</h3>

          <p>{teacherCount}</p>

        </div>

        <div className="card">

          <h3>Total Students</h3>

          <p>{studentCount}</p>

        </div>

      </div>

    </>

  );

};

export default Dashboard;

**Teachers.jsx**

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

import axios from "axios";

import { FaEdit, FaTrash } from "react-icons/fa";

const Teachers = () => {

  const [teachers, setTeachers] = useState([]);

  const [showModal, setShowModal] = useState(false);

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

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

  const [subject, setSubject] = useState("");

  const [editTeacher, setEditTeacher] = useState({});

  const [editModal, setEditModal] = useState(false);

  const [editName, setEditName] = useState("");

  const [editEmail, setEditEmail] = useState("");

  const [editSubject, setEditSubject] = useState("");

  useEffect(() => {

    fetchTeachers();

  }, []);

  const fetchTeachers = async () => {

    try {

      const token = localStorage.getItem("token");

      const response = await axios.get("http://localhost:4500/teachers", {

        headers: {

          Authorization: `${token}`,

        },

      });

      setTeachers(response.data);

    } catch (error) {

      console.log(error);

    }

  };

  const deleteTeacher = async (id) => {

    try {

      const token = localStorage.getItem("token");

      const response = await axios.delete(

        `http://localhost:4500/teachers/${id}`,

        {

          headers: {

            Authorization: `${token}`,

          },

        }

      );

      if (response.status === 200) {

        alert("Teacher deleted successfully");

      }

      fetchTeachers();

    } catch (error) {

      console.log(error);

    }

  };

  const openModal = () => {

    setShowModal(true);

  };

  const closeModal = () => {

    setShowModal(false);

  };

  const addTeacher = async () => {

    try {

      const token = localStorage.getItem("token");

      const response = await axios.post(

        "http://localhost:4500/teachers",

        {

          name,

          email,

          subject,

        },

        {

          headers: {

            Authorization: `${token}`,

          },

        }

      );

      if (response.status === 201) {

        alert("Teacher added successfully");

        closeModal();

        fetchTeachers();

      }

    } catch (error) {

      console.log(error);

    }

  };

  const openEditModal = (teacher) => {

    setEditTeacher(teacher);

    setEditName(teacher.name);

    setEditEmail(teacher.email);

    setEditSubject(teacher.subject);

    setEditModal(true);

  };

  const closeEditModal = () => {

    setEditModal(false);

  };

  const updateTeacher = async (e) => {

    e.preventDefault();

    try {

      const token = localStorage.getItem("token");

      const response = await axios.put(

        `http://localhost:4500/teachers/${editTeacher.\_id}`,

        {

          name: editName,

          email: editEmail,

          subject: editSubject,

        },

        {

          headers: {

            Authorization: `${token}`,

          },

        }

      );

      if (response.status === 200) {

        alert("Teacher updated successfully");

        closeEditModal();

        fetchTeachers();

      }

    } catch (error) {

      console.log(error);

    }

  };

  return (

    <div className="teacherPage">

      <h2 className="page-header">Teacher Data</h2>

      <table className="teacher-table">

        <thead>

          <tr>

            <th>Name</th>

            <th>Email</th>

            <th>Subject</th>

            <th>Actions</th>

          </tr>

        </thead>

        <tbody>

          {teachers.map((teacher) => (

            <tr key={teacher.\_id}>

              <td>{teacher.name}</td>

              <td>{teacher.email}</td>

              <td>{teacher.subject}</td>

              <td className="action-column">

                <FaEdit

                  className="edit-icon"

                  onClick={() => openEditModal(teacher)}

                />

                <FaTrash

                  onClick={() => deleteTeacher(teacher.\_id)}

                  className="delete-icon"

                />

              </td>

            </tr>

          ))}

        </tbody>

      </table>

      <div

        className="btnContainer"

        style={{

          display: "flex",

          justifyContent: "center",

          marginTop: "20px",

        }}

      >

        <button

          style={{

            backgroundColor: "green",

          }}

          className="btn"

          onClick={openModal}

        >

          Add Teacher

        </button>

      </div>

      {showModal && (

        <div className="modal">

          <div className="modal-content">

            <span className="close" onClick={closeModal}>

              &times;

            </span>

            <h2>Add Teacher</h2>

            <form onSubmit={addTeacher}>

              <label>

                Name:

                <input

                  type="text"

                  value={name}

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

                  required

                />

              </label>

              <label>

                Email:

                <input

                  type="email"

                  value={email}

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

                  required

                />

              </label>

              <label>

                Subject:

                <input

                  type="text"

                  value={subject}

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

                  required

                />

              </label>

              <button type="submit">Add Teacher</button>

              <button type="button" onClick={closeModal}>

                Cancel

              </button>

            </form>

          </div>

        </div>

      )}

      {editModal && (

        <div className="modal">

          <div className="modal-content">

            <span className="close" onClick={closeEditModal}>

              &times;

            </span>

            <h2>Edit Teacher</h2>

            <form onSubmit={updateTeacher}>

              <label>

                Name:

                <input

                  type="text"

                  value={editName}

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

                  required

                />

              </label>

              <label>

                Email:

                <input

                  type="email"

                  value={editEmail}

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

                  required

                />

              </label>

              <label>

                Subject:

                <input

                  type="text"

                  value={editSubject}

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

                  required

                />

              </label>

              <button type="submit">Update Teacher</button>

              <button type="button" onClick={closeEditModal}>

                Cancel

              </button>

            </form>

          </div>

        </div>

      )}

    </div>

  );

};

export default Teachers;

**Students.jsx**

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

import axios from "axios";

import { FaEdit, FaTrash } from "react-icons/fa";

const Students = () => {

  const [students, setStudents] = useState([]);

  const [showModal, setShowModal] = useState(false);

  const [updateModal, setUpdateModal] = useState(false);

  const [selectedStudent, setSelectedStudent] = useState(null);

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

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

  const [grade, setGrade] = useState("");

  const [teachers, setTeachers] = useState([]);

  const [teacherId, setTeacherId] = useState("");

  useEffect(() => {

    fetchStudents();

    fetchTeachers();

  }, []);

  const fetchStudents = async () => {

    try {

      const token = localStorage.getItem("token");

      const response = await axios.get("http://localhost:4500/students", {

        headers: {

          Authorization: `${token}`,

        },

      });

      setStudents(response.data);

    } catch (error) {

      console.log(error);

    }

  };

  const fetchTeachers = async () => {

    try {

      const token = localStorage.getItem("token");

      const response = await axios.get("http://localhost:4500/teachers", {

        headers: {

          Authorization: `${token}`,

        },

      });

      setTeachers(response.data);

    } catch (error) {

      console.log(error);

    }

  };

  const deleteStudent = async (id) => {

    try {

      const token = localStorage.getItem("token");

      const response = await axios.delete(

        `http://localhost:4500/students/${id}`,

        {

          headers: {

            Authorization: `${token}`,

          },

        }

      );

      if (response.status === 200) {

        alert("Student deleted successfully");

      }

      fetchStudents();

    } catch (error) {

      console.log(error);

    }

  };

  const openAddModal = () => {

    setShowModal(true);

  };

  const closeAddModal = () => {

    setShowModal(false);

  };

  const openUpdateModal = (student) => {

    setSelectedStudent(student);

    setName(student.name);

    setEmail(student.email);

    setGrade(student.grade);

    setTeacherId(student.teacherId);

    setUpdateModal(true);

  };

  const closeUpdateModal = () => {

    setSelectedStudent(null);

    setName("");

    setEmail("");

    setGrade("");

    setTeacherId("");

    setUpdateModal(false);

  };

  const addStudent = async (e) => {

    e.preventDefault();

    const foundTeacher = teachers.find((teacher) => teacher.\_id === teacherId);

    try {

      const token = localStorage.getItem("token");

      const response = await axios.post(

        "http://localhost:4500/students",

        {

          name,

          email,

          grade,

          teacherId: foundTeacher.\_id,

        },

        {

          headers: {

            Authorization: `${token}`,

          },

        }

      );

      if (response.status === 201) {

        alert("Student added successfully");

        closeAddModal();

        fetchStudents();

      }

    } catch (error) {

      console.log(error);

    }

  };

  const updateStudent = async () => {

    const foundTeacher = teachers.find((teacher) => teacher.\_id === teacherId);

    try {

      const token = localStorage.getItem("token");

      const response = await axios.put(

        `http://localhost:4500/students/${selectedStudent.\_id}`,

        {

          name,

          email,

          grade,

          teacherId: foundTeacher.\_id,

        },

        {

          headers: {

            Authorization: `${token}`,

          },

        }

      );

      if (response.status === 200) {

        alert("Student updated successfully");

        closeUpdateModal();

        fetchStudents();

      }

    } catch (error) {

      console.log(error);

    }

  };

  return (

    <div className="teacherPage">

      <h2 className="page-header">Students Data</h2>

      <table className="teacher-table">

        <thead>

          <tr>

            <th>Name</th>

            <th>Email</th>

            <th>Grade</th>

            <th>Actions</th>

          </tr>

        </thead>

        <tbody>

          {students.map((student) => (

            <tr key={student.\_id}>

              <td>{student.name}</td>

              <td>{student.email}</td>

              <td>{student.grade}</td>

              <td className="action-column">

                <FaEdit

                  className="edit-icon"

                  onClick={() => openUpdateModal(student)}

                />

                <FaTrash

                  onClick={() => deleteStudent(student.\_id)}

                  className="delete-icon"

                />

              </td>

            </tr>

          ))}

        </tbody>

      </table>

      <div

        className="btnContainer"

        style={{

          display: "flex",

          justifyContent: "center",

          marginTop: "20px",

        }}

      >

        <button

          style={{

            backgroundColor: "green",

          }}

          className="btn"

          onClick={openAddModal}

        >

          Add Student

        </button>

      </div>

      {showModal && (

        <div className="modal">

          <div className="modal-content">

            <span className="close" onClick={closeAddModal}>

              &times;

            </span>

            <h2>Add Student</h2>

            <form onSubmit={addStudent}>

              <label>

                Name:

                <input

                  type="text"

                  value={name}

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

                  required

                />

              </label>

              <label>

                Email:

                <input

                  type="email"

                  value={email}

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

                  required

                />

              </label>

              <label>

                Grade:

                <input

                  type="number"

                  min={0}

                  value={grade}

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

                  required

                />

              </label>

              <label>

                Teacher:

                <select

                  value={teacherId}

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

                  className="teacher-select"

                >

                  <option value="">Select Teacher</option>

                  {teachers.map((teacher) => (

                    <option key={teacher.\_id} value={teacher.\_id}>

                      {teacher.name}

                    </option>

                  ))}

                </select>

              </label>

              <button type="submit">Add Student</button>

              <button type="button" onClick={closeAddModal}>

                Cancel

              </button>

            </form>

          </div>

        </div>

      )}

      {updateModal && selectedStudent && (

        <div className="modal">

          <div className="modal-content">

            <span className="close" onClick={closeUpdateModal}>

              &times;

            </span>

            <h2>Edit Student</h2>

            <form onSubmit={updateStudent}>

              <label>

                Name:

                <input

                  type="text"

                  value={name}

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

                  required

                />

              </label>

              <label>

                Email:

                <input

                  type="email"

                  value={email}

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

                  required

                />

              </label>

              <label>

                Grade:

                <input

                  type="number"

                  min={0}

                  value={grade}

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

                  required

                />

              </label>

              <label>

                Teacher:

                <select

                  value={teacherId}

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

                  className="teacher-select"

                >

                  <option value="">Select Teacher</option>

                  {teachers.map((teacher) => (

                    <option key={teacher.\_id} value={teacher.\_id}>

                      {teacher.name}

                    </option>

                  ))}

                </select>

              </label>

              <button type="submit">Update Student</button>

              <button type="button" onClick={closeUpdateModal}>

                Cancel

              </button>

            </form>

          </div>

        </div>

      )}

    </div>

  );

};

export default Students;

**Component**

**SideBar.jsx**

import React from "react";

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

import {

  FaUserGraduate,

  FaChalkboardTeacher,

  FaSignOutAlt,

  FaHome,

} from "react-icons/fa";

const Sidebar = () => {

  const handleLogout = () => {

*// Clear local storage and redirect to the login/signup form*

    localStorage.removeItem("token");

    localStorage.removeItem("user");

    localStorage.removeItem("isLoggedIn");

    window.location.href = "/";

  };

  return (

    <div className="sidebar">

      <div className="sidebar-item">

        <FaHome className="sidebar-icon" />

        <Link to="/dashboard">Dashboard</Link>

      </div>

      <div className="sidebar-item">

        <FaChalkboardTeacher className="sidebar-icon" />

        <Link to="/teachers">Teachers</Link>

      </div>

      <div className="sidebar-item">

        <FaUserGraduate className="sidebar-icon" />

        <Link to="/students">Students</Link>

      </div>

      <div className="sidebar-item logout">

        <button className="btn" onClick={handleLogout}>

          <FaSignOutAlt className="logout-icon" />

          Logout

        </button>

      </div>

    </div>

  );

};

export default Sidebar;

**Layout**

import React from "react";

import Sidebar from "../components/SideBar/SideBar";

const Layout = ({ children }) => {

  return (

    <div className="layout">

      <Sidebar />

      <div className="content">{children}</div>

    </div>

  );

};

export default Layout;