**Index.js:**const express = require("express");

  const mongoose = require("mongoose");

  const cors = require("cors");

  const jwt = require("jsonwebtoken");

  const bcrypt = require("bcryptjs");

  const axios = require("axios");

  const http = require("http");

  const { Server } = require("socket.io");

  const { Tupath\_usersModel, Employer\_usersModel, Project, AssessmentQuestion, Admin } = require("./models/Tupath\_users");

  const nodemailer = require("nodemailer");

  const crypto = require("crypto");

  const cookieParser = require('cookie-parser'); //start to delete this line if trial and error

  //pushin purposes

 require('dotenv').config()

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

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

  const questions = require("./routes/questions")

  const userStats = require("./routes/userStats")

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

  const studentByTags = require("./routes/studentsByTag")

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

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

  const JWT\_SECRET = "your-secret-key";

  const GOOGLE\_CLIENT\_ID = "625352349873-hrob3g09um6f92jscfb672fb87cn4kvv.apps.googleusercontent.com";

  const app = express();

  const server = http.createServer(app);

  const multer = require("multer");

  const path = require("path");

  // Middleware setup

  app.use(cors({ origin: 'http://localhost:5173', credentials: true, })); // Updated CORS for specific origin // SET CREDENTIALS AS TRUE

  app.use('/uploads', express.static('uploads'));

  app.use("/certificates", express.static(path.join(\_\_dirname, "certificates")));

  app.use(express.json({ limit: '50mb' })); // Increase the limit to 50 MB

  app.use(express.urlencoded({ limit: '50mb', extended: true }));

    //ROUTES

    app.use('/', users);

    app.use('/', adminsignup);

    app.use('/', adminLogin);

    app.use('/', questions);

    app.use('/', userStats);

    app.use('/', studentTags);

    app.use('/', studentByTags);

    app.use('/', adminDelete);

  // Middleware for setting COOP headers

  app.use((req, res, next) => {

    res.setHeader('Cross-Origin-Opener-Policy', 'same-origin-allow-popups'); // Added COOP header

    res.setHeader('Cross-Origin-Resource-Policy', 'cross-origin'); // Added CORP header

    next();

  });

x

  app.use(cookieParser());

**AdminLogin.jsx:**

import React from 'react';

import './AdminLogin.css';

import { motion } from 'framer-motion';

import adminheadericon from '../../../assets/logoicon2.png';

import adminloginicon from '../../../assets/user-gear.png';

import { useState } from 'react';

import axiosInstancev2 from '../../../services/axiosInstancev2';

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

const AdminLogin = () => {

  const [formData, setFormData] = useState({ username: '', password: '' });

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

  const handleChange = (e) => {

    const { name, value } = e.target;

    setFormData({ ...formData, [name]: value });

  };

  const navigate = useNavigate();

  const handleSignupRedirect = () => {

    navigate('/adminsignup');

  }

  const handleLogin = (e) => {

    e.preventDefault();

    setError('');

    axiosInstancev2

      .post('/api/admin/login', formData, { withCredentials: true }) // Ensure credentials are sent

      .then((response) => {

        if (response.data.success) {

          // Redirect to admin dashboard

          window.location.href = '/admindashboard';

        } else {

          setError(response.data.message || 'Invalid login credentials.');

        }

      })

      .catch((error) => {

        console.error('Error during login:', error);

        setError(error.response?.data?.message || 'An error occurred. Please try again.');

      });

  };

  return (

    <div className="admin-login-container">

      <header className="admin-header">

        <img src={adminheadericon} alt="Admin Header Icon" />

      </header>

      <motion.div

        className=""

        initial={{ opacity: 0, translateY: 30 }}

        animate={{ opacity: 1, translateY: 0 }}

        transition={{

          duration: 0.6,

          ease: [0.25, 0.1, 0.25, 1],

        }}

        viewport={{ once: true, amount: 0.1 }}

      >

        <div className="loginboxadmin">

          <img src={adminloginicon} alt="Admin Login Icon" />

          <h5>Admin</h5>

        </div>

        <div className="login-box">

          {error && <p className="error-message">{error}</p>}

          <form onSubmit={handleLogin}>

            <input

              type="text"

              name="username"

              placeholder="Username"

              value={formData.username}

              onChange={handleChange}

              required

            />

            <input

              type="password"

              name="password"

              placeholder="Password"

              value={formData.password}

              onChange={handleChange}

              required

            />

            <div className="adminloginbtn">

              <button type="submit">Login</button>

              <button type="submit" onClick={handleSignupRedirect}>Register</button>

            </div>

          </form>

          <p className="footer-note">Only authorized personnel are allowed to access this panel.</p>

        </div>

      </motion.div>

    </div>

  );

};

export default AdminLogin;

**adminLogin.js:**

const express = require('express');

const bcrypt = require('bcryptjs');

const jwt = require('jsonwebtoken');

const { Tupath\_usersModel, Employer\_usersModel, Project, AssessmentQuestion, Admin } = require("../models/Tupath\_users");

const router = express.Router();

const JWT\_SECRET = "your-secret-key";

const app = express();

// Admin Login Route

router.post('/api/admin/login', async (req, res) => {

  const { username, password } = req.body;

  try {

    const admin = await Admin.findOne({ username });

    if (!admin) {

      return res.status(400).json({ success: false, message: 'Invalid username or password.' });

    }

    const isPasswordValid = await bcrypt.compare(password, admin.password);

    if (!isPasswordValid) {

      return res.status(400).json({ success: false, message: 'Invalid username or password.' });

    }

    const token = jwt.sign({ id: admin.\_id }, JWT\_SECRET, { expiresIn: '1h' });

    // Set token in HTTP-only cookie

    res.cookie('adminToken', token, {

      httpOnly: true,

      secure: process.env.NODE\_ENV === 'production', // Use secure cookies in production

      sameSite: 'Strict', // Prevent CSRF

      maxAge: 3600000, // 1 hour

    });

    res.json({ success: true, message: 'Login successful.' });

  } catch (error) {

    console.error('Error logging in admin:', error);

    res.status(500).json({ success: false, message: 'Internal server error.' });

  }

});

module.exports = router;

**axiosInstancev2.js:**

import axios from 'axios';

const axiosInstancev2 = axios.create({

  baseURL: 'http://localhost:3001', // Update to your backend server's URL

  headers: {

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

    'Cache-Control': 'no-store', // Disables caching for API responses

    Pragma: 'no-cache',           // HTTP/1.0 compatibility

  },

  withCredentials: true, // Enables sending cookies with requests

});

// Remove the interceptor since tokens are no longer stored in localStorage

export default axiosInstancev2;

///