**User Registration**

const express = require('express');

const bcrypt = require('bcrypt');

const jwt = require('jsonwebtoken');

const User = require('./models/User'); // MongoDB User model

const router = express.Router();

// Registration endpoint

router.post('/register', async (req, res) => {

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

if (!name || !email || !password || !role) {

return res.status(400).json({ error: 'All fields are required' });

}

try {

const hashedPassword = await bcrypt.hash(password, 10);

const user = new User({ name, email, password: hashedPassword, role });

await user.save();

res.status(201).json({ message: 'User registered successfully' });

} catch (error) {

res.status(500).json({ error: 'Error registering user' });

}

});

module.exports = router;

**User Login**

// Login endpoint

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

const { email, password } = req.body;

if (!email || !password) {

return res.status(400).json({ error: 'Email and password are required' });

}

try {

const user = await User.findOne({ email });

if (!user) {

return res.status(404).json({ error: 'User not found' });

}

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

if (!isPasswordValid) {

return res.status(401).json({ error: 'Invalid credentials' });

}

const token = jwt.sign({ id: user.\_id, role: user.role }, 'your\_jwt\_secret', { expiresIn: '1h' });

res.status(200).json({ token });

} catch (error) {

res.status(500).json({ error: 'Error logging in' });

}

});

**Mark Attendance**

const multer = require('multer');

const Attendance = require('./models/Attendance');

const storage = multer.diskStorage({

destination: (req, file, cb) => cb(null, './uploads/'),

filename: (req, file, cb) => cb(null, `${Date.now()}-${file.originalname}`)

});

const upload = multer({ storage });

router.post('/mark-attendance', upload.single('selfie'), async (req, res) => {

const { userId } = req.body;

if (!req.file || !userId) {

return res.status(400).json({ error: 'Selfie and userId are required' });

}

try {

const attendance = new Attendance({

userId,

date: new Date(),

selfieUrl: req.file.path,

});

await attendance.save();

res.status(201).json({ message: 'Attendance marked successfully' });

} catch (error) {

res.status(500).json({ error: 'Error marking attendance' });

}

});

**Registration Form**

import React, { useState } from 'react';

import axios from 'axios';

const Register = () => {

const [formData, setFormData] = useState({ name: '', email: '', password: '', role: 'Student' });

const handleChange = (e) => {

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

};

const handleSubmit = async (e) => {

e.preventDefault();

try {

const response = await axios.post('/api/auth/register', formData);

alert(response.data.message);

} catch (error) {

alert('Error: ' + error.response?.data?.error || 'Server Error');

}

};

return (

<form onSubmit={handleSubmit}>

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

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

<input name="password" type="password" placeholder="Password" onChange={handleChange} required />

<select name="role" onChange={handleChange}>

<option value="Student">Student</option>

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

</select>

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

</form>

);

};

export default Register;

**Mark Attendance**

import React, { useState } from 'react';

import axios from 'axios';

const MarkAttendance = () => {

const [selfie, setSelfie] = useState(null);

const handleFileChange = (e) => {

setSelfie(e.target.files[0]);

};

const handleSubmit = async (e) => {

e.preventDefault();

const formData = new FormData();

formData.append('selfie', selfie);

formData.append('userId', 'your\_user\_id');

try {

const response = await axios.post('/api/students/attendance', formData);

alert(response.data.message);

} catch (error) {

alert('Error: ' + error.response?.data?.error || 'Server Error');

}

};

return (

<form onSubmit={handleSubmit}>

<input type="file" accept="image/\*" onChange={handleFileChange} required />

<button type="submit">Mark Attendance</button>

</form>

);

};

export default MarkAttendance;

**Add Teacher**

const express = require('express');

const bcrypt = require('bcrypt');

const User = require('./models/User'); // MongoDB User model

const { verifyAdmin } = require('./middleware/auth'); // Middleware to verify admin role

const router = express.Router();

router.post('/add-teacher', verifyAdmin, async (req, res) => {

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

if (!name || !email || !password) {

return res.status(400).json({ error: 'All fields are required' });

}

try {

const hashedPassword = await bcrypt.hash(password, 10);

const teacher = new User({ name, email, password: hashedPassword, role: 'Teacher', status: 'active' });

await teacher.save();

res.status(201).json({ message: 'Teacher added successfully' });

} catch (error) {

res.status(500).json({ error: 'Error adding teacher' });

}

});

module.exports = router;

**Manage User (List and Search)**

router.get('/users', verifyAdmin, async (req, res) => {

const { role, search, startDate, endDate } = req.query;

const filter = {};

if (role) filter.role = role;

if (search) filter.name = new RegExp(search, 'i');

if (startDate || endDate) {

filter.createdAt = {};

if (startDate) filter.createdAt.$gte = new Date(startDate);

if (endDate) filter.createdAt.$lte = new Date(endDate);

}

try {

const users = await User.find(filter).select('-password').sort({ createdAt: -1 });

res.status(200).json(users);

} catch (error) {

res.status(500).json({ error: 'Error fetching users' });

}

});

**Restrict User Login**

router.put('/users/:userId/status', verifyAdmin, async (req, res) => {

const { userId } = req.params;

const { status } = req.body;

if (!['active', 'inactive'].includes(status)) {

return res.status(400).json({ error: 'Invalid status' });

}

try {

const user = await User.findByIdAndUpdate(userId, { status }, { new: true });

if (!user) {

return res.status(404).json({ error: 'User not found' });

}

res.status(200).json({ message: `User status updated to ${status}` });

} catch (error) {

res.status(500).json({ error: 'Error updating user status' });

}

});

**Change Admin Password**

router.put('/change-password', verifyAdmin, async (req, res) => {

const { oldPassword, newPassword } = req.body;

if (!oldPassword || !newPassword) {

return res.status(400).json({ error: 'Old and new passwords are required' });

}

try {

const admin = await User.findById(req.user.id); // Admin's ID from the token

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

if (!isPasswordValid) {

return res.status(401).json({ error: 'Old password is incorrect' });

}

admin.password = await bcrypt.hash(newPassword, 10);

await admin.save();

res.status(200).json({ message: 'Password updated successfully' });

} catch (error) {

res.status(500).json({ error: 'Error updating password' });

}

});

**Add Teacher Form**

import React, { useState } from 'react';

import axios from 'axios';

const AddTeacher = () => {

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

const handleChange = (e) => {

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

};

const handleSubmit = async (e) => {

e.preventDefault();

try {

const token = localStorage.getItem('token'); // Assuming token is stored in localStorage

const response = await axios.post('/api/admin/add-teacher', formData, {

headers: { Authorization: `Bearer ${token}` },

});

alert(response.data.message);

} catch (error) {

alert('Error: ' + error.response?.data?.error || 'Server Error');

}

};

return (

<form onSubmit={handleSubmit}>

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

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

<input name="password" type="password" placeholder="Password" onChange={handleChange} required />

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

</form>

);

};

export default AddTeacher;

**Restrict User Login**

import React from 'react';

import axios from 'axios';

const RestrictUser = ({ userId, currentStatus }) => {

const toggleStatus = async () => {

try {

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

const newStatus = currentStatus === 'active' ? 'inactive' : 'active';

const response = await axios.put(

`/api/admin/users/${userId}/status`,

{ status: newStatus },

{ headers: { Authorization: `Bearer ${token}` } }

);

alert(response.data.message);

} catch (error) {

alert('Error: ' + error.response?.data?.error || 'Server Error');

}

};

return (

<button onClick={toggleStatus}>

{currentStatus === 'active' ? 'Disable Login' : 'Enable Login'}

</button>

);

};

export default RestrictUser;

**Teacher Login**

const express = require('express');

const bcrypt = require('bcrypt');

const jwt = require('jsonwebtoken');

const User = require('./models/User'); // MongoDB User model

const router = express.Router();

// Login endpoint for teachers

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

const { email, password } = req.body;

if (!email || !password) {

return res.status(400).json({ error: 'Email and password are required' });

}

try {

const teacher = await User.findOne({ email, role: 'Teacher' });

if (!teacher) {

return res.status(404).json({ error: 'Teacher not found' });

}

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

if (!isPasswordValid) {

return res.status(401).json({ error: 'Invalid credentials' });

}

const token = jwt.sign({ id: teacher.\_id, role: teacher.role }, 'your\_jwt\_secret', { expiresIn: '1h' });

res.status(200).json({ token });

} catch (error) {

res.status(500).json({ error: 'Error logging in' });

}

});

module.exports = router;

**View Attendance**

const Attendance = require('./models/Attendance'); // MongoDB Attendance model

// Endpoint to get attendance records

router.get('/attendance', async (req, res) => {

const { date, search } = req.query;

const filter = {};

if (date) {

const dayStart = new Date(date).setHours(0, 0, 0, 0);

const dayEnd = new Date(date).setHours(23, 59, 59, 999);

filter.date = { $gte: dayStart, $lte: dayEnd };

}

if (search) {

filter.name = new RegExp(search, 'i'); // Search by student name

}

try {

const attendance = await Attendance.find(filter)

.populate('userId', 'name') // Populate student names

.sort({ date: -1 });

res.status(200).json(attendance);

} catch (error) {

res.status(500).json({ error: 'Error fetching attendance' });

}

});

**Student List**

// Endpoint to fetch the student list for a teacher

router.get('/students', async (req, res) => {

const { search } = req.query;

const filter = { role: 'Student' };

if (search) {

filter.name = new RegExp(search, 'i');

}

try {

const students = await User.find(filter).select('name email createdAt').sort({ name: 1 });

res.status(200).json(students);

} catch (error) {

res.status(500).json({ error: 'Error fetching students' });

}

});

**View Student Profile**

// Endpoint to fetch a student's profile

router.get('/students/:studentId', async (req, res) => {

const { studentId } = req.params;

try {

const student = await User.findById(studentId).select('-password');

if (!student) {

return res.status(404).json({ error: 'Student not found' });

}

const attendance = await Attendance.find({ userId: studentId }).sort({ date: -1 });

res.status(200).json({ student, attendance });

} catch (error) {

res.status(500).json({ error: 'Error fetching student profile' });

}

});

**Teacher Login Form**

import React, { useState } from 'react';

import axios from 'axios';

const TeacherLogin = () => {

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

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

const handleLogin = async (e) => {

e.preventDefault();

try {

const response = await axios.post('/api/teacher/login', { email, password });

localStorage.setItem('token', response.data.token); // Save token

alert('Login successful!');

} catch (error) {

alert('Error: ' + error.response?.data?.error || 'Login failed');

}

};

return (

<form onSubmit={handleLogin}>

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

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

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

</form>

);

};

export default TeacherLogin;

**View Attendance Form**

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

import axios from 'axios';

const AttendanceList = () => {

const [attendance, setAttendance] = useState([]);

const [date, setDate] = useState('');

const [search, setSearch] = useState('');

useEffect(() => {

const fetchAttendance = async () => {

try {

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

const response = await axios.get('/api/teacher/attendance', {

headers: { Authorization: `Bearer ${token}` },

params: { date, search },

});

setAttendance(response.data);

} catch (error) {

alert('Error: ' + error.response?.data?.error || 'Unable to fetch attendance');

}

};

fetchAttendance();

}, [date, search]);

return (

<div>

<input type="date" onChange={(e) => setDate(e.target.value)} />

<input type="text" placeholder="Search student" onChange={(e) => setSearch(e.target.value)} />

<table>

<thead>

<tr>

<th>Student Name</th>

<th>Date</th>

<th>Punch-in Time</th>

<th>Selfie</th>

</tr>

</thead>

<tbody>

{attendance.map((entry) => (

<tr key={entry.\_id}>

<td>{entry.userId.name}</td>

<td>{new Date(entry.date).toLocaleDateString()}</td>

<td>{new Date(entry.date).toLocaleTimeString()}</td>

<td><img src={entry.selfieUrl} alt="Selfie" width="50" /></td>

</tr>

))}

</tbody>

</table>

</div>

);

};

export default AttendanceList;

**View Student List Form**

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

import axios from 'axios';

const StudentList = () => {

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

const [search, setSearch] = useState('');

useEffect(() => {

const fetchStudents = async () => {

try {

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

const response = await axios.get('/api/teacher/students', {

headers: { Authorization: `Bearer ${token}` },

params: { search },

});

setStudents(response.data);

} catch (error) {

alert('Error: ' + error.response?.data?.error || 'Unable to fetch students');

}

};

fetchStudents();

}, [search]);

return (

<div>

<input type="text" placeholder="Search student" onChange={(e) => setSearch(e.target.value)} />

<ul>

{students.map((student) => (

<li key={student.\_id}>{student.name} ({student.email})</li>

))}

</ul>

</div>

);

};

export default StudentList;

**Add Teacher**

const express = require('express');

const bcrypt = require('bcrypt');

const User = require('./models/User'); // MongoDB User model

const { verifyAdmin } = require('./middleware/auth'); // Middleware to verify admin role

const router = express.Router();

router.post('/add-teacher', verifyAdmin, async (req, res) => {

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

if (!name || !email || !password) {

return res.status(400).json({ error: 'All fields are required' });

}

try {

const hashedPassword = await bcrypt.hash(password, 10);

const teacher = new User({ name, email, password: hashedPassword, role: 'Teacher', status: 'active' });

await teacher.save();

res.status(201).json({ message: 'Teacher added successfully' });

} catch (error) {

res.status(500).json({ error: 'Error adding teacher' });

}

});

module.exports = router;

**Manage User (List and Search)**

router.get('/users', verifyAdmin, async (req, res) => {

const { role, search, startDate, endDate } = req.query;

const filter = {};

if (role) filter.role = role;

if (search) filter.name = new RegExp(search, 'i');

if (startDate || endDate) {

filter.createdAt = {};

if (startDate) filter.createdAt.$gte = new Date(startDate);

if (endDate) filter.createdAt.$lte = new Date(endDate);

}

try {

const users = await User.find(filter).select('-password').sort({ createdAt: -1 });

res.status(200).json(users);

} catch (error) {

res.status(500).json({ error: 'Error fetching users' });

}

});

**Restrict User Login**

router.put('/users/:userId/status', verifyAdmin, async (req, res) => {

const { userId } = req.params;

const { status } = req.body;

if (!['active', 'inactive'].includes(status)) {

return res.status(400).json({ error: 'Invalid status' });

}

try {

const user = await User.findByIdAndUpdate(userId, { status }, { new: true });

if (!user) {

return res.status(404).json({ error: 'User not found' });

}

res.status(200).json({ message: `User status updated to ${status}` });

} catch (error) {

res.status(500).json({ error: 'Error updating user status' });

}

});

**Change Admin Password**

router.put('/change-password', verifyAdmin, async (req, res) => {

const { oldPassword, newPassword } = req.body;

if (!oldPassword || !newPassword) {

return res.status(400).json({ error: 'Old and new passwords are required' });

}

try {

const admin = await User.findById(req.user.id); // Admin's ID from the token

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

if (!isPasswordValid) {

return res.status(401).json({ error: 'Old password is incorrect' });

}

admin.password = await bcrypt.hash(newPassword, 10);

await admin.save();

res.status(200).json({ message: 'Password updated successfully' });

} catch (error) {

res.status(500).json({ error: 'Error updating password' });

}

});

**Add Teacher Form**

import React, { useState } from 'react';

import axios from 'axios';

const AddTeacher = () => {

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

const handleChange = (e) => {

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

};

const handleSubmit = async (e) => {

e.preventDefault();

try {

const token = localStorage.getItem('token'); // Assuming token is stored in localStorage

const response = await axios.post('/api/admin/add-teacher', formData, {

headers: { Authorization: `Bearer ${token}` },

});

alert(response.data.message);

} catch (error) {

alert('Error: ' + error.response?.data?.error || 'Server Error');

}

};

return (

<form onSubmit={handleSubmit}>

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

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

<input name="password" type="password" placeholder="Password" onChange={handleChange} required />

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

</form>

);

};

export default AddTeacher;

**Manage User Form**

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

import axios from 'axios';

const UserList = () => {

const [users, setUsers] = useState([]);

const [filters, setFilters] = useState({ role: '', search: '', startDate: '', endDate: '' });

useEffect(() => {

const fetchUsers = async () => {

try {

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

const response = await axios.get('/api/admin/users', {

headers: { Authorization: `Bearer ${token}` },

params: filters,

});

setUsers(response.data);

} catch (error) {

alert('Error: ' + error.response?.data?.error || 'Unable to fetch users');

}

};

fetchUsers();

}, [filters]);

return (

<div>

<select onChange={(e) => setFilters({ ...filters, role: e.target.value })}>

<option value="">All Roles</option>

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

<option value="Student">Students</option>

</select>

<input type="text" placeholder="Search" onChange={(e) => setFilters({ ...filters, search: e.target.value })} />

<ul>

{users.map((user) => (

<li key={user.\_id}>{user.name} ({user.role})</li>

))}

</ul>

</div>

);

};

export default UserList;

**Change Admin Password**

import React, { useState } from 'react';

import axios from 'axios';

const ChangePassword = () => {

const [formData, setFormData] = useState({ oldPassword: '', newPassword: '' });

const handleChange = (e) => {

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

};

const handleSubmit = async (e) => {

e.preventDefault();

try {

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

const response = await axios.put('/api/admin/change-password', formData, {

headers: { Authorization: `Bearer ${token}` },

});

alert(response.data.message);

} catch (error) {

alert('Error: ' + error.response?.data?.error || 'Server Error');

}

};

return (

<form onSubmit={handleSubmit}>

<input name="oldPassword" type="password" placeholder="Old Password" onChange={handleChange} required />

<input name="newPassword" type="password" placeholder="New Password" onChange={handleChange} required />

<button type="submit">Change Password</button>

</form>

);

};

export default ChangePassword;