Admin signup and login , logout and manage curd operation (axios intercept used for automatically passing token in frontend concept)

Folder Structure

```
backend/
|
├── config/
|   └── db.js          # MongoDB connection
|
├── controllers/
|   ├── authController.js  # Signup, Login logic
|   └── recordController.js# CRUD logic for records
|
├── middleware/
|   └── authMiddleware.js  # JWT verification middleware
|
├── models/
|   ├── Admin.js         # Admin schema
|   └── Record.js        # Record schema
|
├── routes/
|   ├── authRoutes.js     # /signup and /login routes
|   └── recordRoutes.js    # /records routes
|
├── .env               # Secrets like JWT_SECRET
├── server.js           # Main entry point
├── package.json
```

└── README.md

mkdir backend

cd backend

npm init –y

npm install express mongoose cors bcryptjs jsonwebtoken

Step-by-Step Code Breakdown

1. **config/db.js:-**

```
const mongoose = require('mongoose');

const connectDB = async () => {
  try {
    await mongoose.connect('mongodb://localhost:27017/recordsDB', {
      useNewUrlParser: true,
      useUnifiedTopology: true,
    });
    console.log('MongoDB connected');
  } catch (err) {
    console.error('MongoDB connection failed:', err.message);
    process.exit(1);
  }
};

module.exports = connectDB;
```

2. **models/Admin.js**

```
const mongoose = require('mongoose');

const adminSchema = new mongoose.Schema({
  username: String,
  password: String,
});

module.exports = mongoose.model('Admin', adminSchema);
```

3. models/Record.js

```
const mongoose = require('mongoose');

const recordSchema = new mongoose.Schema({
  name: String,
  email: String,
  phone: String,
  city: String,
});

module.exports = mongoose.model('Record', recordSchema);
```

4. middleware/authMiddleware.js:-

```
const jwt = require('jsonwebtoken');

const verifyToken = (req, res, next) => {
  const authHeader = req.headers['authorization'];  // e.g. 'Bearer <token>'
  if (!authHeader) return res.status(401).json({ message: 'No token provided'
});

  // Extract token part after 'Bearer '
  const token = authHeader.split(' ')[1];
  if (!token) return res.status(401).json({ message: 'No token provided' });

  jwt.verify(token, process.env.JWT_SECRET, (err, decoded) => {
    if (err) return res.status(403).json({ message: 'Invalid token' });
    req.adminId = decoded.id;
    next();
  });
};
```

```
module.exports = verifyToken;
```

5. controllers/authController.js

```
const Admin = require('../models/Admin');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

exports.signup = async (req, res) => {
  try {
    const { username, password } = req.body;
    if (!username || !password)
      return res.status(400).json({ message: 'Username and password are
required' });

    const existing = await Admin.findOne({ username });
    if (existing)
      return res.status(400).json({ message: 'Username already exists' });

    const hashed = await bcrypt.hash(password, 10);
    await Admin.create({ username, password: hashed });

    res.json({ message: 'Admin created' });
  } catch (err) {
    console.error('Signup error:', err);
    res.status(500).json({ message: 'Internal server error' });
  }
};

exports.login = async (req, res) => {
  try {
    const { username, password } = req.body;
    const admin = await Admin.findOne({ username });

    if (!admin)
      return res.status(404).json({ message: 'Admin not found' });

    const isMatch = await bcrypt.compare(password, admin.password);
    if (!isMatch)
```

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

    const token = jwt.sign({ id: admin._id }, process.env.JWT_SECRET, {
      expiresIn: '1h',
    });

    res.json({ token });
  } catch (err) {
    res.status(500).json({ message: 'Login failed' });
  }
};
```

6. controllers/recordController.js

```
const Record = require('../models/Record');

exports.getAllRecords = async (req, res) => {
  const records = await Record.find();
  res.json(records);
};

exports.getRecordById = async (req, res) => {
  try {
    const record = await Record.findById(req.params.id);
    if (!record) return res.status(404).json({ message: 'Record not found' });
    res.json(record);
  } catch (err) {
    res.status(400).json({ message: 'Invalid record ID' });
  }
};

exports.createRecord = async (req, res) => {
  try {
    const newRecord = new Record(req.body);
    await newRecord.save();
    res.json({ message: 'Record added', record: newRecord });
  } catch (err) {
    res.status(500).json({ message: 'Error creating record' });
  }
};

exports.updateRecord = async (req, res) => {
  try {
    const updatedRecord = await Record.findByIdAndUpdate(req.params.id,
req.body, {
      new: true,
      runValidators: true,
    });
    if (!updatedRecord) return res.status(404).json({ message: 'Record not
found' });
    res.json({ message: 'Record updated', record: updatedRecord });
  } catch (err) {
    res.status(400).json({ message: 'Invalid update request' });
  }
};

exports.deleteRecord = async (req, res) => {
  try {
    const deleted = await Record.findByIdAndDelete(req.params.id);
    if (!deleted) return res.status(404).json({ message: 'Record not found'
});
    res.json({ message: 'Record deleted' });
  } catch (err) {
    res.status(400).json({ message: 'Invalid record ID' });
  }
};
```

routes/authRoutes.js

```
const express = require('express');
const router = express.Router();
const { signup, login } = require('../controllers/authController');

router.post('/signup', signup);
router.post('/login', login);

module.exports = router;
```

routes/recordRoutes.js

```
const express = require('express');
const router = express.Router();
const {
  getAllRecords,
  getRecordById,
  createRecord,
  updateRecord,
  deleteRecord,
} = require('../controllers/recordController');
const verifyToken = require('../middleware/authMiddleware');

router.get('/', verifyToken, getAllRecords);
router.get('/:id', verifyToken, getRecordById);
router.post('/', verifyToken, createRecord);
router.put('/:id', verifyToken, updateRecord);
router.delete('/:id', verifyToken, deleteRecord);

module.exports = router;
```

.env (create inside your backend project folder)

```
JWT_SECRET=a1b2c3d4e5f60123456789abcdef0123456789abcdef0123456789abcdef0123

PORT=5000
```

Don't forget to install dotenv

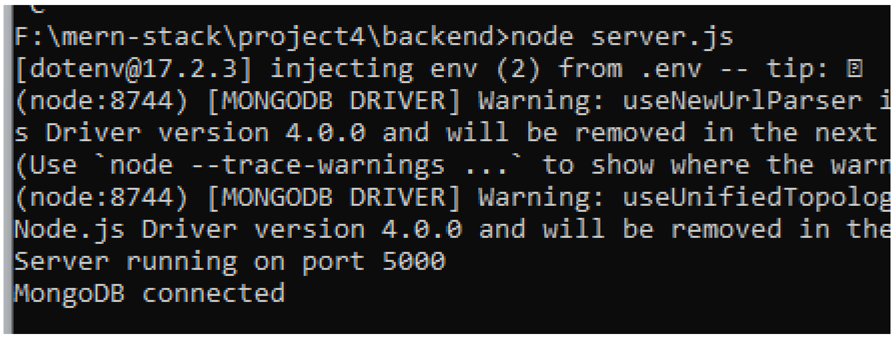
:

```
npm install dotenv
```

server.js file code:-

```
const express = require('express');
const cors = require('cors');
const dotenv = require('dotenv');
const connectDB = require('./config/db');

dotenv.config();
const app = express();

// Middleware
app.use(cors());
app.use(express.json());

// Connect DB
connectDB();

// Routes
app.use('/', require('./routes/authRoutes'));
app.use('/records', require('./routes/recordRoutes'));

// Start Server
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Then run it :- node server.js



```
F:\mern-stack\project4\backend>node server.js
[dotenv@17.2.3] injecting env (2) from .env -- tip: ⍰
(node:8744) [MONGODB DRIVER] Warning: useNewUrlParser i
s Driver version 4.0.0 and will be removed in the next
(Use `node --trace-warnings ...` to show where the warn
(node:8744) [MONGODB DRIVER] Warning: useUnifiedTopolog
Node.js Driver version 4.0.0 and will be removed in the
Server running on port 5000
MongoDB connected
```

⍰ frontend   Folder Structure

```
src/
│
├── api/
│   └── axiosInstance.js        # Axios with token interceptor
│
├── components/
│   ├── Auth/
│   │   └── Signup.js           # Signup form
│   └── Records/
│       └── RecordForm.js       # Add/Edit form
│       └── RecordList.js       # List with edit/delete
│
├── App.js                      # Main app component
└── index.js                    # React entry point
```

npx create-react-app frontend

cd frontend

npm install axios

npm start

Step 1: Axios Interceptor — src/api/axiosInstance.js

This sets up Axios to automatically attach the JWT token to every request and refresh it if needed.

```
import axios from 'axios';

const API = 'http://localhost:5000/';

const axiosInstance = axios.create({
  baseURL: API,
  headers: {
    'Content-Type': 'application/json',
  },
});

axiosInstance.interceptors.request.use(
  (config) => {
    const token = localStorage.getItem('token');
    if (token) {
      config.headers.Authorization = `Bearer ${token}`;
    }
    return config;
  },
  (error) => Promise.reject(error)
);

export default axiosInstance;
```

Step 2: App.js

```
import React, { useState, useEffect } from 'react';
import axios from './api/axiosInstance';
import Signup from './components/Auth/Signup';
import RecordForm from './components/Records/RecordForm';
import RecordList from './components/Records/RecordList';

function App() {
  const [records, setRecords] = useState([]);
  const [token, setToken] = useState(localStorage.getItem('token') || '');
  const [auth, setAuth] = useState({ username: '', password: '' });
  const [showSignup, setShowSignup] = useState(false);
  const [editRecord, setEditRecord] = useState(null);

  const handleLogin = async () => {
    try {
      const res = await axios.post('/login', auth);
      localStorage.setItem('token', res.data.token);
      setToken(res.data.token);
    } catch (err) {
      alert('Login failed');
    }
  };

  const handleLogout = () => {
    localStorage.removeItem('token');
    setToken('');
    setRecords([]);
  };

  const fetchRecords = async () => {
    try {
      const res = await axios.get('/records');
      setRecords(res.data);
    } catch (err) {
      console.error(err);
    }
  };

  useEffect(() => {
    if (token) fetchRecords();
  }, [token]);

  if (!token) {
    return (
      <div style={{ padding: '20px' }}>
        {showSignup ? (
          <>
            <Signup onSignupSuccess={() => {
              setShowSignup(false);
              alert('Signup successful! You can now log in.');
            }} />
          <p>
            Already have an account?{' '}
            <button onClick={() => setShowSignup(false)}>Login</button>
          </p>
        </>
      ) : (
        <>
          <h2>Admin Login</h2>
          <input
            placeholder="Username"
            value={auth.username}
            onChange={(e) => setAuth({ ...auth, username: e.target.value })}
          />
          <input
            type="password"
            placeholder="Password"
            value={auth.password}
            onChange={(e) => setAuth({ ...auth, password: e.target.value })}
          />
          <button onClick={handleLogin}>Login</button>
          <p>
            Don't have an account?{' '}
            <button onClick={() => setShowSignup(true)}>Sign Up</button>
          </p>
        </>
      )}
    </div>
  );
}

return (
  <div style={{ padding: '20px' }}>
    <button onClick={handleLogout}>Logout</button>
    <h2>{editRecord ? 'Edit Record' : 'Add Record'}</h2>
    <RecordForm
      record={editRecord}
      onSave={() => {
        setEditRecord(null);
        fetchRecords();
      }}
      onCancel={() => setEditRecord(null)}
    />
    <h2>Records</h2>
    <RecordList
      records={records}
      onEdit={setEditRecord}
      onDelete={fetchRecords}
    />
    </div>
  );
}

export default App;
```

src/components/Auth/Signup.js

```
import React, { useState } from 'react';
import axios from '../../api/axiosInstance';

function Signup({ onSignupSuccess }) {
  const [form, setForm] = useState({ username: '', password: '' });
  const [message, setMessage] = useState('');

  const handleChange = (e) => {
    setForm({ ...form, [e.target.name]: e.target.value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const res = await axios.post('/signup', form);
      setMessage(res.data.message || 'Signup successful');
      setForm({ username: '', password: '' });
      onSignupSuccess?.();
    } catch (err) {
      if (err.response) {
        setMessage(`Signup failed: ${err.response.data.message}`);
      } else {
        setMessage('Signup failed: ' + err.message);
      }
    }
  };

  return (
    <div>
      <h2>Admin Signup</h2>
      <form onSubmit={handleSubmit}>
        <input name="username" placeholder="Username" value={form.username}
onChange={handleChange} required />
```

```
        <input name="password" type="password" placeholder="Password"
value={form.password} onChange={handleChange} required />
        <button type="submit">Sign Up</button>
      </form>
      {message && <p>{message}</p>}
    </div>
  );
}

export default Signup;
```

src/components/Records/RecordForm.js

```
import React, { useState, useEffect } from 'react';
import axios from '../../api/axiosInstance';

function RecordForm({ record, onSave, onCancel }) {
  const [form, setForm] = useState({ name: '', email: '', phone: '', city: ''
});

  useEffect(() => {
    if (record) setForm(record);
    else setForm({ name: '', email: '', phone: '', city: '' });
  }, [record]);

  const handleChange = (e) => {
    setForm({ ...form, [e.target.name]: e.target.value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      if (record) {
        await axios.put(`/records/${record._id}`, form);
      } else {
        await axios.post('/records', form);
      }
      onSave();
      setForm({ name: '', email: '', phone: '', city: '' });
    } catch {
      alert('Failed to save record');
    }
  };

  return (
    <form onSubmit={handleSubmit}>
```

```
      <input name="name" placeholder="Name" value={form.name}
onChange={handleChange} />
      <input name="email" placeholder="Email" value={form.email}
onChange={handleChange} />
      <input name="phone" placeholder="Phone" value={form.phone}
onChange={handleChange} />
      <input name="city" placeholder="City" value={form.city}
onChange={handleChange} />
      <button type="submit">{record ? 'Update' : 'Add'}</button>
      {record && <button onClick={onCancel}>Cancel</button>}
    </form>
  );
}

export default RecordForm;
```

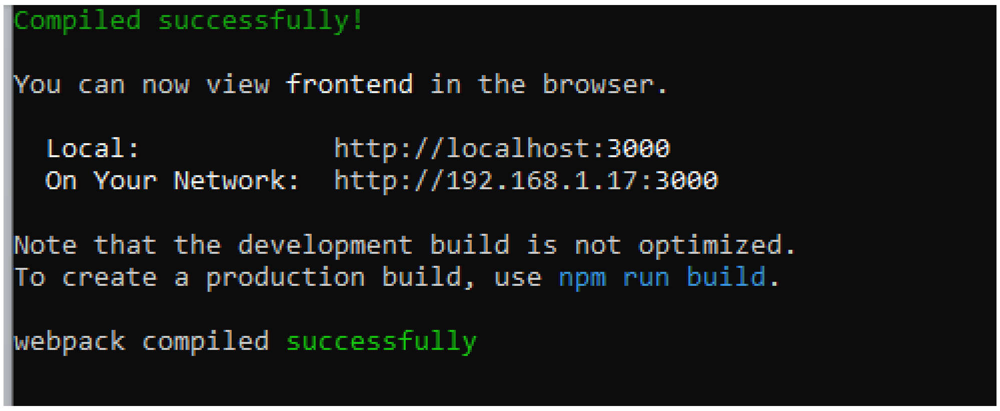
src/components/Records/RecordList.js

```
import React from 'react';
import axios from '../../api/axiosInstance';

function RecordList({ records, onEdit, onDelete }) {
  const handleDelete = async (id) => {
    if (!window.confirm('Are you sure you want to delete this record?'))
return;
    try {
      await axios.delete(`/records/${id}`);
      onDelete();
    } catch {
      alert('Failed to delete record');
    }
  };

  return (
    <ul>
      {records.map((rec) => (
        <li key={rec._id}>
          {rec.name} | {rec.email} | {rec.phone} | {rec.city}{' '}
          <button onClick={() => onEdit(rec)}>Edit</button>{' '}
          <button onClick={() => handleDelete(rec._id)}>Delete</button>
        </li>
      ))}
    </ul>
  );
}
```

```
export default RecordList;
```

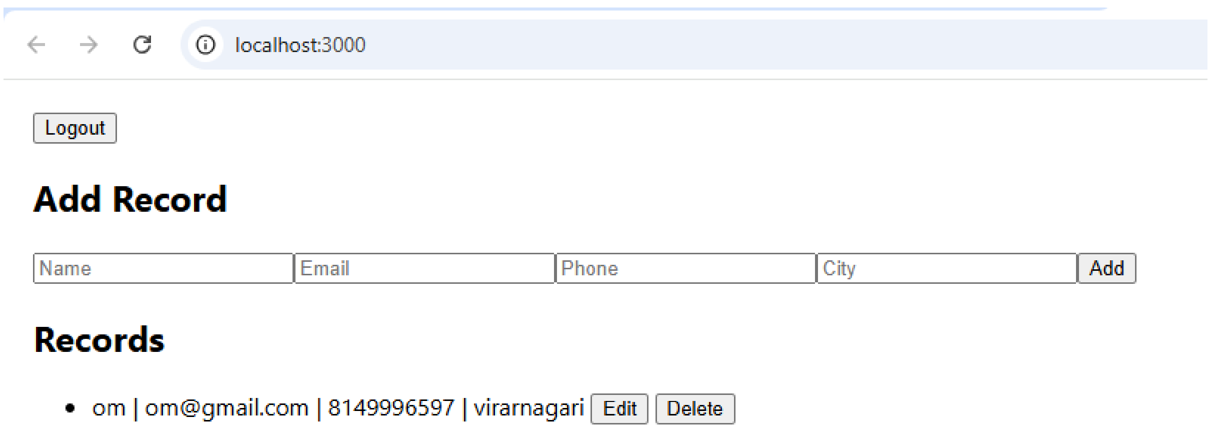
Then run :-

npm start



```
Compiled successfully!

You can now view frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.1.17:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Output:-



localhost:3000

Logout

Add Record

Name | Email | Phone | City | Add

Records

- om | om@gmail.com | 8149996597 | virarnagari Edit Delete