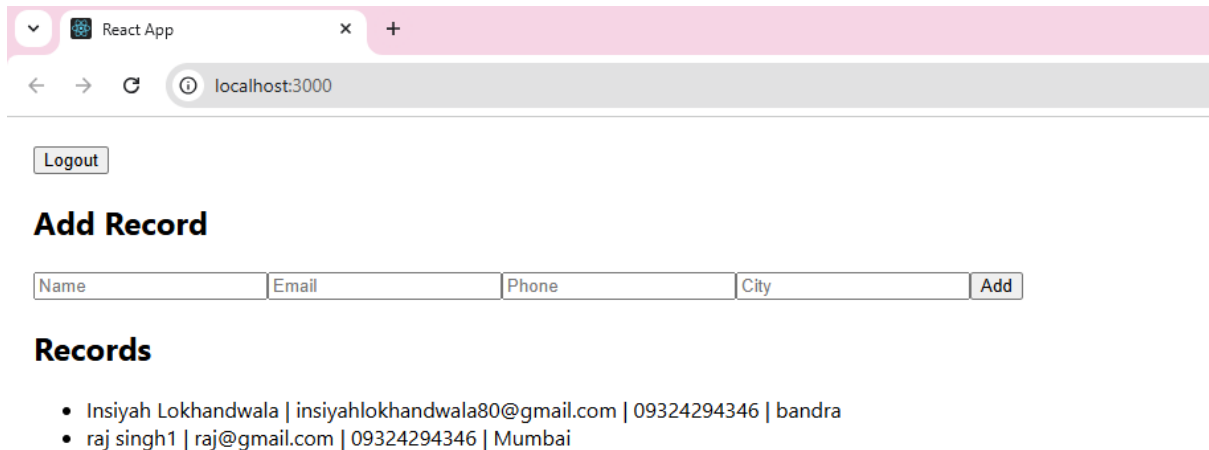


Admin Signup ,login , Logout and admin can add records and see records:-



Logout

Add Record

Name	Email	Phone	City	Add
------	-------	-------	------	-----

Records

- Insiyah Lokhandwala | insiyahlokhandwala80@gmail.com | 09324294346 | bandra
- raj singh1 | raj@gmail.com | 09324294346 | Mumbai

Backend :-

Server.js file code:-

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

const app = express();
app.use(express.json());
app.use(cors());

const JWT_SECRET =
'a1b2c3d4e5f60123456789abcdef0123456789abcdef0123456789abcdef0123'; // Change
this in production

// Connect to MongoDB
```

```
mongoose.connect('mongodb://localhost:27017/recordsDB', {
  useNewUrlParser: true,
  useUnifiedTopology: true
});

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

// Records Schema
const recordSchema = new mongoose.Schema({
  name: String,
  email: String,
  phone: String,
  city: String
});
const Record = mongoose.model('Record', recordSchema);

// Middleware to verify JWT token
function verifyToken(req, res, next) {
  const token = req.headers['authorization'];
  if (!token) return res.status(401).json({ message: 'No token provided' });

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

// Signup (create admin account)
app.post('/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('Error in signup:', err);
      res.status(500).json({ message: 'Internal server error' });
    }
  });

// Login
app.post('/login', async (req, res) => {
  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 }, JWT_SECRET, { expiresIn: '1h' });
  res.json({ token });
});

// Protected Routes
app.get('/records', verifyToken, async (req, res) => {
  const records = await Record.find();
  res.json(records);
});

app.post('/records', verifyToken, async (req, res) => {
  const newRecord = new Record(req.body);
  await newRecord.save();
  res.json({ message: 'Record added' });
});

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

```

Frontend code:-

src/Signup.js file code:-

```

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

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

```

```

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(`${API}/signup`, form, {
        headers: { 'Content-Type': 'application/json' }
      });
      setMessage(res.data.message || 'Signup successful');
      setForm({ username: '', password: '' });
      if (onSignupSuccess) onSignupSuccess();
    } catch (err) {
      console.error('Signup error:', err);

      if (err.response) {
        // server responded with a status outside 2xx
        setMessage(`Signup failed: ${err.response.data.message ||
err.response.statusText}`);
      } else if (err.request) {
        // request was made but no response
        setMessage('Signup failed: No response from server');
      } else {
        // other errors
        setMessage('Signup failed: ' + err.message);
      }
    }
  };

  return (
    <div style={{ padding: '20px' }}>
      <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;

```

App.js file code:-

```

import React, { useState, useEffect } from 'react';
import axios from 'axios';
import Signup from './Signup';

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

function App() {
  const [form, setForm] = useState({ name: '', email: '', phone: '', city: '' });
  const [records, setRecords] = useState([]);
  const [auth, setAuth] = useState({ username: '', password: '' });
  const [token, setToken] = useState(localStorage.getItem('token') || '');
  const [showSignup, setShowSignup] = useState(false);

  const handleLogin = async () => {
    try {
      const res = await axios.post(`${API}/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(`${API}/records`, {
      headers: { Authorization: token }
    });
    setRecords(res.data);
  } catch (err) {
    console.error(err);
  }
};

const addRecord = async (e) => {
  e.preventDefault();
  try {
    await axios.post(`${API}/records`, form, {
      headers: { Authorization: token }
    });
    setForm({ name: '', email: '', phone: '', city: '' });
    fetchRecords();
  } catch (err) {
    alert('Failed to add record');
  }
};

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

// 📌 If not logged in, show Login or Signup
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
          placeholder="Password"
          type="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>
    )}
  </div>
);
}

// Logged in view
return (
  <div style={{ padding: '20px' }}>
    <button onClick={handleLogout}>Logout</button>
    <h2>Add Record</h2>
    <form onSubmit={addRecord}>
      <input name="name" placeholder="Name" value={form.name} onChange={(e)
=> setForm({ ...form, name: e.target.value })} />
      <input name="email" placeholder="Email" value={form.email}
onChange={(e) => setForm({ ...form, email: e.target.value })} />
      <input name="phone" placeholder="Phone" value={form.phone}
onChange={(e) => setForm({ ...form, phone: e.target.value })} />
      <input name="city" placeholder="City" value={form.city} onChange={(e)
=> setForm({ ...form, city: e.target.value })} />
      <button type="submit">Add</button>
    </form>

    <h2>Records</h2>
    <ul>
      {records.map((rec, idx) => (
        <li key={idx}>{rec.name} | {rec.email} | {rec.phone} |
{rec.city}</li>
      ))}
    </ul>
  </div>
);

```

```
}  
  
export default App;
```