Express Session for login , logout, admin only can perform crud on record:-

models/User.js file code:-

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

const userSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  role: { type: String, enum: ['user', 'admin'], default: 'user' }
});

module.exports = mongoose.model('User', userSchema);
```

models/Record.js file code:-

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

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

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

server.js file code:-

```
const express = require('express');
const mongoose = require('mongoose');
const session = require('express-session');
const bodyParser = require('body-parser');
const cors = require('cors');
const authRoutes = require('./routes/auth');
const recordRoutes = require('./routes/record');
const User = require('./models/User');

const app = express();

// --- Middleware ---
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

app.use(cors({
  origin: 'http://localhost:3000',
  credentials: true
}));

app.use(session({
  secret: 'kfj39f#23kfKJf!239kfjsdK!39dkf',
  resave: false,
  saveUninitialized: false,
```

```
}));

// --- MongoDB Connection ---
mongoose.connect('mongodb://127.0.0.1:27017/authDemo', {
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(() => console.log('MongoDB connected'))
 .catch(err => console.log(err));


(async () => {
 const admin = await User.findOne({ username: 'admin' });
 if (!admin) {
  await User.create({ username: 'admin', password: 'admin123', role: 'admin' });
  console.log('✅ Admin user created: username=admin, password=admin123');
 }
})();

// --- Routes ---
app.use('/auth', authRoutes);
app.use('/records', recordRoutes);

app.get('/dashboard', (req, res) => {
 if (!req.session.userId) {
  return res.status(401).send('You must log in first.');
```

```
  }

  // 🔒 Check if admin
  if (req.session.role !== 'admin') {
    return res.status(403).send('Access denied: Admins only.');
  }

  res.send('Welcome Admin! You are viewing the dashboard.');
});

// --- Server Start ---
app.listen(5000, () => console.log('Server running on http://localhost:5000'));
```

middleware/authMiddleware.js file code:-

```
// middleware/authMiddleware.js
exports.isAuthenticated = (req, res, next) => {
  if (!req.session.userId) {
    return res.status(401).json({ message: 'Not logged in' });
  }
  next();
};

exports.isAdmin = (req, res, next) => {
  if (req.session.role !== 'admin') {
    return res.status(403).json({ message: 'Access denied: Admins only' });
  }
  next();
};
```

routes/auth.js file code:-

```
const express = require('express');
const router = express.Router();
const User = require('../models/User');

// Signup
router.post('/signup', async (req, res) => {
  const { username, password, role } = req.body;
  try {
    const user = new User({ username, password, role: role || 'user' });
    await user.save();
    res.status(201).send('User created');
  } catch (err) {
    res.status(400).send(err.message);
  }
});

// Login
router.post('/login', async (req, res) => {
  const { username, password } = req.body;
  const user = await User.findOne({ username });

  if (!user || user.password !== password) {
    return res.status(400).send('Invalid credentials');
  }
```

```
  req.session.userId = user._id;
  req.session.role = user.role;

  res.json({ message: 'Login successful', role: user.role });
});

// Logout
router.post('/logout', (req, res) => {
  req.session.destroy(() => {
    res.clearCookie('connect.sid');
    res.send('Logged out');
  });
});

// □ Check login status
router.get('/check', (req, res) => {
  if (req.session.userId) {
    res.json({
      loggedIn: true,
      role: req.session.role || 'user', // default to 'user' if not set
    });
  } else {
    res.json({
      loggedIn: false,
```

```
      role: null, // explicitly return null for clarity
    });
  }
});


module.exports = router;
```

routes/record.js:-

```
const express = require('express');
const Record = require('../models/Record');
const { isAuthenticated, isAdmin } = require('../middleware/authMiddleware');

const router = express.Router();

// ✅ Create record (authenticated users only)
router.post('/', isAuthenticated, async (req, res) => {
  const { name, email, city } = req.body;
  try {
    const record = new Record({ name, email, city });
    await record.save();
    res.status(201).send('Record created successfully');
  } catch (err) {
    res.status(500).send(err.message);
  }
});
```

```
// 🔒 Get all records (authenticated users only)
router.get('/', isAuthenticated, async (req, res) => {
  try {
    const records = await Record.find();
    res.json(records);
  } catch (err) {
    res.status(500).send(err.message);
  }
});

// 🔒 Update record (authenticated users only)
router.put('/:id', isAuthenticated, async (req, res) => {
  const { name, email, city } = req.body;
  try {
    const updated = await Record.findByIdAndUpdate(
      req.params.id,
      { name, email, city },
      { new: true }
    );
    if (!updated) return res.status(404).send('Record not found');
    res.send('Record updated successfully');
  } catch (err) {
    res.status(500).send(err.message);
```

```
  }
});

// 🗑 Delete record (admin only)
router.delete('/:id', isAuthenticated, isAdmin, async (req, res) => {
  try {
    const deleted = await Record.findByIdAndDelete(req.params.id);
    if (!deleted) return res.status(404).send('Record not found');
    res.send('Record deleted successfully');
  } catch (err) {
    res.status(500).send(err.message);
  }
});

module.exports = router;
```

Now Front-End code with React js with Axios :-

App.js file code:-

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import Signup from './components/Signup';
```

```
import Login from './components/Login';
import Dashboard from './components/Dashboard';
import Navbar from './components/Navbar';

function App() {
  return (
    <Router>
      <Navbar />
      <div style={{ padding: '20px' }}>
        <Routes>
          <Route path="/signup" element={<Signup />} />
          <Route path="/login" element={<Login />} />
          <Route path="/dashboard" element={<Dashboard />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

components/Navbar.js file code:-

```
import React, { useContext } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import axios from 'axios';
import { AuthContext } from '../context/AuthContext';

const Navbar = () => {
  const { loggedIn, role, logout } = useContext(AuthContext);
  const navigate = useNavigate();

  const handleLogout = async () => {
    try {
      await axios.post('http://localhost:5000/auth/logout', {}, { withCredentials: true });
      logout();
      alert('Logged out successfully');
      navigate('/login');
    } catch (err) {
      console.error(err);
```

```
  }
 };

 return (
  <nav style={{ padding: '10px', borderBottom: '1px solid #ccc' }}>
   {!loggedIn ? (
    <>
     <Link to="/signup" style={{ marginRight: 10 }}>Signup</Link>
     <Link to="/login" style={{ marginRight: 10 }}>Login</Link>
    </>
   ) : (
    <>
     {role === 'admin' && <Link to="/dashboard" style={{ marginRight: 10 }}>Dashboard</Link>}
     <button onClick={handleLogout}>Logout</button>
    </>
   )}
  </nav>
 );
};

export default Navbar;
```

components/Signup.js file code:-

```
import React, { useState } from 'react';
import axios from 'axios';
import { useNavigate } from 'react-router-dom';

const Signup = () => {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const navigate = useNavigate();

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await axios.post('http://localhost:5000/auth/signup', { username, password });
      alert('User registered successfully');
      navigate('/login');
    } catch (err) {
      alert(err.response.data);
```

```
  }
 };

 return (
  <div>
   <h2>Signup</h2>
   <form onSubmit={handleSubmit}>
    <input
     type="text"
     placeholder="Username"
     value={username}
     onChange={e => setUsername(e.target.value)}
    /><br /><br />
    <input
     type="password"
     placeholder="Password"
     value={password}
     onChange={e => setPassword(e.target.value)}
    /><br /><br />
    <button type="submit">Signup</button>
   </form>
  </div>
 );
};
```

```
export default Signup;
```

components/Login.js file code:-

```
import React, { useState, useContext } from 'react';
import axios from 'axios';
import { useNavigate } from 'react-router-dom';
import { AuthContext } from '../context/AuthContext';

const Login = () => {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const navigate = useNavigate();
  const { login } = useContext(AuthContext);

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const res = await axios.post('http://localhost:5000/auth/login',
        { username, password },
        { withCredentials: true }
      );
```

```
      login(res.data.role); // ✅ save role in context
      alert('Logged in successfully');
      navigate('/dashboard');
    } catch (err) {
      alert(err.response?.data || 'Login failed');
    }
  };

  return (
    <div>
      <h2>Login</h2>
      <form onSubmit={handleSubmit}>
        <input type="text" placeholder="Username" value={username} onChange={(e) =>
setUsername(e.target.value)} /><br /><br />
        <input type="password" placeholder="Password" value={password} onChange={(e) =>
setPassword(e.target.value)} /><br /><br />
        <button type="submit">Login</button>
      </form>
    </div>
  );
};

export default Login;
```

components/Dashboard.js file:-

```
import React, { useState, useEffect, useContext } from 'react';
import axios from 'axios';
import RecordForm from './RecordForm';
import RecordList from './RecordList';
import { AuthContext } from '../context/AuthContext';

const Dashboard = () => {
  const { loggedIn, role } = useContext(AuthContext);
  const [records, setRecords] = useState([]);
  const [editingRecord, setEditingRecord] = useState(null);
  const [loading, setLoading] = useState(true);

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

```
  const fetchRecords = async () => {
    try {
      const res = await axios.get('http://localhost:5000/records', { withCredentials: true });
      setRecords(res.data);
    } catch (err) {
      console.error('Error fetching records:', err);
    } finally {
      setLoading(false);
    }
  };


  const handleSave = async (record) => {
    try {
      if (editingRecord) {
        await axios.put(`http://localhost:5000/records/${editingRecord._id}`, record, { withCredentials: true
});
      } else {
        await axios.post('http://localhost:5000/records', record, { withCredentials: true });
      }
      fetchRecords();
      setEditingRecord(null);
    } catch (err) {
      console.error('Error saving record:', err);
    }
  };
```

```
const handleDelete = async (id) => {
  try {
    await axios.delete(`http://localhost:5000/records/${id}`, { withCredentials: true });
    fetchRecords();
  } catch (err) {
    console.error('Error deleting record:', err);
  }
};

if (loading) return <p>Loading dashboard...</p>;

if (!loggedIn) {
  return <p style={{ color: 'red' }}>You must be logged in to access the dashboard.</p>;
}

return (
  <div>
    <h1>Dashboard</h1>

    {role === 'admin' ? (
      <>
        <h2>Admin Dashboard</h2>
        <RecordForm onSave={handleSave} editingRecord={editingRecord} />
        <RecordList records={records} onEdit={setEditingRecord} onDelete={handleDelete} />
      </>
```

```
      ) : (
        <>
          <h3>User Dashboard</h3>
          <p>You can view records but not edit or delete them.</p>
          <RecordList records={records} />
        </>
      )}
    </div>
  );
};

export default Dashboard;
```

Context/AuthContext.js file code:-

```
import React, { createContext, useState, useEffect } from 'react';
import axios from 'axios';

export const AuthContext = createContext();
```

```
export const AuthProvider = ({ children }) => {
  const [loggedIn, setLoggedIn] = useState(false);
  const [role, setRole] = useState(null);

  useEffect(() => {
    const checkAuth = async () => {
      try {
        const res = await axios.get('http://localhost:5000/auth/check', { withCredentials: true });
        setLoggedIn(res.data.loggedIn);
        setRole(res.data.role || null);
      } catch {
        setLoggedIn(false);
        setRole(null);
      }
    };
    checkAuth();
  }, []);

  const login = (userRole) => {
    localStorage.setItem('loggedIn', 'true');
    localStorage.setItem('role', userRole);
    setLoggedIn(true);
    setRole(userRole);
  };
```

```
  const logout = () => {
    localStorage.removeItem('loggedIn');
    localStorage.removeItem('role');
    setLoggedIn(false);
    setRole(null);
  };

  return (
    <AuthContext.Provider value={{ loggedIn, role, login, logout }}>
      {children}
    </AuthContext.Provider>
  );
};
```

src/index.js file code:-

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { AuthProvider } from './context/AuthContext';



const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <AuthProvider>
    <App />
  </AuthProvider>
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

components/RecordFrom.js file code:-

```
import React, { useEffect, useState, useContext } from 'react';
import { AuthContext } from '../context/AuthContext';

const RecordForm = ({ onSave, editingRecord }) => {
  const [form, setForm] = useState({ name: '', email: '', city: '' });
  const { role } = useContext(AuthContext); // ✅ get role from context

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

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

```
const handleSubmit = (e) => {
  e.preventDefault();
  onSave(form);
};

// 🔒 If not admin, block form usage
if (role !== 'admin') {
  return (
    <div style={{ marginBottom: '20px', color: 'gray' }}>
      <h3>User Access</h3>
      <p>You can view records, but only admins can add or edit them.</p>
    </div>
  );
}

return (
  <form onSubmit={handleSubmit} style={{ marginBottom: '20px' }}>
    <h3>{editingRecord ? 'Edit Record' : 'Add Record'}</h3>

    <input
      name="name"
      placeholder="Name"
      value={form.name}
      onChange={handleChange}
```

```
        required
      />{' '}
      <input
        name="email"
        placeholder="Email"
        value={form.email}
        onChange={handleChange}
        required
      />{' '}
      <input
        name="city"
        placeholder="City"
        value={form.city}
        onChange={handleChange}
        required
      />{' '}
      <button type="submit">
        {editingRecord ? 'Update' : 'Add'}
      </button>
    </form>
  );
};

export default RecordForm;
```

components/RecordList.js file code:-

```
import React, { useContext } from 'react';
import { AuthContext } from '../context/AuthContext';

const RecordList = ({ records, onEdit, onDelete }) => {
  const { role } = useContext(AuthContext);

  return (
    <table>
      <thead>
        <tr>
          <th>Name</th>
          <th>Email</th>
          <th>City</th>
          {role === 'admin' && <th>Actions</th>}
```

```
        </tr>
      </thead>
      <tbody>
        {records.map((record) => (
          <tr key={record._id}>
            <td>{record.name}</td>
            <td>{record.email}</td>
            <td>{record.city}</td>
            {role === 'admin' && (
              <td>
                <button onClick={() => onEdit(record)}>Edit</button>
                <button onClick={() => onDelete(record._id)}>Delete</button>
              </td>
            )}
          </tr>
        ))}
      </tbody>
    </table>
  );
};

export default RecordList;
```

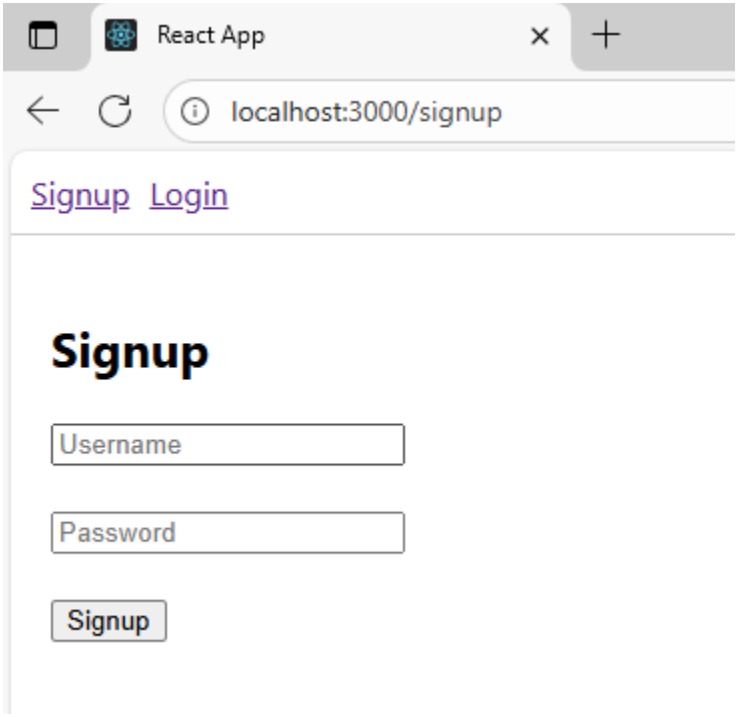
Now run backend :-

node server.js

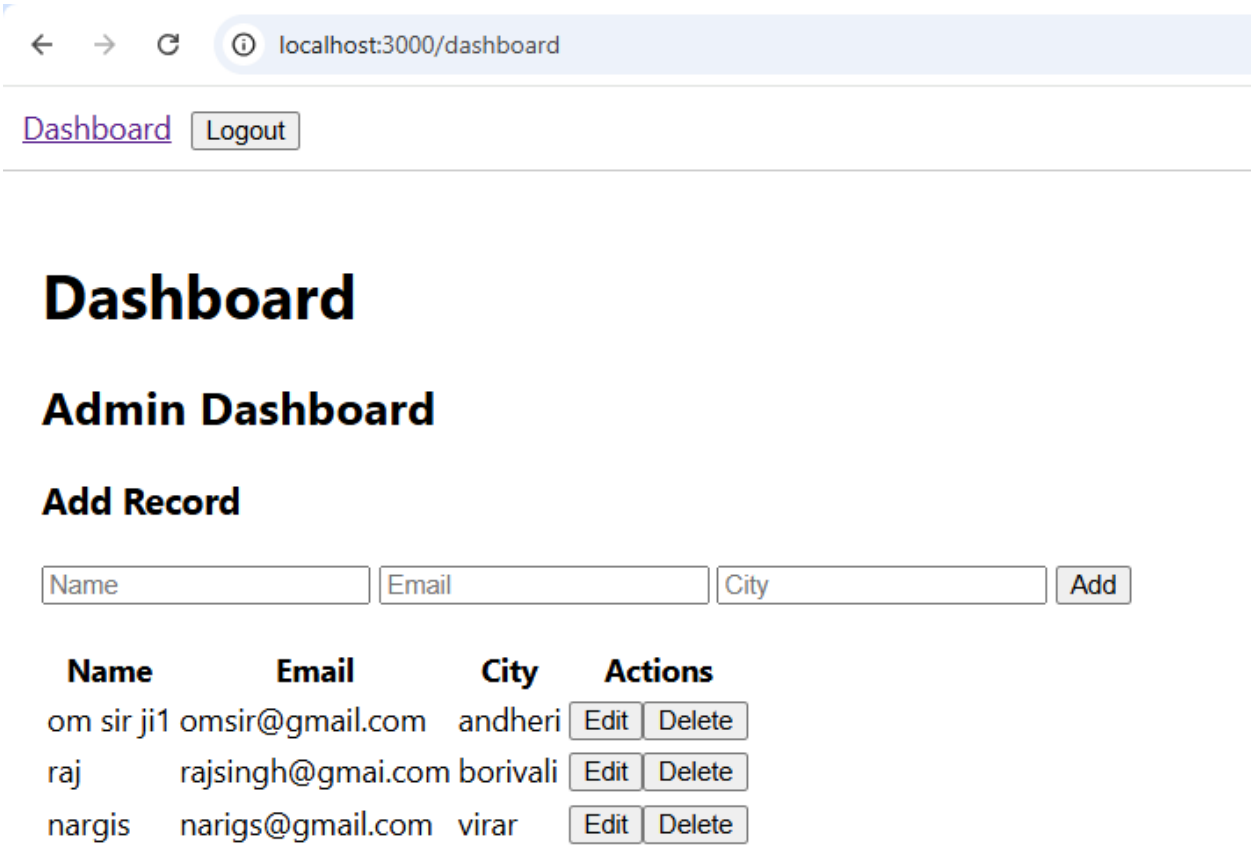
And Frontend with:-

npm start

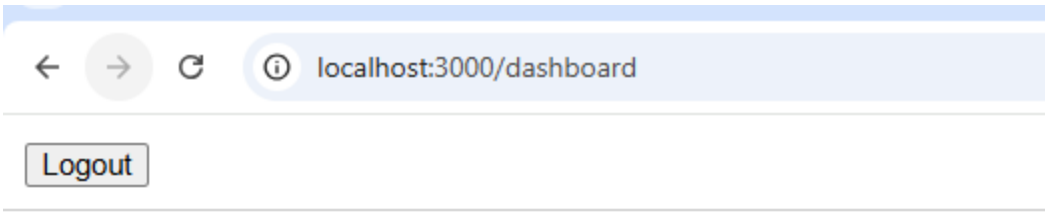
and check output:-



And after admin login :-



After user login:-



Dashboard

User Dashboard

You can view records but not edit or delete them.

Name	Email	City
om sir ji1	omsir@gmail.com	andheri
raj	rajsingh@gmai.com	borivali
nargis	narigs@gmail.com	virar