Full CRUD Operations:

- □ **Create** (Already Implemented)
- □ **Read** (Already Implemented)
- ✏□ **Update** (Edit Form + Update Button)
- □ **Delete** (Delete Button)

□ 1. Backend: Express.js + MongoDB

Install dependencies:

```
mkdir backend
Cd backend
npm init -y
npm install express mongoose cors
```

MVC-like Structure using:

- `models/Record.js` → Mongoose model
- `routes/recordRoutes.js` → Express routes
- `server.js` → App entry point

□ Final Structure

```
your-project/
│
├── models/
│   └── Record.js
│
├── routes/
│   └── recordRoutes.js
│
├── server.js
└── package.json
```

☐ 1. `models/Record.js`

```
// 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);
```

☐ 2. `routes/recordRoutes.js`:-

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


// ==============================
// GET all records
// ==============================
router.get('/', async (req, res) => {
  try {
    const records = await Record.find();
    res.json(records);
  } catch (err) {
    res.status(500).json({ error: 'Failed to fetch records' });
  }
});


// ==============================
// GET a single record by ID
// ==============================
router.get('/:id', async (req, res) => {
  try {
    const record = await Record.findById(req.params.id);
    if (!record) {
      return res.status(404).json({ error: 'Record not found' });
    }
```

```
    res.json(record);
  } catch (err) {
    res.status(500).json({ error: 'Failed to fetch record' });
  }
});


// ==============================
// POST a new record (Create)
// ==============================
router.post('/', async (req, res) => {
  try {
    const newRecord = new Record(req.body);
    await newRecord.save();
    res.json({ message: 'Record added successfully', record: newRecord });
  } catch (err) {
    res.status(400).json({ error: 'Failed to add record' });
  }
});


// ==============================
// PUT (Update) a record by ID
// ==============================
router.put('/:id', async (req, res) => {
  try {
    const updatedRecord = await Record.findByIdAndUpdate(
      req.params.id,
      req.body,
      { new: true } // Return the updated document
    );

    if (!updatedRecord) {
      return res.status(404).json({ error: 'Record not found' });
    }

    res.json({ message: 'Record updated successfully', record: updatedRecord
});
  } catch (err) {
    res.status(400).json({ error: 'Failed to update record' });
  }
});


// ==============================
// DELETE a record by ID
// ==============================
router.delete('/:id', async (req, res) => {
  try {
```

```
    const deletedRecord = await Record.findByIdAndDelete(req.params.id);

    if (!deletedRecord) {
      return res.status(404).json({ error: 'Record not found' });
    }

    res.json({ message: 'Record deleted successfully' });
  } catch (err) {
    res.status(500).json({ error: 'Failed to delete record' });
  }
});


// Export the router to use in server.js
module.exports = router;
```

3)server.js file code:-

```
// server.js
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const recordRoutes = require('./routes/recordRoutes');

const app = express();

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

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/recordsDB', {
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(() => console.log('MongoDB connected'))
  .catch(err => console.error('MongoDB connection error:', err));

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

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

□ 2. Frontend: React + Axios

Create React app (if not already):

```
npx create-react-app records-app
cd records-app
npm install axios
```

Replace `src/App.js` with:

```
// src/App.js
import React, { useEffect, useState } from 'react';
import axios from 'axios';

function App() {
  const [records, setRecords] = useState([]);
  const [form, setForm] = useState({ name: '', email: '', phone: '', city: ''
});
  const [isEditing, setIsEditing] = useState(false);
  const [editId, setEditId] = useState(null);

  // Fetch all records from backend
  const fetchRecords = async () => {
    try {
      const res = await axios.get('http://localhost:5000/records');
      setRecords(res.data);
    } catch (err) {
      console.error('Error fetching records:', err.message);
    }
  };
```

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

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

// Handle form submission (Create or Update)
const handleSubmit = async (e) => {
  e.preventDefault();

  if (isEditing) {
    // UPDATE
    try {
      await axios.put(`http://localhost:5000/records/${editId}`, form);
      setIsEditing(false);
      setEditId(null);
    } catch (err) {
      console.error('Update failed:', err.message);
    }
  } else {
    // CREATE
    try {
      await axios.post('http://localhost:5000/records', form);
    } catch (err) {
      console.error('Create failed:', err.message);
    }
  }

  setForm({ name: '', email: '', phone: '', city: '' });
  fetchRecords();
};

// Handle DELETE
const handleDelete = async (id) => {
  if (window.confirm('Are you sure you want to delete this record?')) {
    try {
      await axios.delete(`http://localhost:5000/records/${id}`);
      fetchRecords();
    } catch (err) {
      console.error('Delete failed:', err.message);
    }
  }
};
```

```
// Load selected record into form for editing
const handleEdit = (record) => {
  setForm({
    name: record.name,
    email: record.email,
    phone: record.phone,
    city: record.city,
  });
  setEditId(record._id);
  setIsEditing(true);
};

return (
  <div style={{ padding: '20px' }}>
    <h2>{isEditing ? 'Edit Record' : 'Add Record'}</h2>

    <form onSubmit={handleSubmit} style={{ marginBottom: '20px' }}>
      <input
        name="name"
        placeholder="Name"
        value={form.name}
        onChange={handleChange}
        required
      />
      <input
        name="email"
        placeholder="Email"
        value={form.email}
        onChange={handleChange}
        required
      />
      <input
        name="phone"
        placeholder="Phone"
        value={form.phone}
        onChange={handleChange}
        required
      />
      <input
        name="city"
        placeholder="City"
        value={form.city}
        onChange={handleChange}
        required
      />
      <button type="submit">{isEditing ? 'Update' : 'Add'}</button>
      {isEditing && (
        <button type="button" onClick={() => {
```

```
            setIsEditing(false);
            setForm({ name: '', email: '', phone: '', city: '' });
          }} style={{ marginLeft: '10px' }}>
            Cancel
          </button>
        )}
      </form>

      <h2>Records</h2>
      <ul>
        {records.map((rec) => (
          <li key={rec._id} style={{ marginBottom: '8px' }}>
            {rec.name} | {rec.email} | {rec.phone} | {rec.city}
            <button onClick={() => handleEdit(rec)} style={{ marginLeft:
'10px' }}>
              Edit
            </button>
            <button onClick={() => handleDelete(rec._id)} style={{ marginLeft:
'5px' }}>
              Delete
            </button>
          </li>
        ))}
      </ul>
    </div>
  );
}

export default App;
```

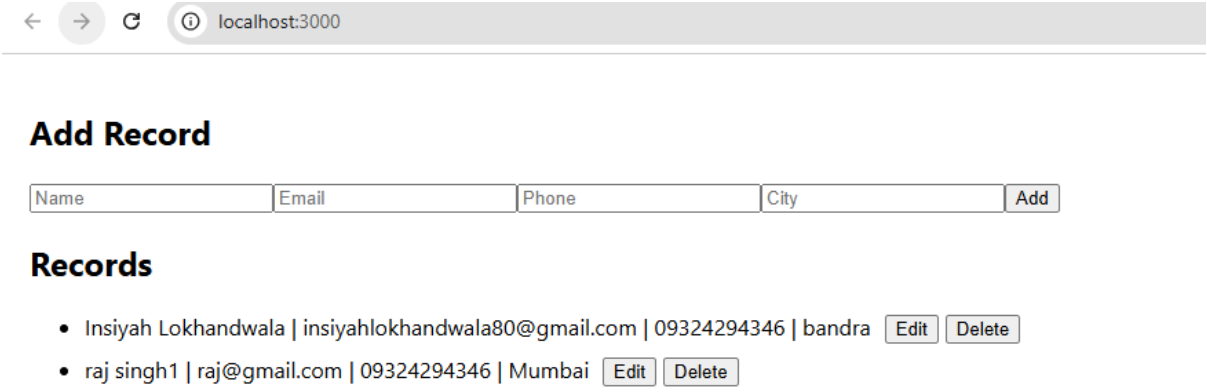
□ 3. Final Steps

Start backend server:

```
node server.js
```

Start frontend React app:

```
npm start
```



You can also organize your React code better and **split the CRUD functionality into separate components or files**, follow this structure:

□ Goal: Modular Code Structure

We'll break the logic and UI into:

File / Component	Purpose
`App.js`	Main container / wrapper
`components/AddEditForm.js`	Handles adding and editing a record
`components/RecordList.js`	Displays records, handles delete/edit
`api/recordApi.js`	Axios calls for CRUD

□ Step-by-Step Breakdown

1□ `src/api/recordApi.js`

```
// src/api/recordApi.js
import axios from 'axios';

const API_URL = 'http://localhost:5000/records';
```

```
export const getRecords = () => axios.get(API_URL);
export const createRecord = (data) => axios.post(API_URL, data);
export const updateRecord = (id, data) => axios.put(`${API_URL}/${id}`, data);
export const deleteRecord = (id) => axios.delete(`${API_URL}/${id}`);
```

---

2 `src/components/AddEditForm.js`(Handles form for add and edit.):-

```
// src/components/AddEditForm.js
import React from 'react';

function AddEditForm({ form, handleChange, handleSubmit, isEditing, cancelEdit
}) {
  return (
    <form onSubmit={handleSubmit} style={{ marginBottom: '20px' }}>
      <input name="name" placeholder="Name" value={form.name}
onChange={handleChange} required />
      <input name="email" placeholder="Email" value={form.email}
onChange={handleChange} required />
      <input name="phone" placeholder="Phone" value={form.phone}
onChange={handleChange} required />
      <input name="city" placeholder="City" value={form.city}
onChange={handleChange} required />
      <button type="submit">{isEditing ? 'Update' : 'Add'}</button>
      {isEditing && (
        <button type="button" onClick={cancelEdit} style={{ marginLeft: '10px'
}}>
          Cancel
        </button>
      )}
    </form>
  );
}
```

```
export default AddEditForm;
```

3 `src/components/RecordList.js`(Handles listing, delete and edit triggers.):-

```
// src/components/RecordList.js
import React from 'react';

function RecordList({ records, onEdit, onDelete }) {
  return (
    <ul>
      {records.map((rec) => (
        <li key={rec._id} style={{ marginBottom: '8px' }}>
          {rec.name} | {rec.email} | {rec.phone} | {rec.city}
          <button onClick={() => onEdit(rec)} style={{ marginLeft: '10px' }}>
            Edit
          </button>
          <button onClick={() => onDelete(rec._id)} style={{ marginLeft: '5px'
}}>
            Delete
          </button>
        </li>
      ))}
    </ul>
  );
}

export default RecordList;
```

4 `src/App.js` **(Updated)**(Now this file will look clean and focused.):-

```
// src/App.js
import React, { useEffect, useState } from 'react';
import {
  getRecords,
  createRecord,
  updateRecord,
```

```
  deleteRecord,
} from './api/recordApi';
import AddEditForm from './components/AddEditForm';
import RecordList from './components/RecordList';

function App() {
  const [records, setRecords] = useState([]);
  const [form, setForm] = useState({ name: '', email: '', phone: '', city: ''
});
  const [isEditing, setIsEditing] = useState(false);
  const [editId, setEditId] = useState(null);

  const fetchRecords = async () => {
    try {
      const res = await getRecords();
      setRecords(res.data);
    } catch (err) {
      console.error('Error fetching records:', err.message);
    }
  };

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

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

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      if (isEditing) {
        await updateRecord(editId, form);
        setIsEditing(false);
        setEditId(null);
      } else {
        await createRecord(form);
      }
      setForm({ name: '', email: '', phone: '', city: '' });
      fetchRecords();
    } catch (err) {
      console.error('Submit failed:', err.message);
    }
  };

  const handleDelete = async (id) => {
    if (window.confirm('Are you sure you want to delete this record?')) {
```

```
      try {
        await deleteRecord(id);
        fetchRecords();
      } catch (err) {
        console.error('Delete failed:', err.message);
      }
    }
  };

  const handleEdit = (record) => {
    setForm({
      name: record.name,
      email: record.email,
      phone: record.phone,
      city: record.city,
    });
    setEditId(record._id);
    setIsEditing(true);
  };

  const cancelEdit = () => {
    setIsEditing(false);
    setForm({ name: '', email: '', phone: '', city: '' });
  };

  return (
    <div style={{ padding: '20px' }}>
      <h2>{isEditing ? 'Edit Record' : 'Add Record'}</h2>

      <AddEditForm
        form={form}
        handleChange={handleChange}
        handleSubmit={handleSubmit}
        isEditing={isEditing}
        cancelEdit={cancelEdit}
      />

      <h2>Records</h2>
      <RecordList records={records} onEdit={handleEdit}
onDelete={handleDelete} />
    </div>
  );
}

export default App;
```

□ Folder Structure

```
src/
├── api/
│   └── recordApi.js
├── components/
│   ├── AddEditForm.js
│   └── RecordList.js
├── App.js
```

□ Benefits of This Structure

- **Reusability**: Each component does one job.
- **Maintainability**: Easy to debug and scale.
- **Cleaner code**: Main logic stays in `App.js`, heavy lifting is separated.