Here's a **complete CRUD (Create, Read, Update, Delete)** example using Html:-

- **MongoDB** (with **Mongoose**)
- **Express.js**
- **HTML form interface** (basic frontend)

What is MongoDB?

MongoDB is a **NoSQL database** that stores data in a flexible, **JSON-like format** called **BSON**. Unlike traditional SQL databases (like MySQL or PostgreSQL), it does **not use tables and rows** — instead, it uses **collections and documents**.

□ Key Concepts

Concept	SQL Equivalent	MongoDB Term	Example
Database	Database	Database	`myDatabase`
Table	Table	Collection	`users`, `orders`
Row	Row	Document	`{ name: "Alice", age: 25 }`
Column	Field	Field	`name`, `email`, `age`

□ Example MongoDB Document

```
{
  "_id": "64cdb123456abc7890",
  "name": "Alice",
  "email": "alice@example.com",
  "age": 25,
  "registered": true
}
```

□ Why Use MongoDB?

□ Advantages:

- □ **Flexible schema** – fields can vary between documents.
- □ **Fast & scalable** – great for big, modern apps.
- □ **Horizontal scaling** – easy to distribute across multiple servers.
- □ **Built for JSON** – works well with JavaScript & web apps.
- ☘□ **MongoDB Atlas** – free cloud hosting with backups and monitoring.

□ MongoDB vs SQL

Feature	SQL (MySQL, PostgreSQL)	MongoDB
Schema	Fixed	Flexible
Data Structure	Tables & Rows	Collections & Documents
Language	SQL	MongoDB Query Language
Relationships	JOINs	Embedded docs / `$lookup`
Scalability	Vertical (scale up)	Horizontal (scale out)

□ How You Use MongoDB

- In **Node.js** (MERN stack), using:

  ```
  npm install mongoose
  ```

  And connect with:

  ```
  mongoose.connect("mongodb://localhost:27017/mydb")
  ```

- In the **Mongo Shell** (`mongosh`):

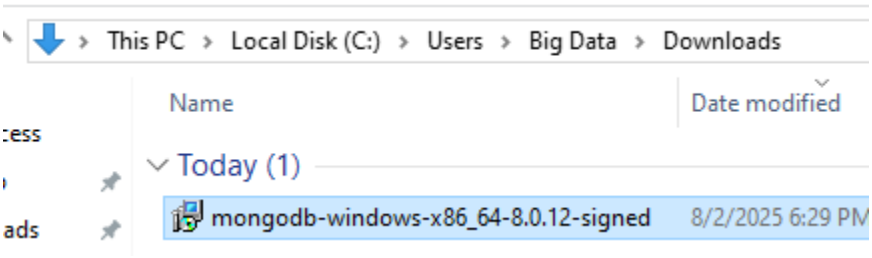
  ```
  use mydb
  db.users.insertOne({ name: "Bob", age: 30 })
  db.users.find()
  ```

Install MongoDB Locally (Offline & Dev Use)

Steps:-

1. Download MongoDB

- Go to https://www.mongodb.com/try/download/community
- Choose your OS: **Windows**, **macOS**, or **Linux**
- Install the **MongoDB Community Server**



2. Install MongoDB Compass (Optional GUI)

- Download from https://www.mongodb.com/try/download/compass
- This is a visual tool to view your databases.

□ Goal:

A basic app where you can:

1. **Create** a user

2. **Read** (view) all users
3. **Update** a user
4. **Delete** a user

---

1. Setup Project

```
mkdir crud-app && cd crud-app
npm init -y
npm install express mongoose body-parser
```

□ Project Structure

```
crud-app/
├── models/
│   └── User.js
├── public/
│   ├── index.html
│   └── edit.html
├── server.js
└── package.json
```

□ 1. Initialize and Install Packages

```
mkdir crud-app && cd crud-app
npm init -y
npm install express mongoose body-parser
```

□ 2. `models/User.js`:-

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

const userSchema = new mongoose.Schema({
  name: String,
  email: String,
  age: Number,
});

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

□ 3. `public/index.html` — List & Create Users

```
<!DOCTYPE html>
<html>
<head>
  <title>Users</title>
</head>
<body>
  <h1>Create User</h1>
  <form action="/create" method="POST">
    <input name="name" placeholder="Name" required />
    <input name="email" placeholder="Email" required />
    <input name="age" type="number" placeholder="Age" required />
    <button type="submit">Create</button>
  </form>

  <h1>All Users</h1>
  <div id="users"></div>

  <script>
    fetch('/users')
      .then(res => res.json())
      .then(data => {
        const container = document.getElementById('users');
        data.forEach(user => {
          container.innerHTML += `
            <p>
              ${user.name} (${user.email}, ${user.age})<br/>
              <a href="/edit.html?id=${user._id}">Edit</a> |
              <a href="/delete/${user._id}">Delete</a>
            </p>`;
        });
      });
  </script>
</body>
</html>
```

---

4. `public/edit.html` — Update User Form

```
<!DOCTYPE html>
<html>
```

```
<head>
  <title>Edit User</title>
</head>
<body>
  <h1>Edit User</h1>
  <form id="editForm" method="POST">
    <input name="name" id="name" placeholder="Name" required />
    <input name="email" id="email" placeholder="Email" required />
    <input name="age" id="age" type="number" placeholder="Age" required />
    <button type="submit">Update</button>
  </form>

  <script>
    const params = new URLSearchParams(window.location.search);
    const id = params.get("id");

    fetch(`/user/${id}`)
      .then(res => res.json())
      .then(user => {
        document.getElementById('name').value = user.name;
        document.getElementById('email').value = user.email;
        document.getElementById('age').value = user.age;
      });

    document.getElementById('editForm').action = `/update/${id}`;
  </script>
</body>
</html>
```

□ 5. `server.js` — Main Server Code

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const path = require('path');
const User = require('./models/User');

const app = express();

// Middleware
app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.json());
app.use(express.static(path.join(__dirname, 'public')));

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

// Create
app.post('/create', async (req, res) => {
```

```
  const { name, email, age } = req.body;
  await new User({ name, email, age }).save();
  res.redirect('/');
});

// Read all
app.get('/users', async (req, res) => {
  const users = await User.find();
  res.json(users);
});

// Read single
app.get('/user/:id', async (req, res) => {
  const user = await User.findById(req.params.id);
  res.json(user);
});

// Update
app.post('/update/:id', async (req, res) => {
  const { name, email, age } = req.body;
  await User.findByIdAndUpdate(req.params.id, { name, email, age });
  res.redirect('/');
});

// Delete
app.get('/delete/:id', async (req, res) => {
  await User.findByIdAndDelete(req.params.id);
  res.redirect('/');
});

// Server
app.listen(3000, () => {
  console.log('Server running on http://localhost:3000');
});
```

□ How to Run It

1. Start MongoDB (if running locally).
2. Run the server:

   ```
   node server.js
   ```

3. Open your browser at:

   ```
   http://localhost:3000
   ```

🎯 Goal: CRUD app using MongoDB, Express.js, and EJS

I'll show you how to:

1. Set up an EJS-based Express app
2. Create a form to insert data
3. Use MongoDB via Mongoose

📁 Folder Structure

```
crud-app/
├── models/
│   └── User.js
├── views/
│   ├── index.ejs
│   └── edit.ejs
├── public/
├── server.js
└── package.json
```

🛠 1. Set up the Project

```
mkdir crud-app && cd crud-app
npm init -y
npm install express mongoose body-parser ejs
```

📄 2. `models/User.js` — Mongoose Schema

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

const userSchema = new mongoose.Schema({
  name: String,
  email: String,
  age: Number,
});

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

📄 3. `views/index.ejs` — Home Page + Form

```
<!DOCTYPE html>
<html>
<head>
  <title>Users</title>
</head>
<body>
  <h1>Create User</h1>
  <form action="/create" method="POST">
    <input name="name" placeholder="Name" required />
    <input name="email" placeholder="Email" required />
    <input name="age" type="number" placeholder="Age" required />
    <button type="submit">Create</button>
  </form>

  <h2>All Users</h2>
  <% users.forEach(user => { %>
    <p>
      <%= user.name %> (<%= user.email %>, <%= user.age %>)
      | <a href="/edit/<%= user._id %>">Edit</a>
      | <a href="/delete/<%= user._id %>">Delete</a>
    </p>
  <% }) %>
</body>
</html>
```

□ 4. `views/edit.ejs` — Edit Form

```
<!DOCTYPE html>
<html>
<head>
  <title>Edit User</title>
</head>
<body>
  <h1>Edit User</h1>
  <form action="/update/<%= user._id %>" method="POST">
    <input name="name" value="<%= user.name %>" required />
    <input name="email" value="<%= user.email %>" required />
    <input name="age" type="number" value="<%= user.age %>" required />
    <button type="submit">Update</button>
  </form>
</body>
</html>
```

□ 5. `server.js` — Main Server Logic

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const User = require('./models/User');
const path = require('path');

const app = express();
```

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

// Middleware
app.use(bodyParser.urlencoded({ extended: true }));
app.set('view engine', 'ejs');
app.use(express.static(path.join(__dirname, 'public')));

// Routes

// View all users
app.get('/', async (req, res) => {
  const users = await User.find();
  res.render('index', { users });
});

// Create user
app.post('/create', async (req, res) => {
  const { name, email, age } = req.body;
  await new User({ name, email, age }).save();
  res.redirect('/');
});

// Edit form
app.get('/edit/:id', async (req, res) => {
  const user = await User.findById(req.params.id);
  res.render('edit', { user });
});

// Update user
app.post('/update/:id', async (req, res) => {
  const { name, email, age } = req.body;
  await User.findByIdAndUpdate(req.params.id, { name, email, age });
  res.redirect('/');
});

// Delete user
app.get('/delete/:id', async (req, res) => {
  await User.findByIdAndDelete(req.params.id);
  res.redirect('/');
});

// Start server
app.listen(3000, () => {
  console.log('Server running at http://localhost:3000');
});
```

□ How to Run It

1. Start MongoDB (`mongod` in terminal)
2. Start the app:

   ```
   node server.js
   ```

3. Open browser:

   ```
   http://localhost:3000
   ```