Here is a complete step-by-step guide for creating four APIs (Create, Read, Update, Delete) using Node.js, Express.js, MongoDB, and MongoDB Compass:

✅ Step 1: Install Required Software

1. Install Node.js:
   - Download from https://nodejs.org/
   - Verify installation:
   - `node -v`
   - `npm -v`
2. Install MongoDB:
   - Download MongoDB Community Edition from: https://www.mongodb.com/try/download/community
   - Install MongoDB Compass (GUI for DB) from: https://www.mongodb.com/products/compass
   - Start MongoDB Server:
     On Windows:
   - `"C:\Program Files\MongoDB\Server\6.0\bin\mongod.exe"`

     (You may add MongoDB to PATH variable or use MongoDB as a Windows service.)

3. Verify MongoDB installation:
4. `mongo`

   You should see the MongoDB shell prompt.

✅ Step 2: Set Up Your Project

1. Create a folder:
2. `mkdir curd-api`
3. `cd curd-api`
4. Initialize the project:
5. `npm init -y`
6. Install required packages:
7. `npm install express mongoose dotenv`
8. `npm install --save-dev nodemon`
9. Modify package.json:
10. `"scripts": {`
11. `  "start": "nodemon index.js"`
12. `}`

✅ Step 3: Set Up Project Structure

```
📁 curd-api
├── index.js
├── .env
├── models
│   └── Item.js
├── routes
```

```
│   └── items.js
├── package.json
```

✅ Step 4: Create .env File

Create a file named .env:

```
PORT=5000
MONGO_URI=mongodb://127.0.0.1:27017/curd_api_db
```

✅ Step 5: index.js (Main Server File)

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

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

const itemRoutes = require('./routes/items');
app.use('/api/items', itemRoutes);

mongoose.connect(process.env.MONGO_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true
})
.then(() => console.log('MongoDB connected'))
.catch(err => console.log(err));

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

✅ Step 6: models/Item.js

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

const ItemSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true
  },
  quantity: {
    type: Number,
    default: 0
  }
});

module.exports = mongoose.model('Item', ItemSchema);
```

✅ Step 7: routes/items.js

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

// CREATE
```

```
router.post('/', async (req, res) => {
  try {
    const newItem = new Item(req.body);
    const saved = await newItem.save();
    res.status(201).json(saved);
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
});

// READ ALL
router.get('/', async (req, res) => {
  try {
    const items = await Item.find();
    res.json(items);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// UPDATE
router.put('/:id', async (req, res) => {
  try {
    const updatedItem = await Item.findByIdAndUpdate(
      req.params.id,
      req.body,
      { new: true }
    );
    res.json(updatedItem);
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
});

// DELETE
router.delete('/:id', async (req, res) => {
  try {
    const deletedItem = await Item.findByIdAndDelete(req.params.id);
    res.json({ message: 'Deleted', item: deletedItem });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

module.exports = router;
```

✅ Step 8: Run the API

```
npm start
```

You should see:

```
MongoDB connected
Server started on port 5000
```

✅ Step 9: Test API

Use Postman or curl:

- POST: http://localhost:5000/api/items — to add an item.
- GET: http://localhost:5000/api/items — to list items.
- PUT: http://localhost:5000/api/items/ — to update.
- DELETE: http://localhost:5000/api/items/ — to delete.

✅ Step 10: View in MongoDB Compass

1. Open MongoDB Compass.
2. Connect to:
3. `mongodb://127.0.0.1:27017`
4. Look for your database: curd_api_db
5. Check the "items" collection to see data added through the API.

Would you like a diagram or Postman collection to help with testing the API?

📌 Test Each CRUD Operation

Operation	Method	URL
Create	POST	`http://localhost:5000/api/items`
Read	GET	`http://localhost:5000/api/items`
Update	PUT	`http://localhost:5000/api/items/<id>`
Delete	DELETE	`http://localhost:5000/api/items/<id>`

🟢 Create Item (POST)

- **URL**: `http://localhost:5000/api/items`
- **Method**: POST
- **Body** → raw → JSON:

```
json
CopyEdit
{
  "name": "Apples",
  "quantity": 5
}
```

- Click **Send**
- Response: 201 Created with item JSON

● Read All Items (GET)

- **URL**: `http://localhost:5000/api/items`
- **Method**: GET

- Click **Send**
- Response: Array of items

🟡 **Update Item (PUT)**

- First, get an ID from a previous GET or POST response.
- **URL**: `http://localhost:5000/api/items/<paste_item_id_here>`
- **Method**: PUT
- **Body** → raw → JSON:

```
json
CopyEdit
{
  "name": "Bananas",
  "quantity": 10
}
```

- Click **Send**

⚫ **Delete Item (DELETE)**

- **URL**: `http://localhost:5000/api/items/<paste_item_id_here>`
- **Method**: DELETE
- Click **Send**