Perfect — let's build a simple Express + Mongoose backend for a collection called `Record` that stores:

➡ **`name`, `city`, and `phone`.**

I'll show you the **complete working example**, step-by-step

Step 1: Setup Project

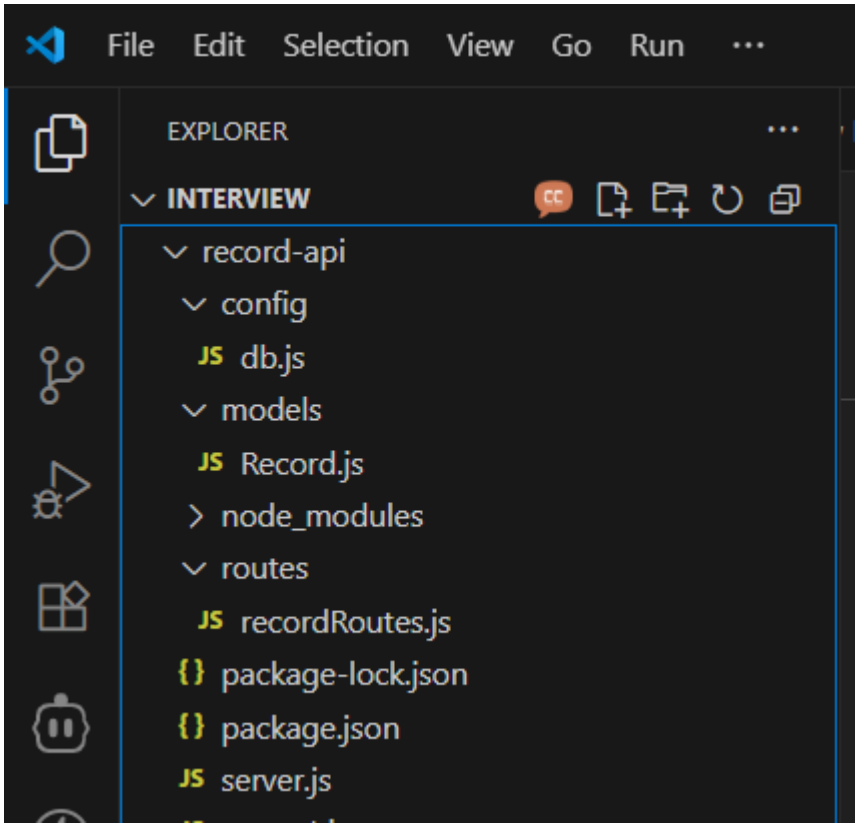
In your command prompt or terminal:

```
mkdir record-api
cd record-api
npm init -y
npm install express mongoose
```

Project Structure :-

Here's how your folders/files should look

```
record-api/
│
├── server.js
├── config
│   └── db.js
├── models
│   └── Record.js
└── routes
└── recordRoutes.js
```



config/db.js file code:-

```
const mongoose = require('mongoose');
const connectDB = async () => {
try {
await mongoose.connect('mongodb://localhost:27017/recorddb');
console.log('✅ MongoDB connected');
} catch (error) {
console.error('❌ MongoDB connection failed:', error.message);
process.exit(1);
}
};
module.exports = connectDB;
```

models/Record.js file code:-

```
// models/Record.js
const mongoose = require('mongoose');

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

// Create model
const Record = mongoose.model('Record', recordSchema);

module.exports = Record;
```

routes/recordRoutes.js file code:-

```
// routes/recordRoutes.js
const express = require('express');
const router = express.Router();
const Record = require('../models/Record');
// ➤ GET all records
router.get('/', async (req, res) => {
const records = await Record.find();
res.json(records);
});
// ➤ POST (Add new record)
router.post('/', async (req, res) => {
const record = new Record({
name: req.body.name,
city: req.body.city,
phone: req.body.phone
});
await record.save();
res.send('✅ Record saved successfully!');
});
// ➤ GET by ID
router.get('/:id', async (req, res) => {
const record = await Record.findById(req.params.id);
if (!record) return res.status(404).send('Record not found');
res.json(record);
});

// ➤ PUT (Update)
router.put('/:id', async (req, res) => {
const record = await Record.findByIdAndUpdate(
req.params.id,
{ name: req.body.name, city: req.body.city, phone: req.body.phone },
{ new: true }
);
if (!record) return res.status(404).send('Record not found');
res.json(record);
});
// ➤ DELETE
router.delete('/:id', async (req, res) => {
const record = await Record.findByIdAndDelete(req.params.id);
if (!record) return res.status(404).send('Record not found');
res.send('✅ Record deleted successfully!');
});
module.exports = router;
```

server1.js file code:-

```
const express = require('express');
const connectDB = require('./config/db');
const recordRoutes = require('./routes/recordRoutes');
const app = express();
// Middleware
app.use(express.json());
// Connect to MongoDB
connectDB();
// Routes
app.use('/api/records', recordRoutes);
// Start server
app.listen(5000, () => console.log('🚀 Server running on port 5000'));
```

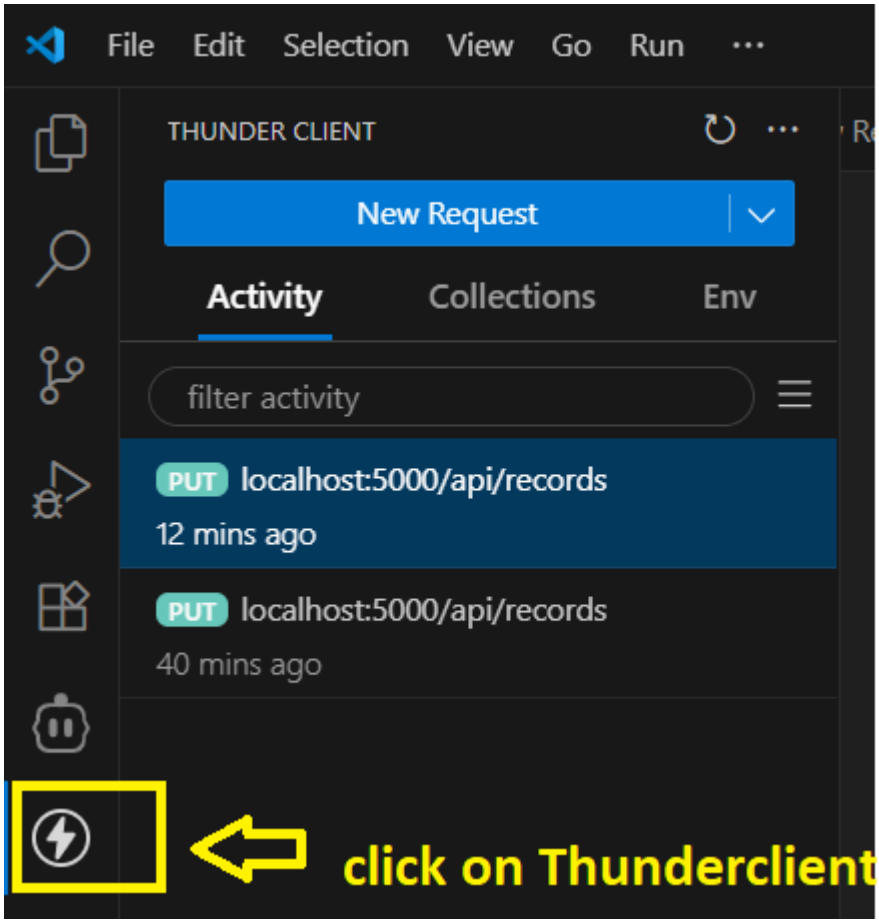
now run it :-

open command prompt and type

node server1.js

```
F:\mern-stack\interview\record-api>node server1.js
🚀 Server running on port 5000
✅ MongoDB connected
```

Now test it using ThunderClient in vs code:-



And click on New Request and

To add record :- select method Post and use url

http://localhost:5000/api/records and body type:-

{

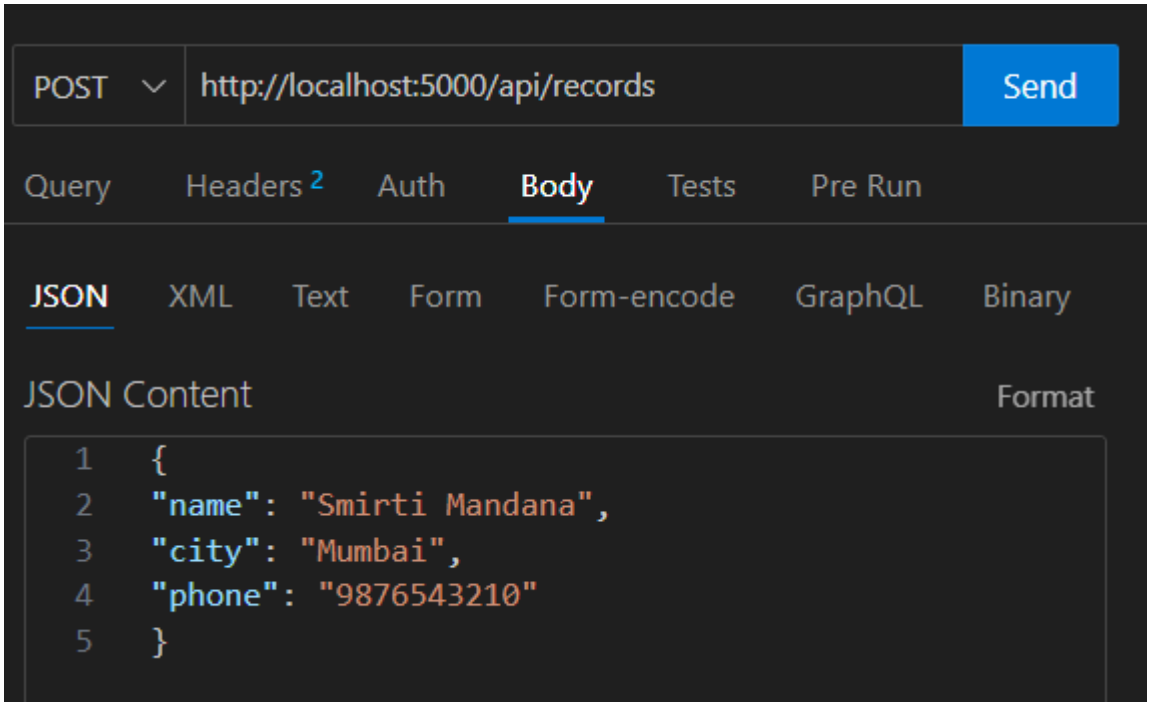
"name": "Smirti Mandana",

"city": "Mumbai",

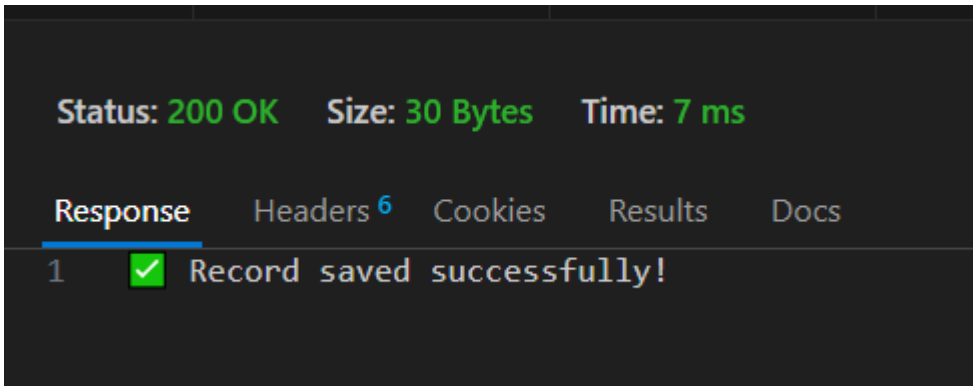
"phone": "9876543210"

}

As shown below :-

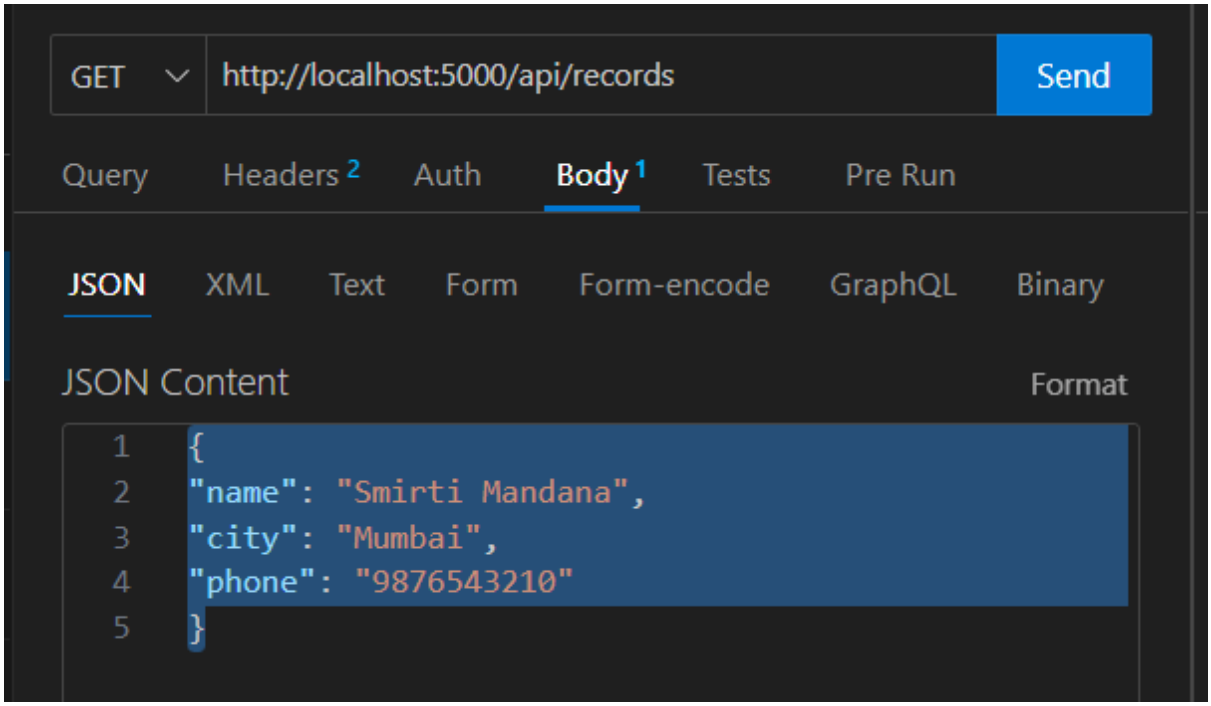


After click on send you will see response:-

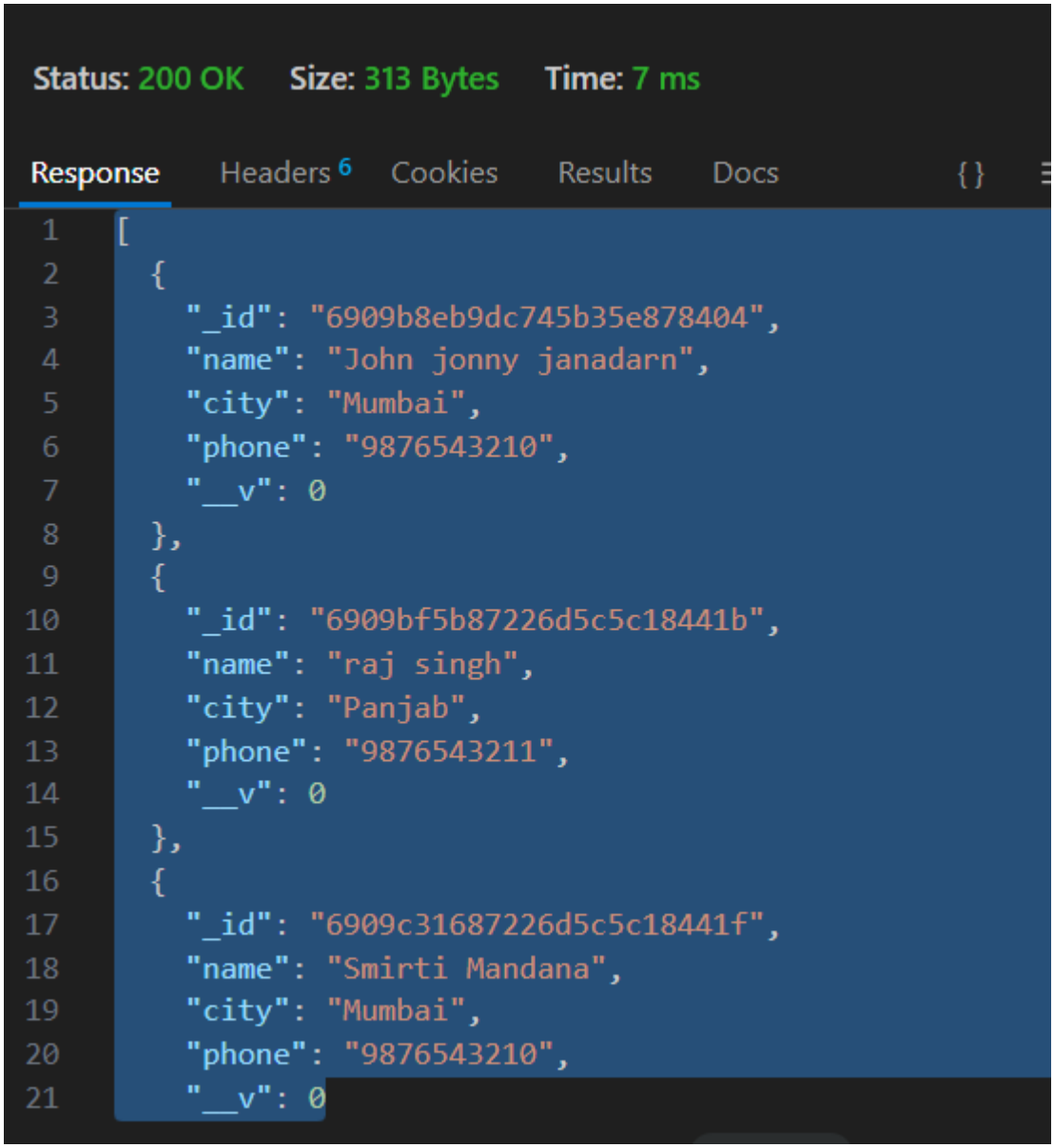


Now to display all records:-

Select GET method and url http://localhost:5000/api/records



After click on send you will see response:-

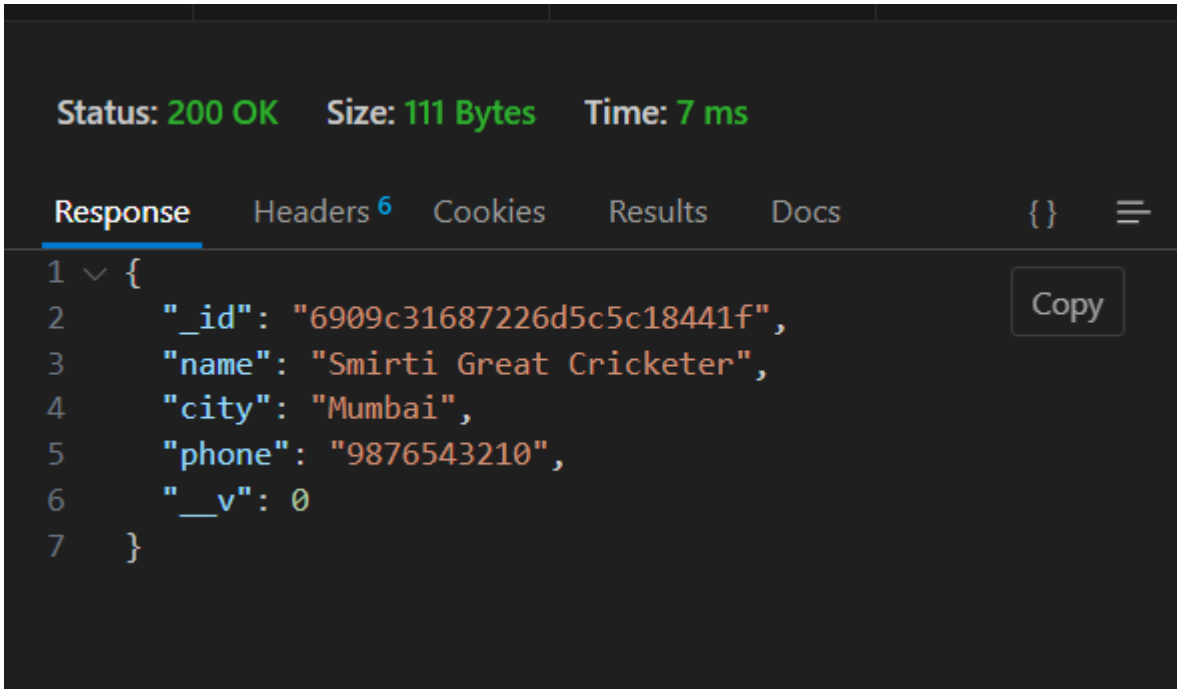


And for update :-

Pass Id in url and select put method as shown below :-

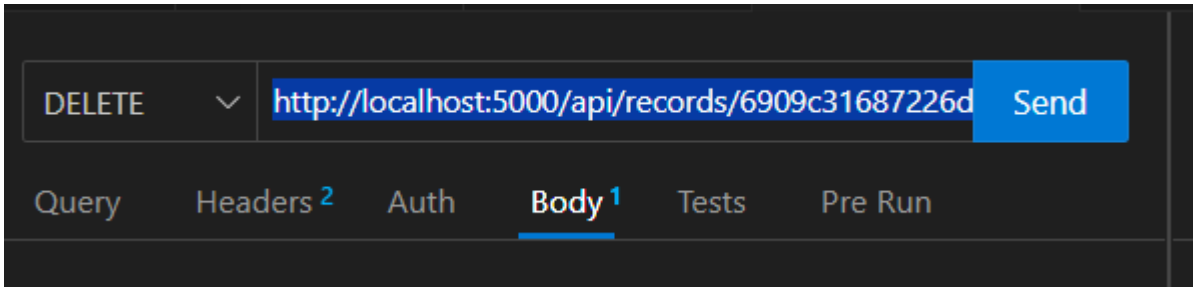
http://localhost:5000/api/records/<id>

In Response you will see updated record:-



➢ Delete Record :- select **DELETE** method and url as shown below with id

```
http://localhost:5000/api/records/<id>
```



Response: `"Record deleted successfully!"`

This is a real, full CRUD API for a "Record" collection using:
□ MongoDB (for storage)
□ Mongoose (for schema + queries)
□ Express (for routes)