CRUD Operations with Express.js and MongoDB

This guide explains how to implement CRUD (Create, Read, Update, Delete) operations using Express.js and MongoDB. CRUD is a fundamental concept for interacting with databases and is essential for most web applications.

Prerequisites

- Basic knowledge of JavaScript, Node.js, and Express.js.
- MongoDB installed locally or access to a MongoDB cloud service (e.g., MongoDB Atlas).
- npm installed.

Project Setup

1. Initialize a Node.js project:

```
mkdir crud-express-mongodb
 cd crud-express-mongodb
 npm init -y
```

1. Install dependencies:

```
npm install express mongoose body-parser cors dotenv
```

1. Create the following folder structure:

```
crud-express-mongodb/
 |-- models/
 |-- routes/
 |-- .env
 |-- server.js
```

Step 1: Setup MongoDB Connection

File: .env

```
MONGO_URI=mongodb://localhost:27017/crudDB
PORT=5000
```

File: server.js

```
const express = require("express");
const mongoose = require("mongoose");
const bodyParser = require("body-parser");
const cors = require("cors");
require("dotenv").config();

const app = express();

app.use(bodyParser.json());
app.use(cors());

const mongoDBUri = 'mongodb://localhost:27017/myDatabase';
async function connectDB() {
 try {
 await mongoose.connect(mongoDBUri);
 console.log('Connected to MongoDB successfully!');
 } catch (err) {
 console.error('Failed to connect to MongoDB:', err);
 process.exit(1);
 }
}
connectDB();


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


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

```
}
```

Step 2: Create a MongoDB Schema

File: models/Item.js

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

const itemSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  description: {
    type: String,
    required: true,
  },
  price: {
    type: Number,
    required: true,
  },
}, { timestamps: true });

module.exports = mongoose.model("Item", itemSchema);
```

Step 3: Implement CRUD Routes

File: routes/itemRoutes.js

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

router.post("/", async (req, res) => {
  try {
    const newItem = new Item(req.body);
    const savedItem = await newItem.save();
    res.status(201).json(savedItem);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});
```

```
// Get all items
router.get("/", async (req, res) => {
  try {
    const items = await Item.find();
    res.status(200).json(items);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// Get an item by ID
router.get("/:id", async (req, res) => {
  try {
    const item = await Item.findById(req.params.id);
    if (!item) return res.status(404).json({ error: "Item not found" });
    res.status(200).json(item);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// Update an item by ID
router.put("/:id", async (req, res) => {
  try {
    const updatedItem = await Item.findByIdAndUpdate(req.params.id, req.body, { new: true
});
    if (!updatedItem) return res.status(404).json({ error: "Item not found" });
    res.status(200).json(updatedItem);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});
```

```
// Delete an item by ID
router.delete("/:id", async (req, res) => {
  try {
    const deletedItem = await Item.findByIdAndDelete(req.params.id);
    if (!deletedItem) return res.status(404).json({ error: "Item not found" });
    res.status(200).json({ message: "Item deleted" });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

module.exports = router;
```

Step 4: Testing the API

Use tools like Postman or cURL to test the following endpoints:

1. **Create an item:**

```
 POST /api/items
  Content-Type: application/json
  {
    "name": "Laptop",
    "description": "A powerful laptop",
    "price": 1200
  }
```

1. **Get all items:**

```
 GET /api/items
```

[redacted]

1. Get an item by ID:

```
 GET /api/items/:id
```

[redacted]

1. **Update an item:**

```
 PUT /api/items/:id
```

```
Content-Type: application/json
{
  "name": "Gaming Laptop",
  "description": "A high-end gaming laptop",
  "price": 1500
}
```

1. **Delete an item:**
   DELETE/api/items/id: