Experiment – 4.1

Name: Akash
UID: 23BCC70039
Subject: Full Stack

Title

CRUD Operations for Product Database Using Mongoose

Objective

- Learn how to implement **Create, Read, Update, and Delete (CRUD)** operations using **MongoDB** with **Mongoose** in Node.js.
- Understand schema design, database connectivity, and structured data handling.
- Apply practical Node.js skills for backend development.

Code Implementation (JavaScript / Node.js)

1️⃣ **server.js**

```
// Import dependencies
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');

// Initialize app
const app = express();
app.use(bodyParser.json());

// MongoDB connection
const MONGO_URL = process.env.MONGO_URL; // From Replit Secret or .env file
```

```
mongoose.connect(MONGO_URL, {
    useNewUrlParser: true,
    useUnifiedTopology: true
})
.then(() => console.log("✅ Connected to MongoDB"))
.catch(err => console.log("❌ MongoDB connection error:", err));

// Product schema
const productSchema = new mongoose.Schema({
    name: { type: String, required: true },
    price: { type: Number, required: true },
    category: { type: String, required: true }
});

const Product = mongoose.model('Product', productSchema);

// CRUD Routes

// Create product
app.post('/products', async (req, res) => {
    try {
        const product = new Product(req.body);
        const savedProduct = await product.save();
        res.status(201).json(savedProduct);
    } catch (err) {
        res.status(400).json({ error: err.message });
    }
});

// Read all products
app.get('/products', async (req, res) => {
    try {
        const products = await Product.find();
        res.json(products);
    } catch (err) {
        res.status(500).json({ error: err.message });
    }
});

// Update product by ID
app.put('/products/:id', async (req, res) => {
    try {
        const updatedProduct = await Product.findByIdAndUpdate(
            req.params.id,
```

```
        req.body,
        { new: true, runValidators: true }
    );
    res.json(updatedProduct);
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
});

// Delete product by ID
app.delete('/products/:id', async (req, res) => {
  try {
    await Product.findByIdAndDelete(req.params.id);
    res.json({ message: "✅ Deleted successfully" });
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
});

// Start server
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => console.log(`🚀 Server running on port ${PORT}`));
```

2️⃣ **package.json**

```
{
  "name": "productsapp",
  "version": "1.0.0",
  "description": "CRUD operations for product database using Mongoose",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "^4.18.2",
    "mongoose": "^7.3.4",
    "body-parser": "^1.20.2"
  }
}
```

Working Procedure

1. **Setup MongoDB Atlas**
    - **Create a cluster and database named .**
    - **Add a database user with a readWrite role.**
    - **Add your IP to the allowed IP list.**
2. **Connect Node.js app**
3. **Run Server**
4. **Server connects to MongoDB and listens on port 3000.**
5. **Test CRUD Operations .**

server.js | package.json | Secrets | Shell | Publishing | +

server.js > ...

```javascript
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');

const app = express();

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

// MongoDB connection (from Replit Secret)
const MONGO_URL = process.env.MONGO_URL;
if (!MONGO_URL) {
  console.error('❌ Missing MONGO_URL environment variable.');
  process.exit(1);
}

mongoose.connect(MONGO_URL, {
  useNewUrlParser: true,
  useUnifiedTopology: true
})
  .then(() => console.log('✅ Connected to MongoDB Atlas'))
  .catch(err => {
    console.error('❌ MongoDB connection error:', err.message);
    process.exit(1);
  });
```

AKASH'S ORG - 2025-09-09 > PRODUCTSAPP

Database Access

Database Users | Custom Roles

User	Description	Authentication Method
AK		SCRAM