EXP-4.1:

AIM:

Create a Node.js application that connects to a MongoDB database using Mongoose. Define a Product model with properties such as name, price, and category. Implement routes or functions to perform CRUD operations: add a new product, retrieve all products, update a product by its ID, and delete a product by its ID. Use appropriate Mongoose methods for each operation and ensure that all data validations and error handling are included.

CODE-

1. Initialize Project and Install Packages:

Start a new directory and set up a Node.js project:

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

2. Connect to MongoDB Using Mongoose:

Create [app.js]:

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const app = express();
app.use(bodyParser.json());
mongoose.connect('mongodb://localhost:27017/productsdb', {
  useNewUrlParser: true,
  useUnifiedTopology: true
})
.then(() => console.log('MongoDB connected'))
.catch(err => console.error(err));
```

3. Define the Product Model:

Create models/Product.js:

```
const mongoose = require('mongoose');
const productSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
    trim: true
  },
  price: {
    type: Number,
    required: true,
    min: 0
  },
  category: {
    type: String,
    required: true,
    trim: true
  }
});

module.exports = mongoose.model('Product', productSchema);
```

4. Implement CRUD Routes:

In [app.js]:

```
 const Product = require('./models/Product');

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

// Read
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
app.put('/products/:id', async (req, res) => {
   try {
      const product = await Product.findByIdAndUpdate(
         req.params.id, req.body, { new: true, runValidators: true }
      );
      if (!product) return res.status(404).json({ error: 'Product not found' });
      res.json(product);
```

```
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
});

// Delete
app.delete('/products/:id', async (req, res) => {
  try {
    const product = await Product.findByIdAndDelete(req.params.id);
    if (!product) return res.status(404).json({ error: 'Product not found' });
    res.json({ message: 'Product deleted' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.listen(3000, () => console.log('Server running on port 3000'));
```

<u>OUTPUTS:</u>

1)Create Product (POST /products):

Request Body:

```
{
  "name": "Laptop",
  "price": 1200,
  "category": "Electronics"
}
{
  "_id": "650f1c8fbcfbd4021f895a6b",
  "name": "Laptop",
  "price": 1200,
  "category": "Electronics",
  "__v": 0
}
```

2)Read Products (GET /products):

Request Body:

```
[
  {
    "_id": "650f1c8fbcfbd4021f895a6b",
    "name": "Laptop",
    "price": 1200,
    "category": "Electronics",
    "__v": 0
  },
  {
    "_id": "650f1c9ebcfbd4021f895a6c",
    "name": "Desk Chair",
    "price": 150,
    "category": "Furniture",
    "__v": 0
  }
]
```

3)Update Product (PUT /products/:id)

Request Body:

```
{
  "price": 1250
}
```

```
{
  "_id": "650f1c8fbcfbd4021f895a6b",
  "name": "Laptop",
  "price": 1250,
  "category": "Electronics",
  "__v": 0
}
```

4)Delete Product (DELETE /products/:id):

Sample Response:

```
{
  "message": "Product deleted"
}
```