Practical 12.

WAP to CRUD Operation in NodeJS with MongoDB.

server.js

```
const express = require("express");
const mongoose = require("mongoose");
const app = express();
const PORT = 3100;

app.use(express.json());

const MONGODB_URI = "Your Connection string";

mongoose
  .connect(MONGODB_URI)
  .then(() => console.log("Connected to MongoDB"))
  .catch((err) => console.error("Could not connect to MongoDB:", err));

const itemSchema = new mongoose.Schema({
  name: { type: String, required: true },
  description: String,
  createdAt: { type: Date, default: Date.now },
});
```

```
const Item = mongoose.model("Item",
                itemSchema);
app.post("/items", async(req, res)
  => {
    try {
      const newItem = new Item(req.body);
      const savedItem = await
                newItem.save();
      res.status(201).json(savedItem);
}
catch(error) {
  res.status(400).json({ message
      : error.message });
}
});
// get Single Id.
app.get("/items", async(req, res)
  => {
    try {
      const items = await Item.find()
      res.json(items);
  } catch(error) {
  res.status(500).json({ message:
          error.message });
} });
```

Scanned with OKEN Scanner

```
// Update Item by ID
app.put("/items/:id", async (req, res)
    => {

  try {
    const updatedItem
    = await Item.findByIdAndUpdate
      (req.params.id, req.body,
  {
    new: true,
  });
  if(!updatedItem){
    return res.status(404).json(
      {message: "Item not found."});
  }
  res.json(updatedItem);
  }
  catch (error) {
    res.status(400).json({message:
      error.message});
  }
});

// Delete Item by ID
```

```
app.delete("/items/:id", async (req, res)
  => {
  try {
    const deletedItem = await
      Item.findByIdAndDelete(req.params.
                             id);
    if (!deletedItem) {
      return res.status(404).json
        ({ message: "Item not found" });
    }
    res.json({ message: "Item deleted
                        successfully" });
  } catch (error) {
    res.status(500).json({ message:
      error.message });
  }
});

app.listen(PORT, () => {
  console.log('Server is running on
    http://localhost:${PORT}');
});
```

Scanned with OKEN Scanner

Output

1. Create

POST :- http://localhost:3100/items
body -> raw -> json

```
{
  "name": "Item 1",
  "description": "Iphone 16 pro max"
}
```

Output

```
{
  "name": "Item 1",
  "description": "Iphone 16 pro max",
  "_id": "68065f7d61325311fee085f4",
  "createdAt": "2025-04-21T14:13:17.284Z",
  "__v": 0
}
```

2. Read

GET :- http://localhost:3100/items

Output:

```
[
  {
    "_id": "68065f7d61325311fee085f4",
```

Scanned with OKEN Scanner

```
"name": "Item 1",
"description": "Iphone 16 pro max",
"createdAt": "2025-04-21T14:13:19.284Z",
"__v": 0
}
{
"_id": "6806f3e6132531fee085e5",
"name": "Item 2",
"description": "This is a description
                for Item 2",
"createdAt": "2025-04-21T13:59:26
             .331Z",
"__v": 0
},
```

3. Update

PUT :- http://localhost:3100/items/6806527d6132531fee085f4

body → raw → json

```
{
"name": "Mobile 1",
"description": "Iphone 16 pro max"
}
```

Output :-

```
{
"_id": "6806527d6132531fee085f4",
"name": "Mobile 1",
"description": "Iphone 16 pro max",
"createdAt": "2025-04-21T14:13:17.284Z",
"__v": 0
}
```

Scanned with OKEN Scanner

4. Delete

DELETE :- http://localhost:3100/items/68064f3e613253111fee085e5

Output :-

```
{
  "message": "Item deleted successfully"
}
```

Scanned with OKEN Scanner