Worksheet No.: 5

Student Name: Harsh
UID: 24MCA20045
Branch: MCA (General)
Section/Group: 1-A
Semester: 2nd
Date of Performance:31/03/2025
Subject Name: Advanced internet programming lab
Subject Code: 24CAP-652

1. Aim of the practical:

Implement CRUD operation with database on NodeJS with MongoDB/Postman.

2. Apparatus:

Visual Studio code (Vs code), Mongodb, Postman.

3. Code for experiment/practical:

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const cors = require('cors');

const app = express();
const PORT = 5000;

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

// MongoDB Connection
mongoose.connect("mongodb://127.0.0.1:27017/crudDB")
    .then(() => console.log("Connected to MongoDB"))
    .catch((err) => console.error("MongoDB connection error:", err));
```

```
// Schema and Model
const ItemSchema = new mongoose.Schema({
    name: String,
    email: String
});

const Item = mongoose.model('Item', ItemSchema);

// Routes

// Create (POST)
app.post('/items', async (req, res) => {
    try {
        const newItem = new Item(req.body);
        await newItem.save();
        res.status(201).json(newItem);
    } catch (err) {
        res.status(500).json({ message: err.message });
    }
});

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

// Create multiple items (Bulk Insert)
app.post('/items/bulk', async (req, res) => {
    try {
        const newItems = await Item.insertMany(req.body); // Insert an array of items
        res.status(201).json(newItems);
    } catch (err) {
        res.status(500).json({ message: err.message });
    }
});
```

```
// Update (PUT)
app.put('/items/:id', async (req, res) => {
   try {
      const updatedItem = await Item.findByIdAndUpdate(req.params.id, req.body, { new: true
});
      res.json(updatedItem);
   } catch (err) {
      res.status(500).json({ message: err.message });
   }
});

// Delete (DELETE)
app.delete('/items/:id', async (req, res) => {
   try {
      await Item.findByIdAndDelete(req.params.id);
      res.json({ message: 'Item deleted' });
   } catch (err) {
      res.status(500).json({ message: err.message });
   }
});

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

4. Result/Output/Writing Summary:

```
PS D:\nodejs> node mongo.js
  >>
  Server running on http://localhost:5000
  Connected to MongoDB
```

New Collection / http://localhost:5000/items/bulk

Save Share

POST http://localhost:5000/items/ Send

Params Authorization Headers (8) Body Scripts Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
[
  {
    "name": "Harsh",
    "email": "harshjangid@gmail.com"
  }
]
```

Body Cookies Headers (8) Test Results 201 Created 100 ms 314 B Save Response

JSON Preview Visualize

```
{
    "_id": "67eae3987f1e7817074bd867",
    "__v": 0
}
```

New Collection / http://localhost:5000/items/bulk

Save Share

GET http://localhost:5000/items/ Send

Params Authorization Headers (8) Body Scripts Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
[
  {
    "name": "Harsh",
    "email": "harshjangid@gmail.com"
  }
]
```

Body Cookies Headers (8) Test Results 200 OK 18 ms 634 B Save Response

JSON Preview Visualize

```
[
    {
        "_id": "67eabc3f35740663277d5afd",
        "__v": 0
    },
    {
        "_id": "67eabc7235740663277d5b01",
        "name": "khsdka",
        "email": "laskndlaska090@gmail.com",
        "__v": 0
    },
```

DELETE http://localhost:5000/items/67eabc7235740663277d5b01 Send

Params Authorization Headers (8) Body Scripts Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
[

  {
    "name": "Harsh",
    "email": "harshjangid@gmail.com"
  }
]
```

Body Cookies Headers (8) Test Results 200 OK 23 ms 293 B Save Response

JSON Preview Visualize

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

Documents 4 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or Generate query Explain Reset Find Options

ADD DATA EXPORT DATA UPDATE DELETE 25 1 - 4 of 4

```
_id: ObjectId('67eabc3f35740663277d5afd')
__v : 0

_id: ObjectId('67eabc7235740663277d5b01')
name : "khsdka"
email : "laskndlaska090@gmail.com"
__v : 0

_id: ObjectId('67eabfde9beebb1fdd53e650')
name : "Harsh"
email : "harshjangid@gmail.com"
__v : 0

_id: ObjectId('67eacb5556afe97447e5d599')
name : "khsdka"
email : "laskndlaska090@gmail.com"
__v : 0
```

5. Learning outcomes (What I have learnt):

1. Understanding CRUD: Learn to perform Create, Read, Update, and Delete operations in Node.js with MongoDB.
2. Setup Node.js & MongoDB: Configure a Node.js server with Express and connect it to MongoDB using Mongoose.
3. Build RESTful APIs: Implement APIs for CRUD operations and test them using Postman.
4. Handle API Requests: Work with HTTP methods (GET, POST, PUT, DELETE) and manage request data.
5. Validate & Manage Data: Use Mongoose schemas for data validation and error handling in MongoB.

5. Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.	Worksheet		8 Marks
2.	Viva		10 Marks
3.	Simulation		12 Marks
	Total		30 Marks

Teacher Signature