Backend Development Lab

CSFS3101P_1

Submitted By: Sahil Naranag

Course: B.Tech. C.S.E.

Batch: 2

Sap.id.: 500119480

Enrollment no.: R2142230356

Submitted to:

Satyam Tiwari

Index

GitHub link - https://github.com/Sahil390/Backend

Experiment - 1

Write code for the below questions:

1. Exporting an object from a module using exports Object.

exp1 > JS p1.js > ...

```
const object = {
    name : "Sahil Narang",
    age : 21,

    greet: function() {
        return `Hello from ${this.name}!`;
    },
}

module.exports = object;
```

2. Exporting nested objects and function from Module using exports Object.

exp1 > JS p2.js > ...

```
const object = {
    neobject : {
        sapid : 500119480.
    },
    isStudent: function() {
        return 'Yes He is student';
    },
}

module.exports = object;
```

To test p1 and p2 :

exp1 > JS test.js > ...

```
1
2   const io1 = require('./p1.js');
3   const io2 = require('./p2.js');
4
5   console.log('program1 :', io1.greet());
6   console.log('sap id : ', io2.neobject.sapid);
7   console.log('Is Student: ', io2.isStudent());
8
```

Output :

```
sahil@Algon:~/Developer/Collage/Backend/exp1> node test.js
program1 : Hello from Sahil Narang!
sap id :  500119480
Is Student:  Yes He is student
sahil@Algon:~/Developer/Collage/Backend/exp1>
```

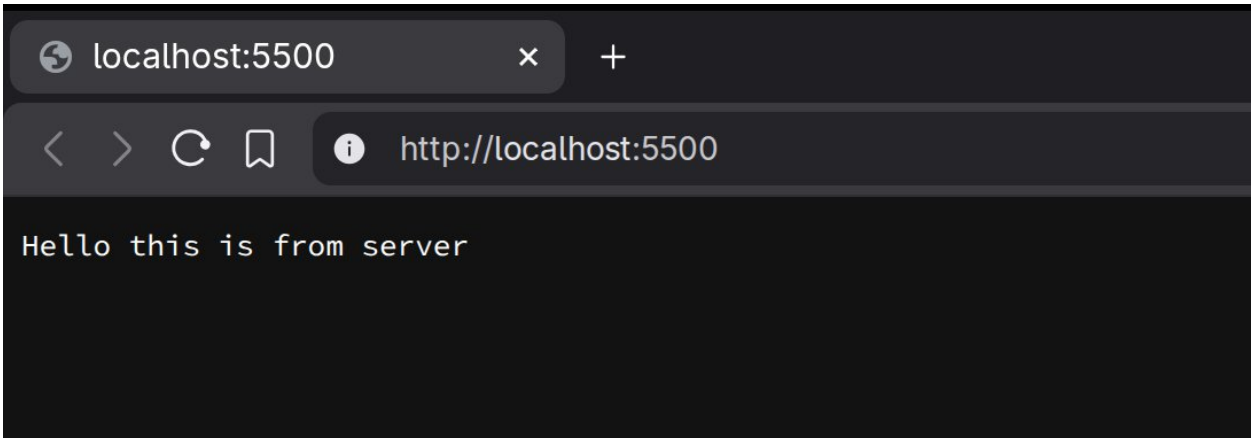
3. Writing to the server using request-response statements as a callback in createServer() function.

exp1 > JS p3.js > ...

```
1   const http = require('http');
2
3   const server = http.createServer(function(req,res) {
4       res.write("Hello this is from server");
5       res.end();
6   })
7
8   server.listen(5500);
```

Output :

Now if we run “p3.js” and then open the localhost port on browser we get output



4. Reading into a file asynchronously and writing code for handling error if file not found to read.

```
exp1 > JS p4.js > ...
1   const fs = require('fs');
2
3   fs.readFile('tfile.txt','utf8', function(err,data){
4       if(err) {
5           console.error("Error ",err.message);
6           return;
7       }
8
9       console.log(data);
10  });
```

Here is the sample text file i used

```
exp1 > tfile.txt
1   Hello my name is sahil narang
2   and this is sample text file
```

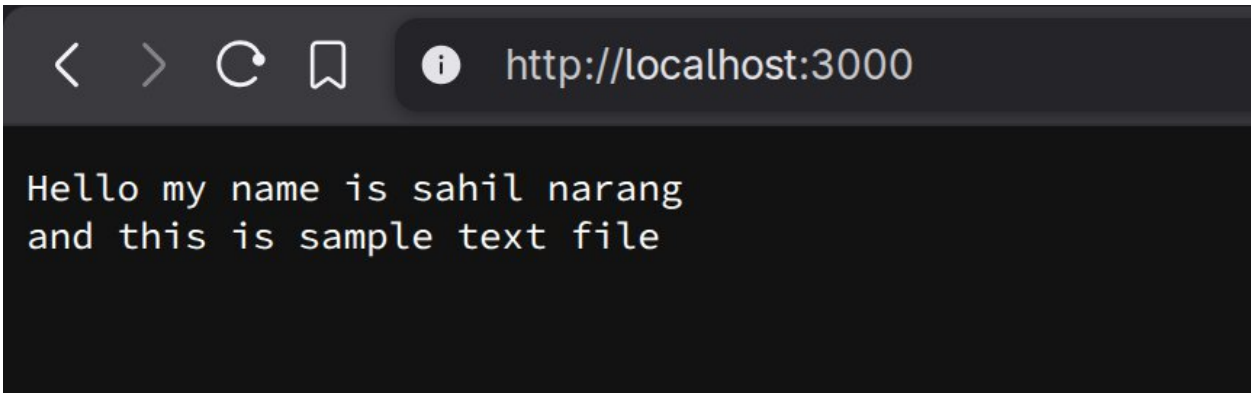
Output :

```
sahil@Algon:~/Developer/Collage/Backend/exp1> node p4.js
Hello my name is sahil narang
and this is sample text file
sahil@Algon:~/Developer/Collage/Backend/exp1>
```

5. Reading a text file on the server using http and fs module.

```
exp1 > JS p5.js > ...
1   const fs = require('fs');
2   const http = require('http');
3
4   const server = http.createServer(function(req,res) {
5       fs.readFile('./tfile.txt','utf8', function(err,data){
6           if(err) {
7               res.write("File not found or error reading file.");
8               return;
9           }
10          res.write(data);
11          res.end();
12      });
13  });
14  server.listen(3000);
```

Now same as 3rd program when we open the port 3000 on browser we get output :



Experiment - 2

Write code for given questions:

1. Write a program that uses a Readable stream to read data from a file (data.txt). Output the file content to the console. Ensure the file exists before reading, and handle any errors if the file is missing.

exp2 > JS p1.js > ...

```
1  const fs = require('fs');
2
3  let co = '';
4  const readStream = fs.createReadStream('data.txt',{encoding: 'utf-8'});
5  readStream.on('data',(chunk) => {
6      co += chunk;
7  });
8
9  readStream.on('error', (err) => {
10   console.error('An error occurred:', err);
11 });
12 readStream.on('end', () => {
13     console.log(co);
14 });
15
```

Sample text file :

exp2 > data.txt

```
1   Hi My name is sahil
```

Output :

```
sahil@Algon:~/Developer/Collage/Backend/exp2> node p1.js
Hi My name is sahil
sahil@Algon:~/Developer/Collage/Backend/exp2>
```

2. Write a program that creates a Writable stream to write a string ("Hello, Node.js!") to a file (output.txt). If the file already exists, overwrite it. Print a success message once the data is written.

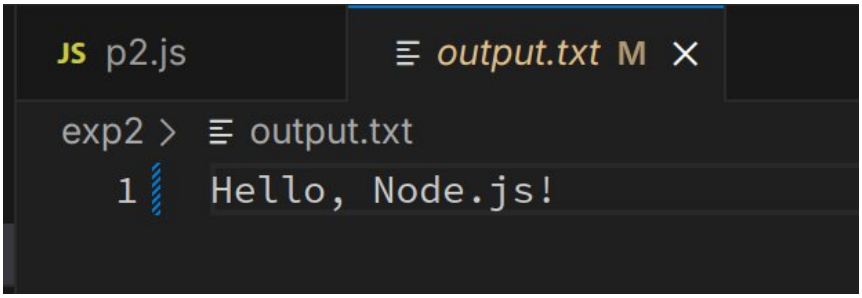
exp2 > JS p2.js > ...

```
1  const fs = require('fs');
2  const writer = fs.createWriteStream('output.txt');
3
4  writer.write("Hello, Node.js!");
5  writer.end();
6
7  writer.on('finish', () =>
8  {
9      console.log("success!");
10 });
```

Output :

```
sahil@Algon:~/Developer/Collage/Backend/exp2> node p2.js
success!
sahil@Algon:~/Developer/Collage/Backend/exp2>
```

Text file “output.txt” :-



3. Write a program that demonstrates stream piping. Use a Readable stream to read data from a file (input.txt), and pipe it to a Writable stream that writes to another file (output.txt).

4. Write a program that uses process.stdin to read user input from the command line. Ask the user for their name and greet them by printing "Hello, [name]!" to the console.

Output:

```
sahil@Algon:~/Developer/Collage/Backend/exp2> node p4.js
Sahil
Hello, Sahil
```

5. Write a program that demonstrates error handling in streams. Create a Readable stream that tries to read from a non-existent file and handles the error by emitting an error event.

exp2 > JS p5.js > ...

```
1  const fs = require('fs');
2
3  const readStream = fs.createReadStream('data2.txt',{encoding: 'utf-8'});
4  readStream.on('error', (err) => {
5    console.error('An error occurred:', err);
6  });
```

Output:

Here “data2.txt” file not exist in my device

```
sahil@Algon:~/Developer/Collage/Backend/exp2> node p5.js
An error occurred: [Error: ENOENT: no such file or directory, open 'data2.txt'] {
  errno: -2,
  code: 'ENOENT',
  syscall: 'open',
  path: 'data2.txt'
}
sahil@Algon:~/Developer/Collage/Backend/exp2>
```

Experiment - 3

Objective - Understand the basics of backend development.

Lab Activities:

- Overview of HTTP methods (GET, POST, PUT, DELETE).
- Introduction to REST architecture.
- Building a simple RESTful API.

Sample Project: Create an API for managing a list of items (e.g., a to-do list) using HTTP modules and express framework.

Server.js :-

```
onst http = require('http');

let items = [
  { id: 1, name: 'Item 1', description: 'This is the first item.' },
  { id: 2, name: 'Item 2', description: 'This is the second item.' },
  { id: 3, name: 'Item 3', description: 'This is the third item.' }
];
let nextId = 4;

const server = http.createServer((req, res) => {
  res.setHeader('Access-Control-Allow-Origin', '*');
  res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, OPTIONS');
  res.setHeader('Access-Control-Allow-Headers', 'Content-Type');

  if (req.method === 'OPTIONS') {
    res.writeHead(204);
    res.end();
    return;
  }

  res.setHeader('Content-Type', 'application/json');

  const { method, url } = req;
  if (method === 'GET' && url === '/items') {
    res.writeHead(200);
    res.end(JSON.stringify(items));
  }

  else if (method === 'GET' && url.match(/\/items\/([0-9]+)/)) {
    const id = parseInt(url.split('/')[2]);
    const item = items.find(i => i.id === id);

    if (item) {
      res.writeHead(200);
      res.end(JSON.stringify(item));
    } else {
      res.writeHead(404);
      res.end(JSON.stringify({ message: 'Item not found' }));
    }
  }

  else if (method === 'POST' && url === '/items') {
    let body = '';
```

```
  req.on('data', chunk => {
    body += chunk.toString();
  });
  req.on('end', () => {
    const { name, description } = JSON.parse(body);
    const newItem = { id: nextId++, name, description };
    items.push(newItem);
    res.writeHead(201); // 201 Created
    res.end(JSON.stringify(newItem));
  });
}
else if (method === 'PUT' && url.match(/\/items\/([0-9]+)/)) {
  const id = parseInt(url.split('/')[2]);
  let body = '';
  req.on('data', chunk => {
    body += chunk.toString();
  });
  req.on('end', () => {
    const { name, description } = JSON.parse(body);
    const itemIndex = items.findIndex(i => i.id === id);

    if (itemIndex !== -1) {
      const updatedItem = { id, name, description };
      items[itemIndex] = updatedItem;
      res.writeHead(200);
      res.end(JSON.stringify(updatedItem));
    } else {
      res.writeHead(404);
      res.end(JSON.stringify({ message: 'Item not found' }));
    }
  });
}

else if (method === 'DELETE' && url.match(/\/items\/([0-9]+)/)) {
  const id = parseInt(url.split('/')[2]);
  const itemIndex = items.findIndex(i => i.id === id);

  if (itemIndex !== -1) {
    items.splice(itemIndex, 1);
    res.writeHead(204);
    res.end();
  } else {
    res.writeHead(404);
    res.end(JSON.stringify({ message: 'Item not found' }));
  }
```

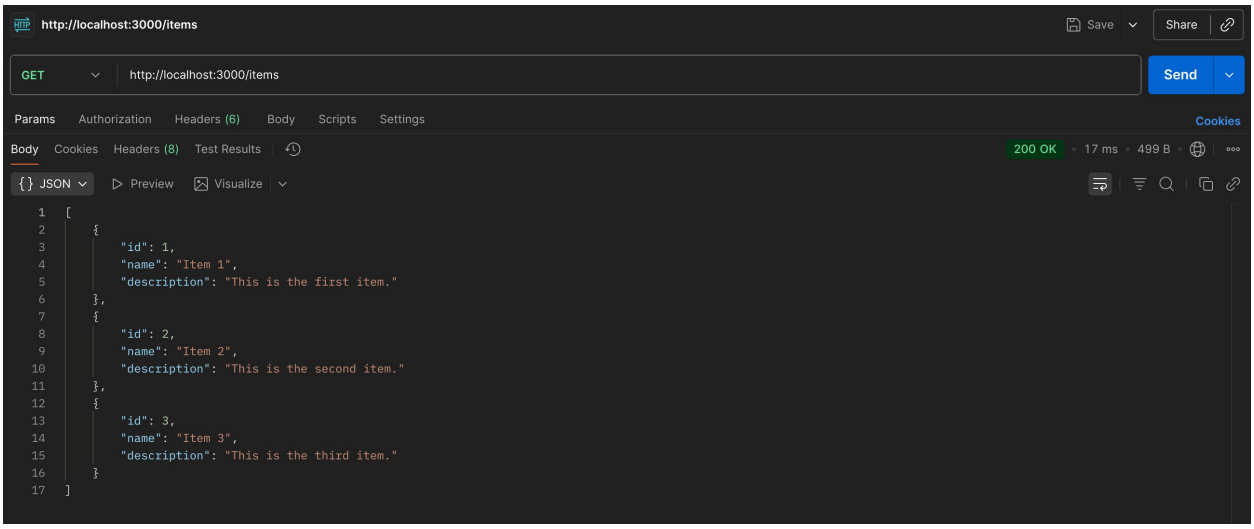
```
  }

  else {
    res.writeHead(404);
    res.end(JSON.stringify({ message: 'Route not found' }));
  }
});

const port = 3000;
server.listen(port, () => {
  console.log(`Server is running on http://localhost:${port} using the native HTTP module.`);
});
```

Output:

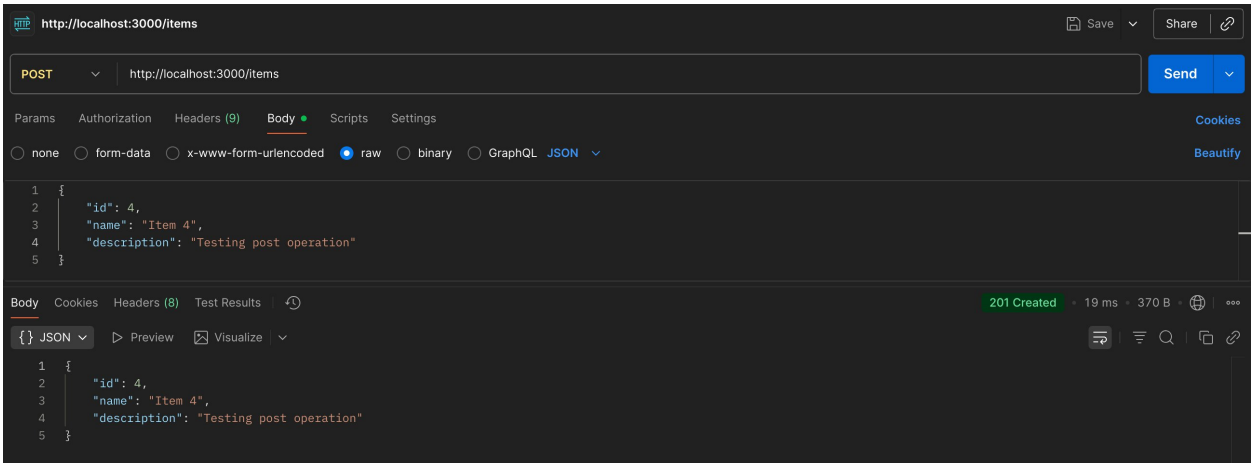
```
sahil@Algon:~/Developer/Collage/Backend/exp3/crud_http> node server.js
Server is running on http://localhost:3000 using the native HTTP module.
```

Postman screenshot for crud opeationes :

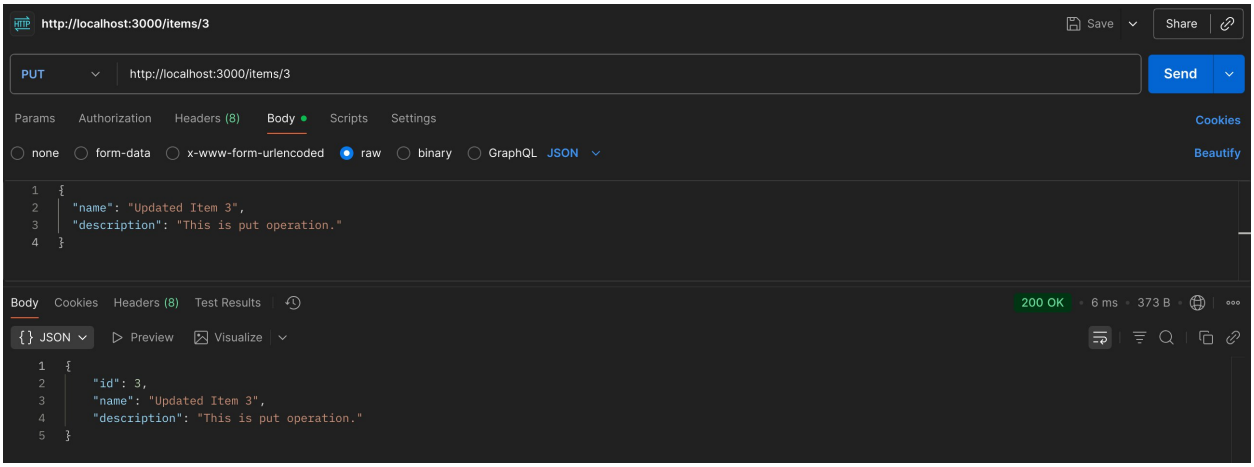
Get :



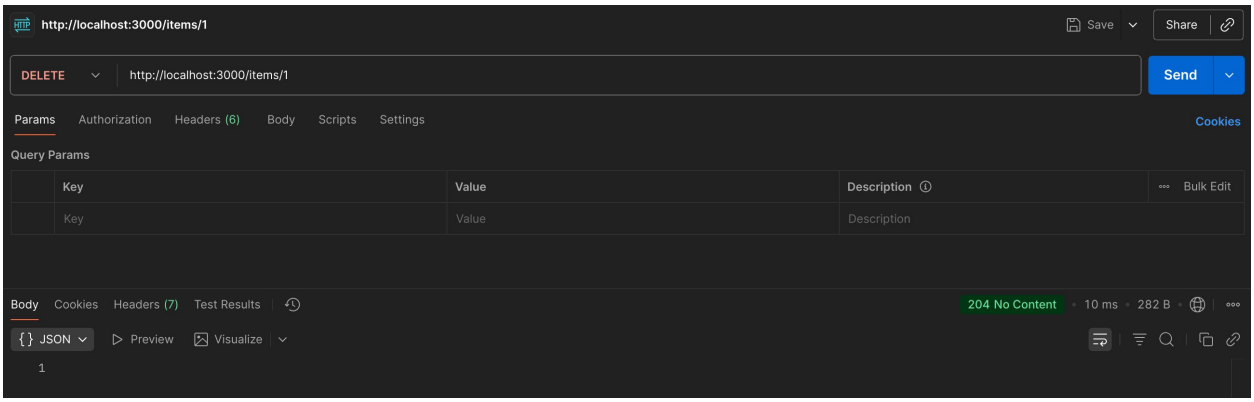
POST:



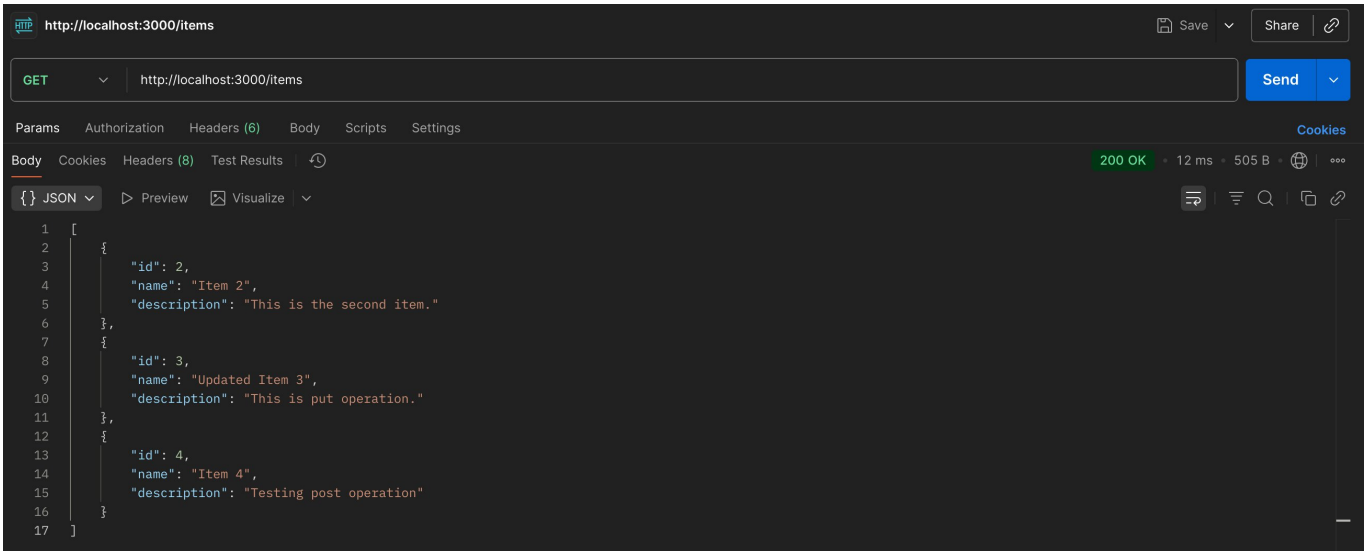
PUT:



DELETE:



Get Operation after changes :



Experiment - 4

Objective: Integrate databases with backend applications.

Lab Activities:

- Introduction to relational a Performing basic CRUD operations.
- and non-relational databases (MongoDB).
- Setting up and connecting to a database.

Sample Project: Extend the to-do list API to store items in a database and also create repository based CRUD operation.

Connect.js :-

```
JS connect.js > ...
1   require('dotenv').config();
2   const mongoose = require('mongoose');
3
4   async function test() {
5     try {
6       await mongoose.connect(process.env.MONGODB_URI, {
7         useNewUrlParser: true,
8         useUnifiedTopology: true,
9       });
10      console.log('Connected to MongoDB Atlas');
11      await mongoose.disconnect();
12      process.exit(0);
13    } catch (err) {
14      console.error('Connection error', err);
15      process.exit(1);
16    }
17  }
18  test();
```

This code is used to test the connection with mongoose where MONGODB_URI is defined in “.env” file.

Output :

```
sahil@Algon:~/Developer/Collage/Backend/exp4> node connect.js
[dotenv@17.2.2] injecting env (2) from .env -- tip: ⚙ write to custom o
v: myObject }
(node:17327) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated o
 has no effect since Node.js Driver version 4.0.0 and will be removed in
n
(Use `node22 --trace-warnings ...` to show where the warning was created
(node:17327) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecate
pology has no effect since Node.js Driver version 4.0.0 and will be remo
version
Connected to MongoDB Atlas
sahil@Algon:~/Developer/Collage/Backend/exp4>
```

Server.js :-

This is the main file :-

```
const http = require('http');
require('dotenv').config();
const mongoose = require('mongoose');
const repo = require('./repositories/itemRepository');

const start = async () => {
  try {
    await mongoose.connect(process.env.MONGODB_URI);
    console.log('Connected to MongoDB');

    const server = http.createServer(async (req, res) => {
      res.setHeader('Access-Control-Allow-Origin', '*');
      res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, OPTIONS');
      res.setHeader('Access-Control-Allow-Headers', 'Content-Type');

      if (req.method === 'OPTIONS') {
        res.writeHead(204);
        res.end();
        return;
      }

      res.setHeader('Content-Type', 'application/json');

      const { method, url } = req;

      try {
        // GET /items
        if (method === 'GET' && url === '/items') {
          const items = await repo.findAll();
          res.writeHead(200);
          res.end(JSON.stringify(items));
          return;
        }

        // GET /items/:id
        if (method === 'GET' && url.match(/\/items\/[0-9]+/)) {
          const id = url.split('/')[2];
          const item = await repo.findById(id);
          if (item) {
            res.writeHead(200);
            res.end(JSON.stringify(item));
```

```
    } else {
      res.writeHead(404);
      res.end(JSON.stringify({ message: 'Item not found' }));
    }
    return;
  }

  // POST /items
  if (method === 'POST' && url === '/items') {
    let body = '';
    req.on('data', chunk => body += chunk.toString());
    req.on('end', async () => {
      const { name, description } = JSON.parse(body || '{}');
      if (!name) {
        res.writeHead(400);
        res.end(JSON.stringify({ message: 'Name is required' }));
        return;
      }
      const newItem = await repo.create({ name, description });
      res.writeHead(201);
      res.end(JSON.stringify(newItem));
    });
    return;
  }

  if (method === 'PUT' && url.match(/\/items\/[0-9]+/)) {
    const id = url.split('/')[2];
    let body = '';
    req.on('data', chunk => body += chunk.toString());
    req.on('end', async () => {
      const { name, description } = JSON.parse(body || '{}');
      const updated = await repo.update(id, { name, description });
      if (updated) {
        res.writeHead(200);
        res.end(JSON.stringify(updated));
      } else {
        res.writeHead(404);
        res.end(JSON.stringify({ message: 'Item not found' }));
      }
    });
    return;
  }

  // DELETE /items/:id
  if (method === 'DELETE' && url.match(/\/items\/[0-9]+/)) {
```

```
        const id = url.split('/')[2];
        const ok = await repo.delete(id);
        if (ok) {
          res.writeHead(204);
          res.end();
        } else {
          res.writeHead(404);
          res.end(JSON.stringify({ message: 'Item not found' }));
        }
        return;
      }

      // Not found
      res.writeHead(404);
      res.end(JSON.stringify({ message: 'Route not found' }));
    } catch (err) {
      console.error('Request handler error', err);
      res.writeHead(500);
      res.end(JSON.stringify({ message: 'Internal server error' }));
    }
  });

  const port = process.env.PORT || 3000;
  server.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
  });

 } catch (err) {
  console.error('Failed to start server', err);
  process.exit(1);
 }
};

start();
```

We are importing “itemRepository.js” in which i have defined all the MongoDB Operationes

“itemRepository.js” :-

```
const Item = require('../models/item');
const mongoose = require('mongoose');
const counterSchema = new mongoose.Schema({ _id: String, seq: { type: Number, default:
0 } });
const Counter = mongoose.models.Counter || mongoose.model('Counter', counterSchema);

async function getNextSequence(name) {
  const ret = await Counter.findByIdAndUpdate(
    name,
    { $inc: { seq: 1 } },
    { new: true, upsert: true }
  ).lean();
  return ret.seq;
}

module.exports = {

  async create(data) {

    const nextId = await getNextSequence('items');
    const doc = await Item.create({ id: nextId, ...data });
  return { id: doc.id, name: doc.name, description: doc.description };
  },

  async findAll(options = {}) {
    const { filter = {}, page = 1, limit = 50, sort = { createdAt: -1 } } = options;
    const skip = Math.max(0, page - 1) * limit;
    const docs = await Item.find(filter).sort(sort).skip(skip).limit(limit).lean();
  return docs.map(d => ({ id: d.id, name: d.name, description: d.description }));
  },

  async findById(id) {
  const doc = await Item.findOne({ id: Number(id) }).lean();
  if (!doc) return null;
  return { id: doc.id, name: doc.name, description: doc.description };
  },

  async update(id, changes) {
  const updated = await Item.findOneAndUpdate({ id: Number(id) }, changes, { new: true,
runValidators: true }).lean();
  if (!updated) return null;
  return { id: updated.id, name: updated.name, description: updated.description };
  },
```

```
  async delete(id) {
    const res = await Item.findOneAndDelete({ id: Number(id) }).lean();
    return !!res;
  },

  async count(filter = {}) {
    return Item.countDocuments(filter);
  }
};
```

We are importing item.js file in it. In this i have defined the Mongoose shema model for an Item document stored in MongoDB.

“item.js” :-

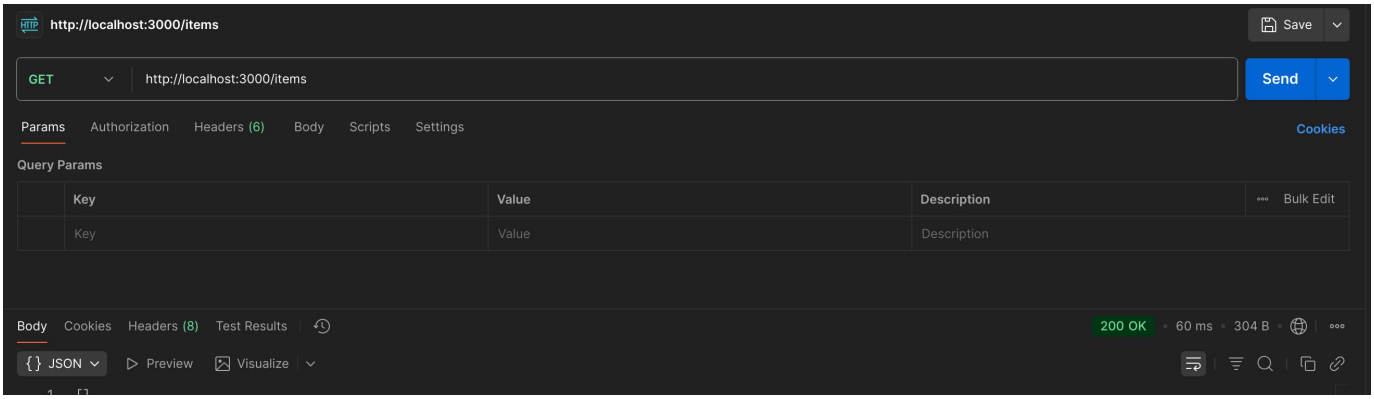
```
models > JS item.js > ...
1   // models/item.js
2   const { Schema, model } = require('mongoose');
3
4
5   const itemSchema = new Schema({
6     id: { type: Number, required: true, unique: true, index: true },
7     name: { type: String, required: true, trim: true },
8     description: { type: String, default: '' }
9   }, { timestamps: true });
10
11  module.exports = model('Item', itemSchema);
```

Now if we run the entire project :-

```
sahil@Algon:~/Developer/Collage/Backend/exp4> npm start

> start
> node server.js

[dotenv@17.2.2] injecting env (2) from .env -- tip: 📡 vers
Connected to MongoDB
Server is running on http://localhost:3000
```

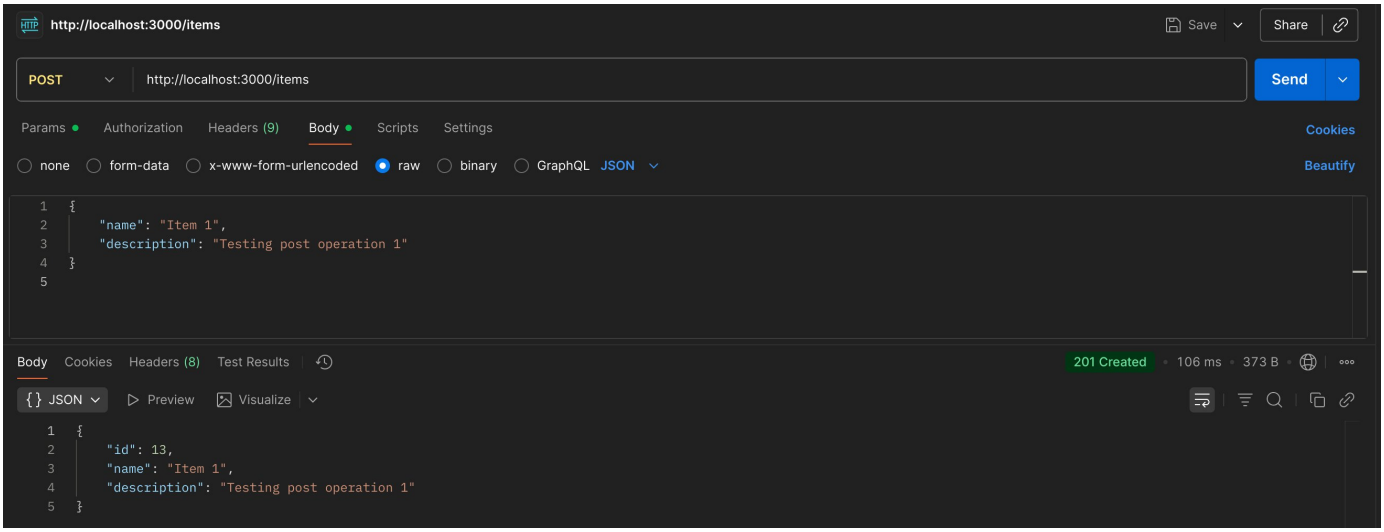
Now here if i perform all CRUD operation:

GET operation :-



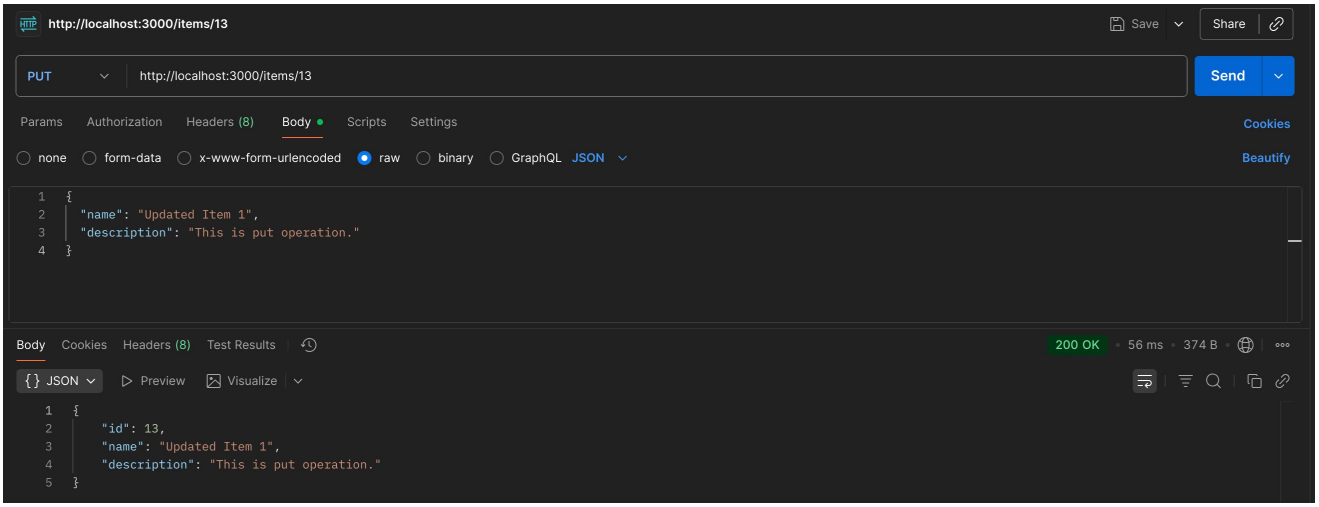
At first our repo is empty

POST operation :-



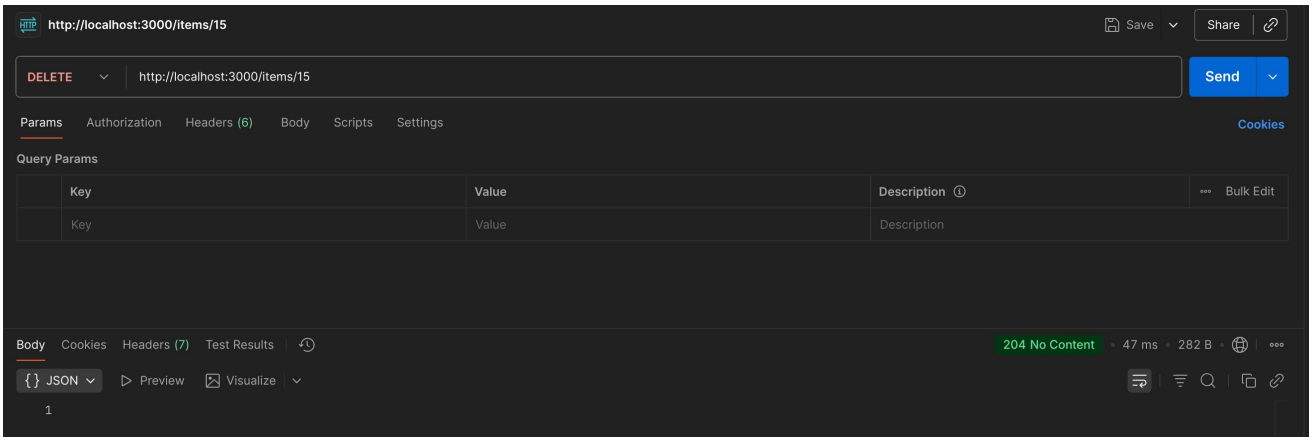
ID is 13 because i used this program before it increment each time new entry is added, Same like this i perform 2 more post operations

PUT operation :-



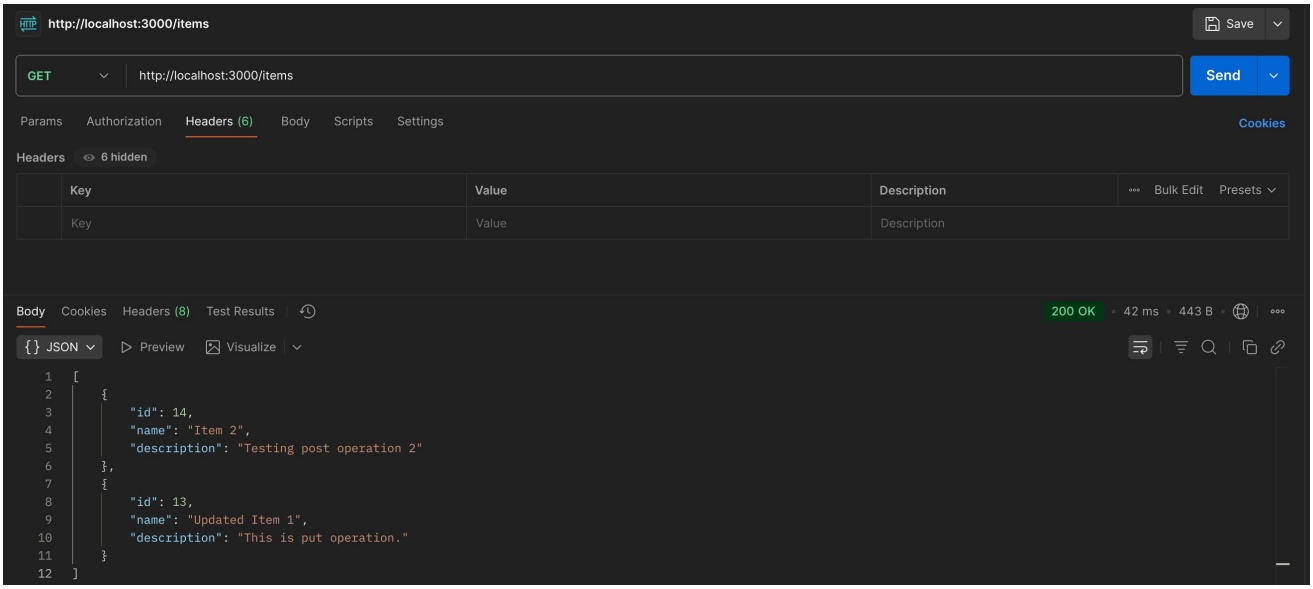
So here i change the entry where id is 13.

DELETE operation :-



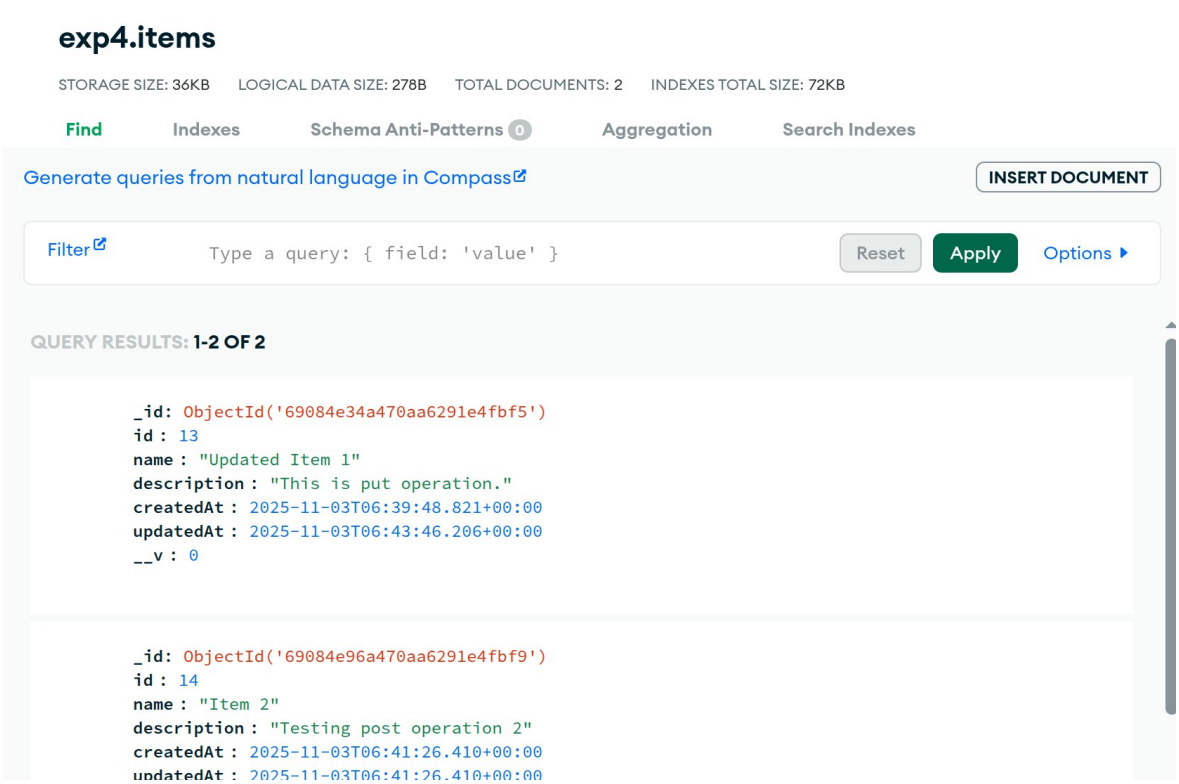
So here i deleted the entry where id is 15;

Now if we again do **GET** operation :-



We can see all the changes.

Now if i open my repo in MongoDB Atlas we can see changes in repo :-



Experiment - 5

Objective: Understanding MongoDB operation and ODM (Mongoose).

Lab Activities:

- Non-relational databases (MongoDB) and ODM (mongoose).
- Setting up and connecting to a database.

Sample Project: Perform all the mongoose operations like pagination,triggers, virtual and indexing in the to-do list API.

Server.js :-

I made changes in Server.js and Added Item/Audit imports, Item.syncIndexes(), a MongoDB change-stream audit writer, and enhanced URL parsing to support pagination/search/sort (repo.findAll now accepts options).

```
const http = require('http');
require('dotenv').config();
const mongoose = require('mongoose');
const repo = require('./repositories/itemRepository');
const Item = require('./models/item');
const Audit = require('./models/audit');

const start = async () => {
  try {
    await mongoose.connect(process.env.MONGODB_URI);
    console.log('Connected to MongoDB');

    // ensure model indexes exist (text index etc.)
    await Item.syncIndexes();
    console.log('Item indexes synced');

    // change-stream: listen for DB changes (requires replica set / Atlas)
    try {
      const changeStream = Item.collection.watch();
      changeStream.on('change', async change => {
        console.log('ChangeStream event:', change.operationType);
        try {
          if (change.fullDocument) {
            await Audit.create({ itemId: change.fullDocument.id, op: change.operationType, doc:
change.fullDocument });
          } else {
            await Audit.create({ itemId: change.documentKey && change.documentKey._id, op:
change.operationType, doc: change });
          }
        } catch (e) { console.warn('Audit save failed', e.message); }
      });
      console.log('Item change-stream started');
    } catch (err) {
      console.warn('Change-stream not started:', err.message);
    }
```

```
const server = http.createServer(async (req, res) => {
  res.setHeader('Access-Control-Allow-Origin', '*');
  res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, OPTIONS');
  res.setHeader('Access-Control-Allow-Headers', 'Content-Type');

  if (req.method === 'OPTIONS') {
    res.writeHead(204);
    res.end();
    return;
  }

  res.setHeader('Content-Type', 'application/json');

  const { method, headers } = req;
  // parse URL and query params reliably
  const base = `http://${headers.host || 'localhost'}`;
  const reqUrl = new URL(req.url, base);
  const pathname = reqUrl.pathname;

  try {
    // GET /items with pagination/search/sort
    if (method === 'GET' && pathname === '/items') {
      const page = Number(reqUrl.searchParams.get('page') || 1);
      const limit = Math.min(100, Number(reqUrl.searchParams.get('limit') || 50));
      const search = reqUrl.searchParams.get('search') || undefined;
      const sortParam = reqUrl.searchParams.get('sort') || undefined;
      let sort;
      if (sortParam) {
        // simple format: field:dir or field1:dir1,field2:dir2  (dir = 1 or -1)
        sort = {};
        sortParam.split(',').forEach(pair => {
          const [field, dir] = pair.split(':');
          sort[field.trim()] = Number(dir) || 1;
        });
      }
      const result = await repo.findAll({ page, limit, search, sort });
      res.writeHead(200);
      res.end(JSON.stringify(result));
      return;
    }

    // GET /items/:id
    if (method === 'GET' && pathname.match(/^\/items\/[0-9]+$/)) {
      const id = pathname.split('/')[2];
      const item = await repo.findById(id);
```

```
    if (item) {
      res.writeHead(200);
      res.end(JSON.stringify(item));
    } else {
      res.writeHead(404);
      res.end(JSON.stringify({ message: 'Item not found' }));
    }
    return;
  }

  // POST /items
  if (method === 'POST' && pathname === '/items') {
    let body = '';
    req.on('data', chunk => body += chunk.toString());
    req.on('end', async () => {
      const { name, description } = JSON.parse(body || '{}');
      if (!name) {
        res.writeHead(400);
        res.end(JSON.stringify({ message: 'Name is required' }));
        return;
      }
      const newItem = await repo.create({ name, description });
      res.writeHead(201);
      res.end(JSON.stringify(newItem));
    });
    return;
  }

  // PUT /items/:id
  if (method === 'PUT' && pathname.match(/^\/items\/[0-9]+$/)) {
    const id = pathname.split('/')[2];
    let body = '';
    req.on('data', chunk => body += chunk.toString());
    req.on('end', async () => {
      const { name, description } = JSON.parse(body || '{}');
      const updated = await repo.update(id, { name, description });
      if (updated) {
        res.writeHead(200);
        res.end(JSON.stringify(updated));
      } else {
        res.writeHead(404);
        res.end(JSON.stringify({ message: 'Item not found' }));
      }
    });
    return;
```

```
      }

      // DELETE /items/:id
      if (method === 'DELETE' && pathname.match(/^\/items\/[0-9]+$/)) {
        const id = pathname.split('/')[2];
        const ok = await repo.delete(id);
        if (ok) {
          res.writeHead(204);
          res.end();
        } else {
          res.writeHead(404);
          res.end(JSON.stringify({ message: 'Item not found' }));
        }
        return;
      }

      // Not found
      res.writeHead(404);
      res.end(JSON.stringify({ message: 'Route not found' }));
    } catch (err) {
      console.error('Request handler error', err);
      res.writeHead(500);
      res.end(JSON.stringify({ message: 'Internal server error' }));
    }
  });

  const port = process.env.PORT || 3000;
  server.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
  });

 } catch (err) {
  console.error('Failed to start server', err);
  process.exit(1);
 }
};

start();
```

I also make changes in item.js and Added a text index and shortDescription virtual (exposed in JSON), tightened validation/trimming, and post-save/update/delete hooks that create Audit records.

Item.js :-

```
const { Schema, model } = require('mongoose');
const Audit = require('./audit');
const leanVirtuals = require('mongoose-lean-virtuals');

const itemSchema = new Schema({
  id: { type: Number, required: true, unique: true, index: true },
  name: { type: String, required: true, trim: true, minlength: 1, maxlength: 200 },
  description: { type: String, default: '', maxlength: 1000, trim: true }
}, { timestamps: true, toJSON: { virtuals: true }, toObject: { virtuals: true } });

itemSchema.index({ name: 'text', description: 'text' });

itemSchema.virtual('shortDescription').get(function () {
  return (this.description || '').slice(0, 100);
});

itemSchema.pre('save', function (next) {
  if (this.name) this.name = this.name.trim();
  if (this.description && typeof this.description === 'string') {
    this.description = this.description.trim();
  }
  next();
});

itemSchema.post('save', function (doc) {
  Audit.create({ itemId: doc.id, op: 'insert', doc }).catch(()=>{});
  console.log('Trigger: item saved', { id: doc.id, name: doc.name });
});

itemSchema.post('findOneAndUpdate', function (doc) {
  if (doc) {
    Audit.create({ itemId: doc.id, op: 'update', doc }).catch(()=>{});
    console.log('Trigger: item updated', { id: doc.id, name: doc.name });
  }
```

```
});

itemSchema.post('findOneAndDelete', function (doc) {
  if (doc) {
    Audit.create({ itemId: doc.id, op: 'delete', doc }).catch(()=>{});
    console.log('Trigger: item deleted', { id: doc.id });
  }
});

itemSchema.plugin(leanVirtuals);

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

audit.js :-

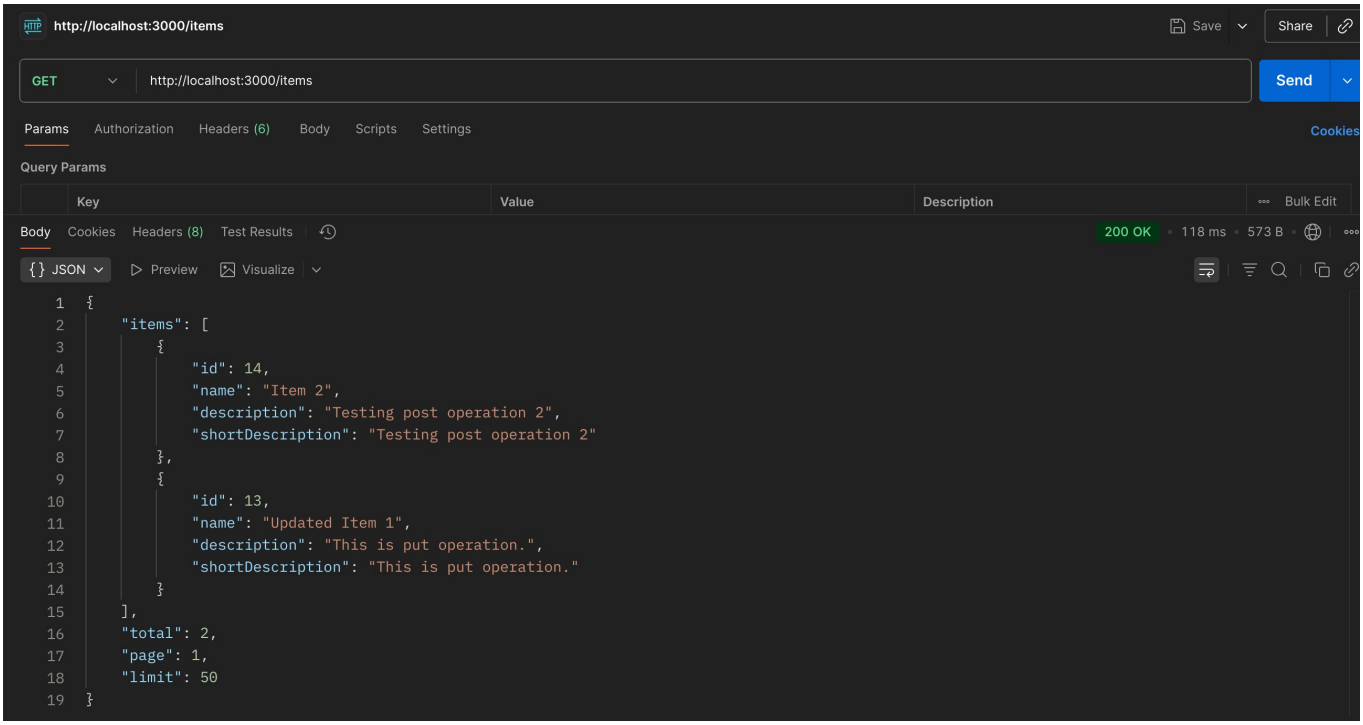
```
const { Schema, model } = require('mongoose');

const auditSchema = new Schema({
  itemId: Number,
  op: String, // insert/update/delete
  doc: Schema.Types.Mixed,
  ts: { type: Date, default: Date.now }
});

module.exports = model('Audit', auditSchema);
```

Now if i run this entire project :-

```
sahil@Algon:~/Developer/Collage/Backend/exp5> npm start

> start
> node server.js

[dotenv@17.2.2] injecting env (2) from .env -- tip: ⚙️ override existing env vars with { override: true }
Connected to MongoDB
Item indexes synced
Item change-stream started
Server is running on http://localhost:3000
```

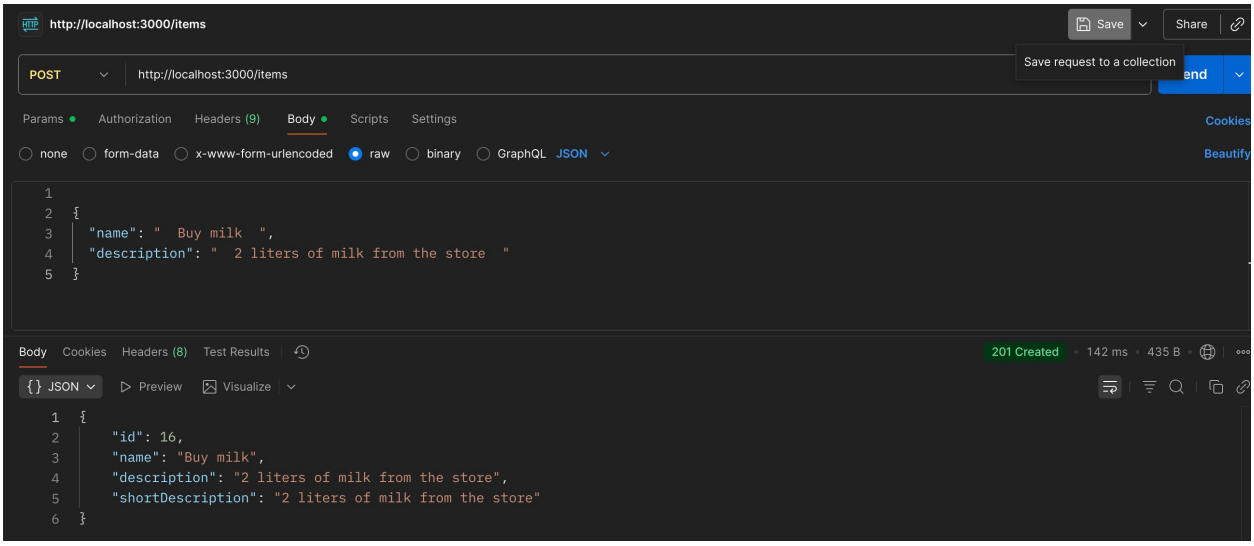
So i am using same repo which i used in exp4 :-

So if we perform **GET operation:**

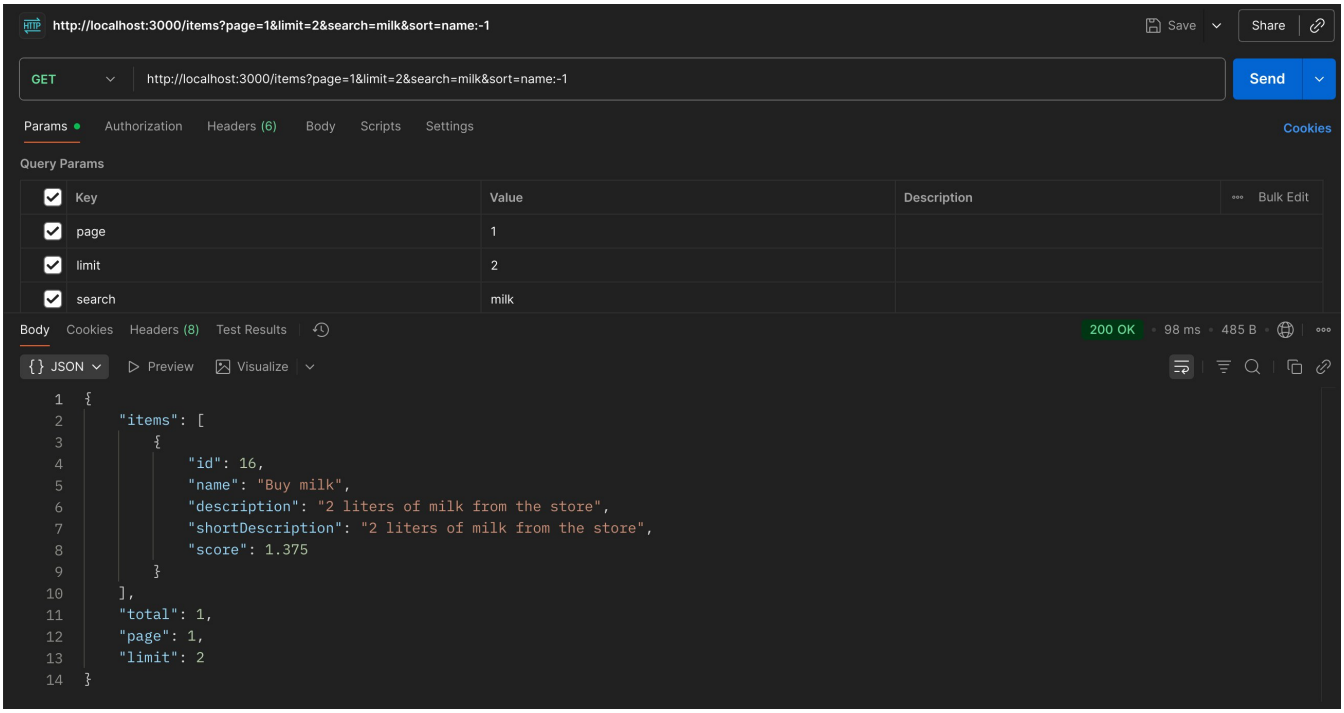


Now as we can observe the changes its showing pages.

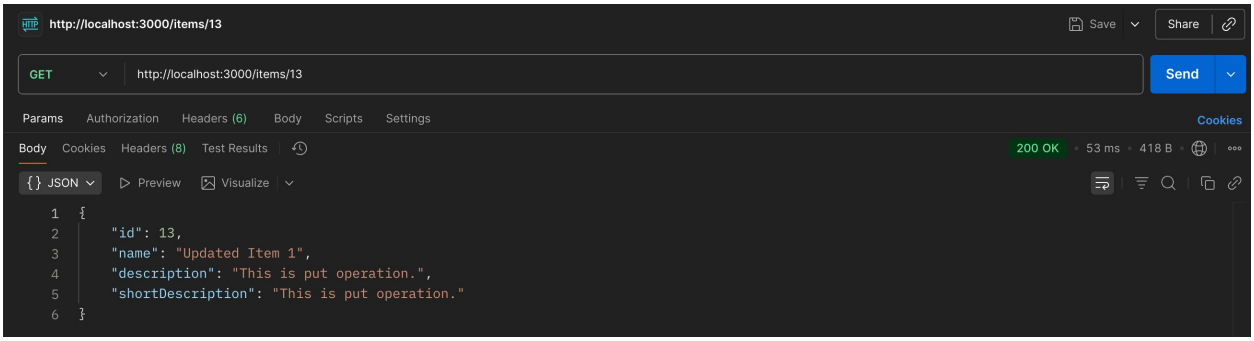
Here if we add entry using **POST:**



Now if i apply pagination + search + sort in **GET** operation :



To access single element



Experiment - 6

Objective: Understand middleware functions and advanced routing.

Lab Activities:

- Introduction to middleware in backend frameworks.
- Creating custom middleware for logging, error handling, etc.
- Advanced routing techniques (nested routes, dynamic routes).

Sample Project: Implement logging and error handling middleware in a To-Do List API.

So for ERROR handling i created **errorHandler.js** :-

middleware > JS errorHandler.js > ...

```
module.exports = function errorHandler(err, req, res, next) {

  if (!err) return next();
  const status = err.status && Number(err.status) >= 400 ? Number(err.status) : 500;
  const payload = { message: err.message || 'Internal server error' };

  if (process.env.NODE_ENV !== 'production') {
    payload.stack = err.stack;
  }
  console.error('ErrorHandler:', err.message || err);
  res.status(status).json(payload);
};

```

And for manage logging i created **logger.js :-**

middleware > JS logger.js > ...

```
module.exports = function logger(req, res, next) {
  const start = Date.now();

  const { method, originalUrl } = req;

  res.on('finish', () => {
    const duration = Date.now() - start;
    const msg = `[${new Date().toISOString()}] ${method} ${originalUrl} ${res.statusCode} - ${duration}ms`

    if (method === 'POST' || method === 'PUT') {
      let preview;
      try { preview = JSON.stringify(req.body); } catch (e) { preview = '[unserializable]'; }
      const previewShort = preview && preview.length > 200 ? preview.slice(0,200) + '…' : preview;
      console.log(msg + ' ' + previewShort);
    } else {
      console.log(msg);
    }
  });

  next();
};
```

And imported these two in server.js

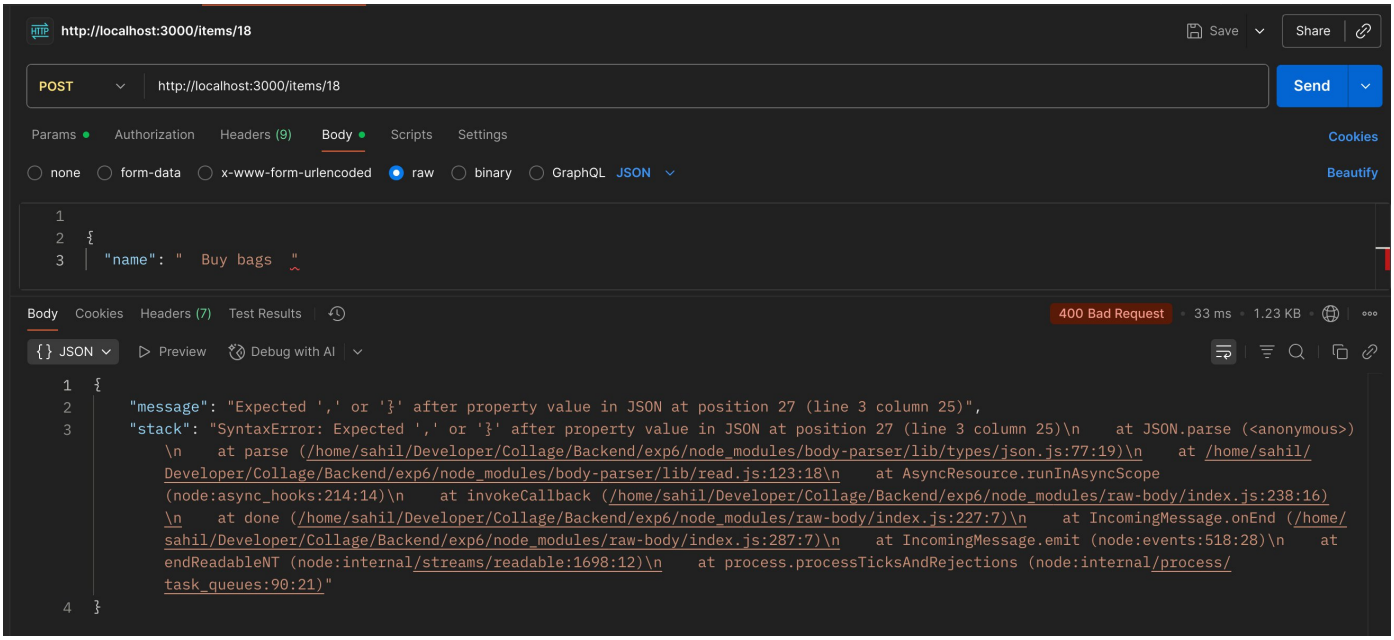
Now if i run the project :-

```
sahil@Algon:~/Developer/Collage/Backend/exp6> npm start

> start
> node server.js

[dotenv@17.2.2] injecting env (2) from .env -- tip: ⚙️ suppress all logs with { quiet: true }
Connected to MongoDB
Item indexes synced
Item change-stream started
Server is running on http://localhost:3000
```

Now as i perform any CRUD operation in postman i can see its log one terminal :-

```
sahil@Algon:~/Developer/Collage/Backend/exp6> npm start

> start
> node server.js

[dotenv@17.2.2] injecting env (2) from .env -- tip: ⚙️ suppress all logs with { quiet: true }
Connected to MongoDB
Item indexes synced
Item change-stream started
Server is running on http://localhost:3000
[2025-11-03T16:32:40.962Z] GET /items 200 - 97ms
Trigger: item saved { id: 17, name: 'Buy book' }
[2025-11-03T16:33:40.557Z] POST /items 201 - 116ms {"name":"  Buy book  ","description":"  4 books  "}
ChangeStream event: insert
```

Now if i add broken json data in postman :



Terminal :

```
sahil@Algon:~/Developer/Collage/Backend/exp6> npm start

> start
> node server.js

[dotenv@17.2.2] injecting env (2) from .env -- tip: ⚙️ override existing env vars with { override: true }
Connected to MongoDB
Item indexes synced
Item change-stream started
Server is running on http://localhost:3000
[2025-11-03T16:39:18.421Z] GET /items 200 - 109ms
ErrorHandler: Expected ',' or '}' after property value in JSON at position 27 (line 3 column 25)
```

Now we can see error handling working her in terminal output.