

## NIMBUS (MERN PROJECT)

### Steps you follow:

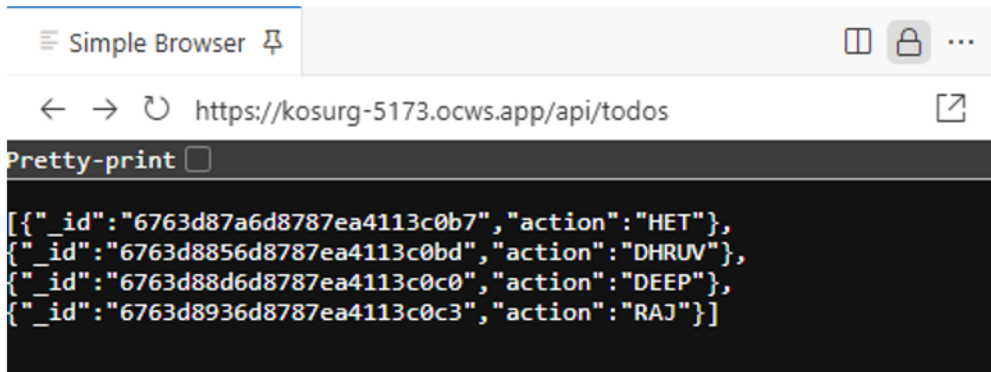
|        |             |
|--------|-------------|
| Step-1 | cd client   |
| Step-2 | npm i       |
| Step-3 | npm run dev |
| Step-4 | cd ..       |
| Step-5 | npm i       |
| Step-6 | npm run dev |

Simple Browser: <https://kosurg-5173.ocws.app>

Add- TODOS NAMES

- HET
- DHRUV
- RAJ
- DEEP

Simple Browser: <https://kosurg-5173.ocws.app/api/todos>



## Thunder Clients:

### New Request –

|     |  |      |
|-----|--|------|
| GET | https://kosurg-5173.ocws.app/api/todos | Send |
|-----|--|------|

### Output:

|  |  |      |
|--|--|------|
| GET  | https://kosurg-5173.ocws.app/api/todos | Send |
| Status: 200 OK   Size: 204 Bytes   Time: 1.00 s   Response   |  |      |
| <pre>1  [ 2    { 3      "_id": "6763d87a6d8787ea4113c0b7", 4      "action": "HET" 5    }, 6    { 7      "_id": "6763d8856d8787ea4113c0bd", 8      "action": "DHRUV" 9    }, 10   { 11     "_id": "6763d88d6d8787ea4113c0c0", 12     "action": "DEEP" 13   }, 14   { 15     "_id": "6763d8936d8787ea4113c0c3", 16     "action": "RAJ" 17   } 18 ]</pre> |  |      |

## CRUD OPERATION IN NIMBUS

### *CRUD – PROGRAMS* *(index.js file)*

---

```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const routes = require('./routes/userroutes.js');
require('dotenv').config();

const app = express();

const port = process.env.PORT || 5000;

// Connect to the database
mongoose
  .connect(process.env.MONGO_DB_URI, { useNewUrlParser: true })
  .then(() => console.log(` Database connected successfully `))
  .catch((err) => console.log(err));

// Since mongoose's Promise is deprecated, we override it with Node's Promise
mongoose.Promise = global.Promise;

app.use((req, res, next) => {
  res.header('Access-Control-Allow-Origin', '*');
  res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept');
  next();
});
```

```
app.use(bodyParser.json());
```

```
app.use('/api', routes);
```

```
app.use((err, req, res, next) => {
```

```
  console.log(err);
```

```
  next();
```

```
});
```

```
app.listen(port, () => {
```

```
  console.log(` Server running on port ${port} `);
```

```
});
```

---

*(usercontrollers.js file)*

---

```
const User = require("../models/usermodels.js");

const createUser= async (req,res)=>{
  try{
    const userData = new User(req.body);
    if(!userData)
    {
      res.status(404).json({msg:"user not found"});
    }

    await userData.save();
    res.status(200).json({msg:"user got created successfully"})
  }
  catch(error)
  {
    res.status(500).json({err:error})
  }
}

const getAll = async(req,res)=>{
  try{const userData= await User.find();
    if(!userData){
      res.status(404).json({message:"User not found."});
    }
    res.status(200).json(userData);
  }
```

```
}  
catch(error){  
  res.status(500).json({err:error});  
}  
  
}
```

```
const findUserById = async(req,res)=>{  
  try{  
    const userid= req.params.id;  
  
    const userData= await User.findById(userid);  
  
    if(!userData){  
      res.status(404).json({message:"User not found."});  
    }  
    res.status(200).json(userData);  
  }  
  catch(error){  
    res.status(500).json({err:error});  
  }  
  
}
```

```
const Updateuser= async(req,res)=>{  
  try{  
    const userid= req.params.id;
```

```

const userData= await User.findById(userid);

if(!userData){
  res.status(404).json({message:"User not found."});
}

const updatedata= await User.findByIdAndUpdate(userid,req.body,{new:true});
res.status(200).json({msg:"User Updated successfully"});
}
catch(error){
  res.status(500).json({err:error});
}
}

```

```

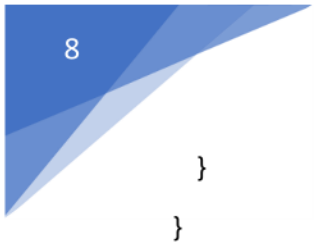
const DeleteUser = async(req,res)=>{
  try{
    const userid= req.params.id;

    const userData= await User.findById(userid);

    if(!userData){
      res.status(404).json({message:"User not found."});
    }

    const deleteData = await User.findByIdAndDelete(userid);
    res.status(200).json({msg:"User Deleted Successfully"});
  }
  catch(error){
    res.status(500).json({err:error});
  }
}

```



```
module.exports={  
  DeleteUser,  
  createUser,  
  getAll,  
  findUserById,  
  Updateuser  
};
```



---

*(usermodels.js file)*

---

```
const mongoose =require('mongoose');
const userschemas =new mongoose.Schema(
  {
    name:{
      type:String,
      required:true
    },
    email:{
      type:String,
      required:true
    },
    password:{
      type:String,
      required:true
    },
    age:{
      type:Number,
      required:true
    }
  },
  {
    timestamps:true
  }
)
const User=mongoose.model('User',userschemas);
module.exports=User;
```

---

**(userroutes.js file)**

---

```
const {createUser,getAll,findUserById,Updateuser,DeleteUser} =
require("../controllers/usercontrollers.js");

const express=require("express");

const route= express.Router();

route.post('/create',createUser)

route.get('/getall',getAll)

route.get("/finduser/:id",findUserById)

route.put("/update/:id",Updateuser)

route.delete('/delete/:id',DeleteUser)

module.exports= route;
```

## Thunder Clients (**CREATE METHOD OUTPUT**)

STEP:1 OPEN **THUNDER CLIENT**

STEP:2 **NEW REQUEST**

STEP:3 SET ON (**POST**)

STEP:4 <http://localhost:5000/api/create>

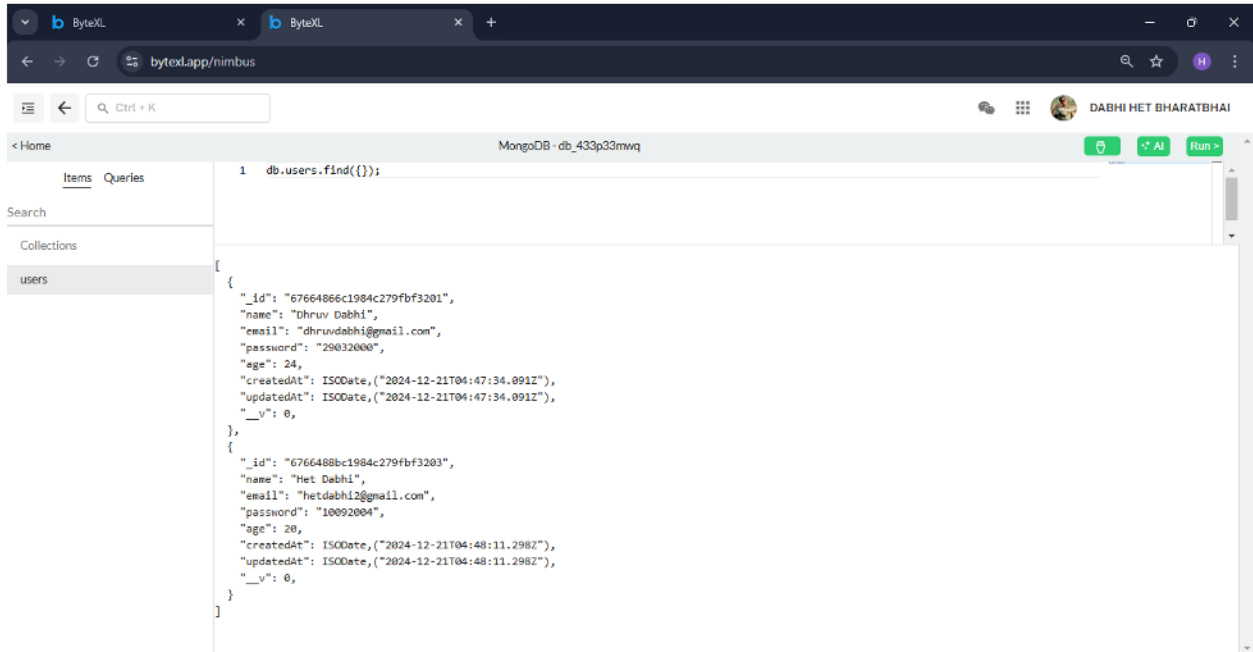
STEP:5 **SEND**

The screenshot displays the Thunder Client interface. At the top, a request is configured as a POST to `http://localhost:5000/api/create`. The 'Body' tab is selected, showing a JSON payload: `{ "name": "HetDabhi", "email": "hetdabhi2@gmail.com", "password": "10092004", "age": 20 }`. Below the request, the status is '200 OK', size is '39 Bytes', and time is '217 ms'. The 'Response' tab is expanded, showing a JSON response: `{ "msg": "user got created successfully" }`. A 'Copy' button is visible next to the response.

**Output:**

```
root@f553e810bfc5:~/my-project# node index.js
(node:834) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option:
e next major version
(Use `node --trace-warnings ...` to show where the warning was created)
Server running on port 5000
Database connected successfully
█
```

## MongoDB (CREATE METHOD OUTPUT)



The screenshot shows the MongoDB Compass web interface. The browser address bar displays 'bytexlapp/nimbus'. The interface includes a sidebar with 'Collections' and 'users' selected. The main area shows the query '1 db.users.find({});' and its output as a JSON array of two user documents. The first document is for 'Dhruv Dabhi' and the second is for 'Het Dabhi'. Both documents include fields for '\_id', 'name', 'email', 'password', 'age', 'createdAt', 'updatedAt', and '\_\_v'.

```
1 db.users.find({});
```

```
[
  {
    "_id": "67664866c1984c279fbf3201",
    "name": "Dhruv Dabhi",
    "email": "dhrumdabhi@gmail.com",
    "password": "29032000",
    "age": 24,
    "createdAt": ISODate("2024-12-21T04:47:34.091Z"),
    "updatedAt": ISODate("2024-12-21T04:47:34.091Z"),
    "__v": 0,
  },
  {
    "_id": "6766488bc1984c279fbf3203",
    "name": "Het Dabhi",
    "email": "hetdabhi2@gmail.com",
    "password": "10092004",
    "age": 20,
    "createdAt": ISODate("2024-12-21T04:48:11.298Z"),
    "updatedAt": ISODate("2024-12-21T04:48:11.298Z"),
    "__v": 0,
  }
]
```

## Thunder Clients (GETALL DATA METHOD OUTPUT)

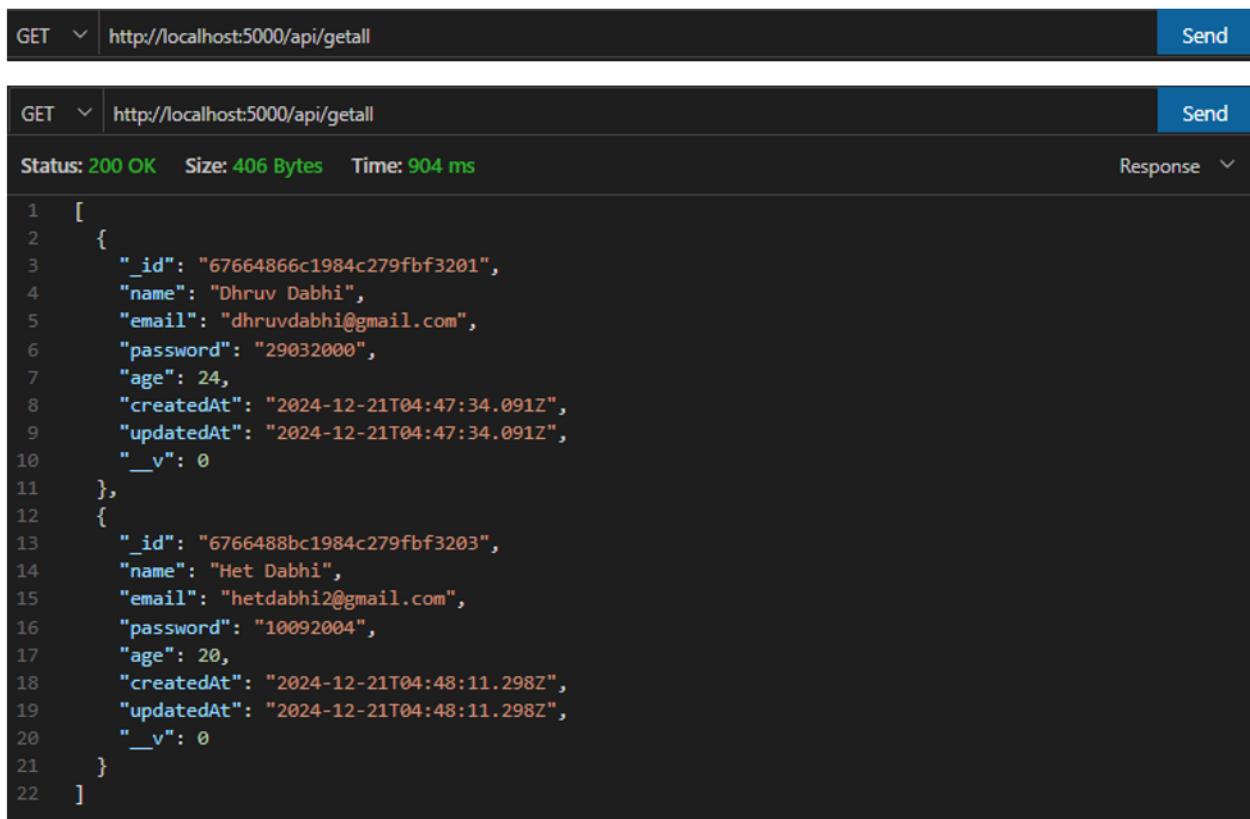
STEP:1 OPEN THUNDER CLIENT

STEP:2 **NEW REQUEST**

STEP:3 SET ON (GET)

STEP:4 <http://localhost:5000/api/getall>

STEP:5 **SEND**



```
GET  http://localhost:5000/api/getall  Send

GET  http://localhost:5000/api/getall  Send

Status: 200 OK  Size: 406 Bytes  Time: 904 ms  Response  v

1  [
2  {
3    "_id": "67664866c1984c279fbf3201",
4    "name": "Dhruv Dabhi",
5    "email": "dhruvdabhi@gmail.com",
6    "password": "29032000",
7    "age": 24,
8    "createdAt": "2024-12-21T04:47:34.091Z",
9    "updatedAt": "2024-12-21T04:47:34.091Z",
10   "__v": 0
11  },
12  {
13    "_id": "6766488bc1984c279fbf3203",
14    "name": "Het Dabhi",
15    "email": "hetdabhi2@gmail.com",
16    "password": "10092004",
17    "age": 20,
18    "createdAt": "2024-12-21T04:48:11.298Z",
19    "updatedAt": "2024-12-21T04:48:11.298Z",
20    "__v": 0
21  }
22  ]
```

## Thunder Clients (**FINDUSER DATA METHOD OUTPUT**)

STEP:1 OPEN **THUNDER CLIENT**

STEP:2 **NEW REQUEST**

STEP:3 SET ON (**GET**)

STEP:4 <http://localhost:5000/api/finduser/6766488bc1984c279fbf3203>

STEP:5 **SEND**

The screenshot displays the Thunder Client interface. At the top, a request bar shows a GET method to the URL `http://localhost:5000/api/finduser/6766488bc1984c279fbf3203` with a 'Send' button. Below this, a tabbed interface includes 'Query', 'Headers', 'Auth', 'Body', 'Tests', and 'Pre Run'. The 'Query' tab is active, showing the response status as '200 OK', size as '200 Bytes', and time as '199 ms'. The response body is a JSON object with the following fields: `_id`, `name`, `email`, `password`, `age`, `createdAt`, `updatedAt`, and `__v`. The JSON is displayed with line numbers 1 through 10.

```
1  {
2    "_id": "6766488bc1984c279fbf3203",
3    "name": "Het Dabhi",
4    "email": "hetdabhi2@gmail.com",
5    "password": "10092004",
6    "age": 20,
7    "createdAt": "2024-12-21T04:48:11.298Z",
8    "updatedAt": "2024-12-21T04:48:11.298Z",
9    "__v": 0
10 }
```

## Thunder Clients (**UPDATE DATA METHOD OUTPUT**)

STEP:1 OPEN **THUNDER CLIENT**

STEP:2 **NEW REQUEST**

STEP:3 SET ON (**PUT**)

STEP:4 <http://localhost:5000/api/update/6766488bc1984c279fbf3203>

STEP:5 **SEND**

The screenshot displays the Thunder Client interface. At the top, a PUT request is configured with the URL `http://localhost:5000/api/update/6766488bc1984c279fbf3203`. The 'Body' tab is selected, showing a JSON payload: 

```
{
  "name": "Het Bharatbhai Dabhi",
  "email": "hetdabhi2@gmail.com",
  "password": "10092004",
  "age": 20
}
```

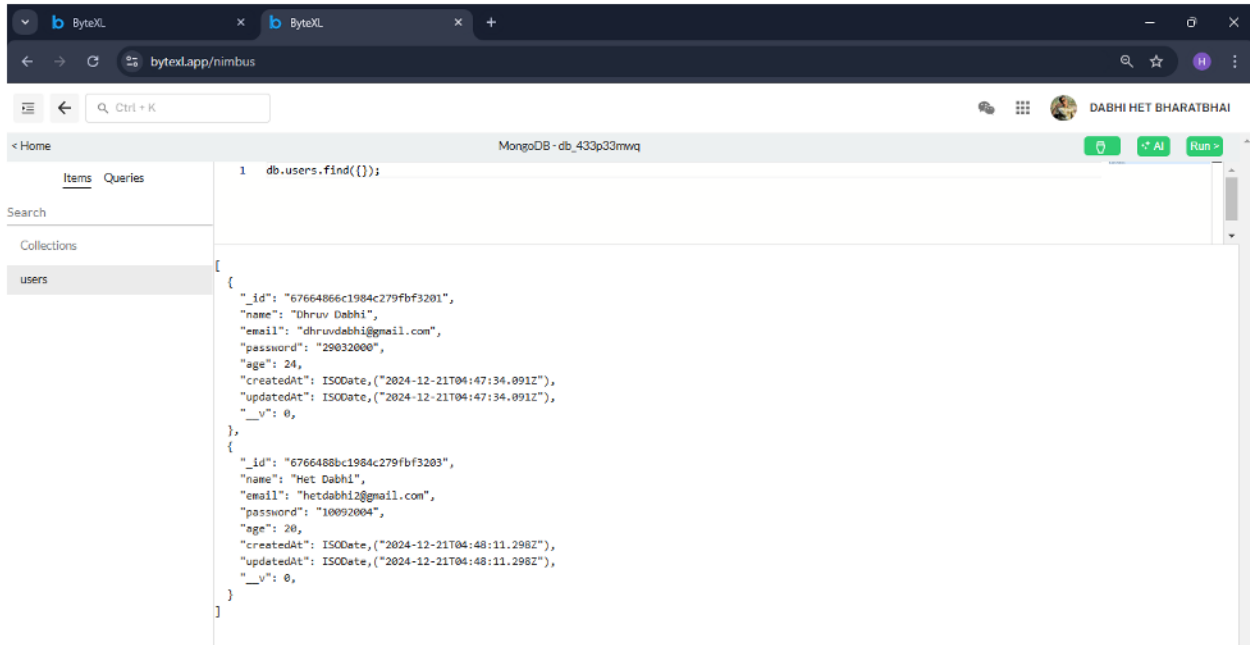
. Below the body, the status is **200 OK**, with a size of **35 Bytes** and a time of **395 ms**. The 'Response' tab shows the JSON output: 

```
{
  "msg": "User Updated successfully"
}
```

. A 'Copy' button is visible next to the response.

## MONGODB

### Before:

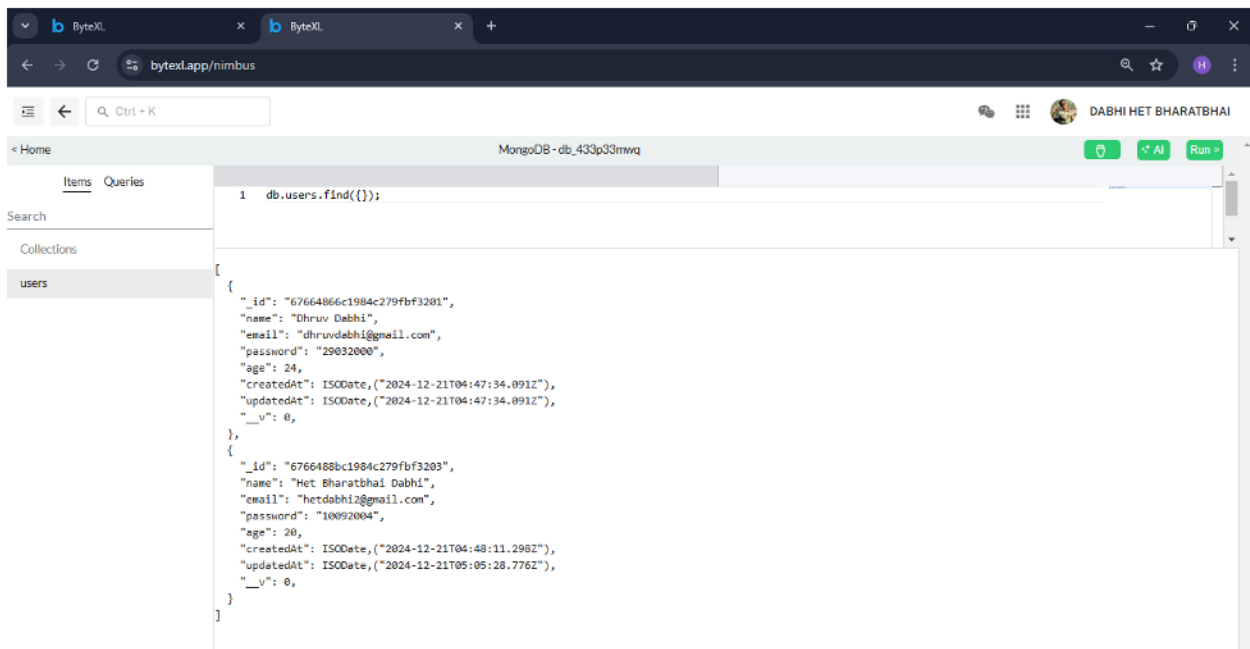


The screenshot shows the MongoDB Compass interface. The query bar contains `1 db.users.find({});`. The results pane shows two documents from the `users` collection. The first document is for 'Dhruv Dabhi' and the second is for 'Het Dabhi'.

```
1 db.users.find({});
```

```
{
  "_id": "67664866c1984c279fbf3201",
  "name": "Dhruv Dabhi",
  "email": "dhruvdabhi@gmail.com",
  "password": "29032000",
  "age": 24,
  "createdAt": ISODate("2024-12-21T04:47:34.091Z"),
  "updatedAt": ISODate("2024-12-21T04:47:34.091Z"),
  "__v": 0,
},
{
  "_id": "6766488bc1984c279fbf3203",
  "name": "Het Dabhi",
  "email": "hetdabhi2@gmail.com",
  "password": "10092004",
  "age": 20,
  "createdAt": ISODate("2024-12-21T04:48:11.298Z"),
  "updatedAt": ISODate("2024-12-21T04:48:11.298Z"),
  "__v": 0,
}
```

### After:



The screenshot shows the MongoDB Compass interface after a change. The query bar still contains `1 db.users.find({});`. The results pane now shows three documents. The first two are the same as before, but the third document is for 'Het Bharatbhai Dabhi'.

```
1 db.users.find({});
```

```
{
  "_id": "67664866c1984c279fbf3201",
  "name": "Dhruv Dabhi",
  "email": "dhruvdabhi@gmail.com",
  "password": "29032000",
  "age": 24,
  "createdAt": ISODate("2024-12-21T04:47:34.091Z"),
  "updatedAt": ISODate("2024-12-21T04:47:34.091Z"),
  "__v": 0,
},
{
  "_id": "6766488bc1984c279fbf3203",
  "name": "Het Bharatbhai Dabhi",
  "email": "hetdabhi2@gmail.com",
  "password": "10092004",
  "age": 20,
  "createdAt": ISODate("2024-12-21T04:48:11.298Z"),
  "updatedAt": ISODate("2024-12-21T05:05:28.776Z"),
  "__v": 0,
}
```



## Thunder Clients (**DELETE DATA METHOD OUTPUT**)

STEP:1 OPEN **THUNDER CLIENT**

STEP:2 **NEW REQUEST**

STEP:3 SET ON (**DELETE**)

STEP:4 <http://localhost:5000/api/delete/67664866c1984c279fbf3201>

STEP:5 **SEND**

The screenshot displays the Thunder Client interface. At the top, a request bar shows the method 'DELETE', a dropdown arrow, the URL 'http://localhost:5000/api/delete/67664866c1984c279fbf3201', and a 'Send' button. Below this, a tabbed interface includes 'Query', 'Headers 2', 'Auth', 'Body' (selected), 'Tests', and 'Pre Run'. The 'Body' tab shows the response status 'Status: 200 OK', size 'Size: 35 Bytes', and time 'Time: 1.00 s'. A 'Response' dropdown is on the right. The response body is a JSON object: { "msg": "User Deleted Successfully" }, displayed across three lines.

```
DELETE http://localhost:5000/api/delete/67664866c1984c279fbf3201 Send
```

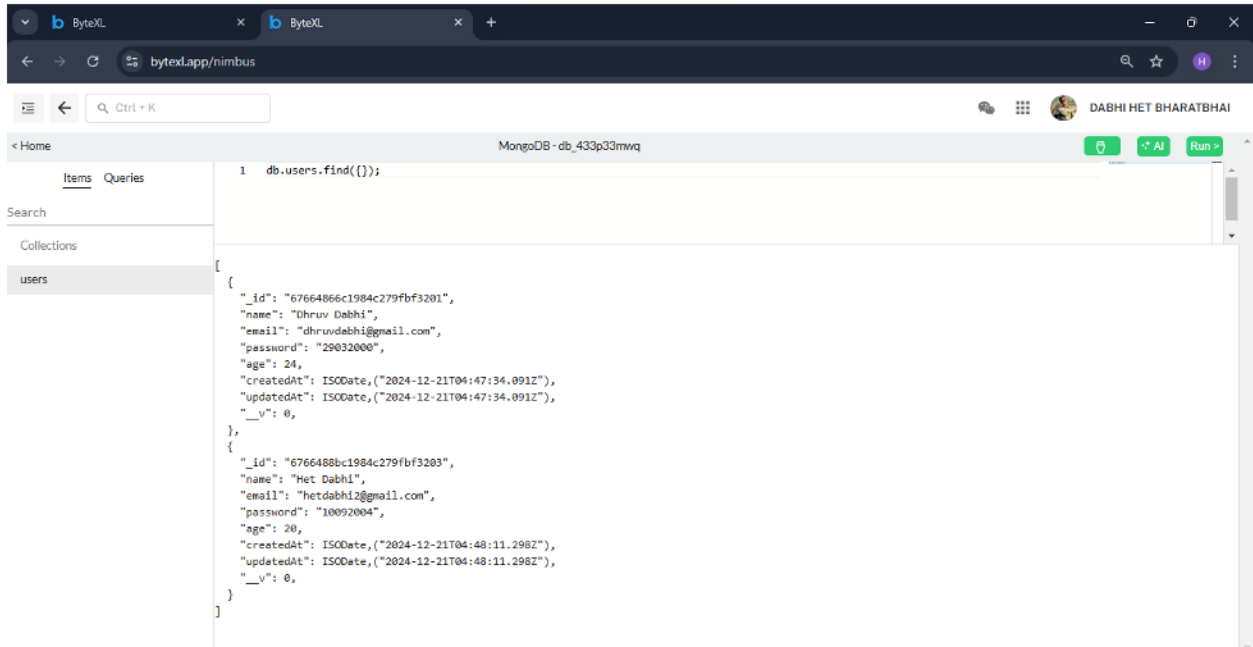
Query Headers 2 Auth **Body** Tests Pre Run

Status: 200 OK Size: 35 Bytes Time: 1.00 s Response

```
1 {
2   "msg": "User Deleted Successfully"
3 }
```

## MONGODB

### Before:

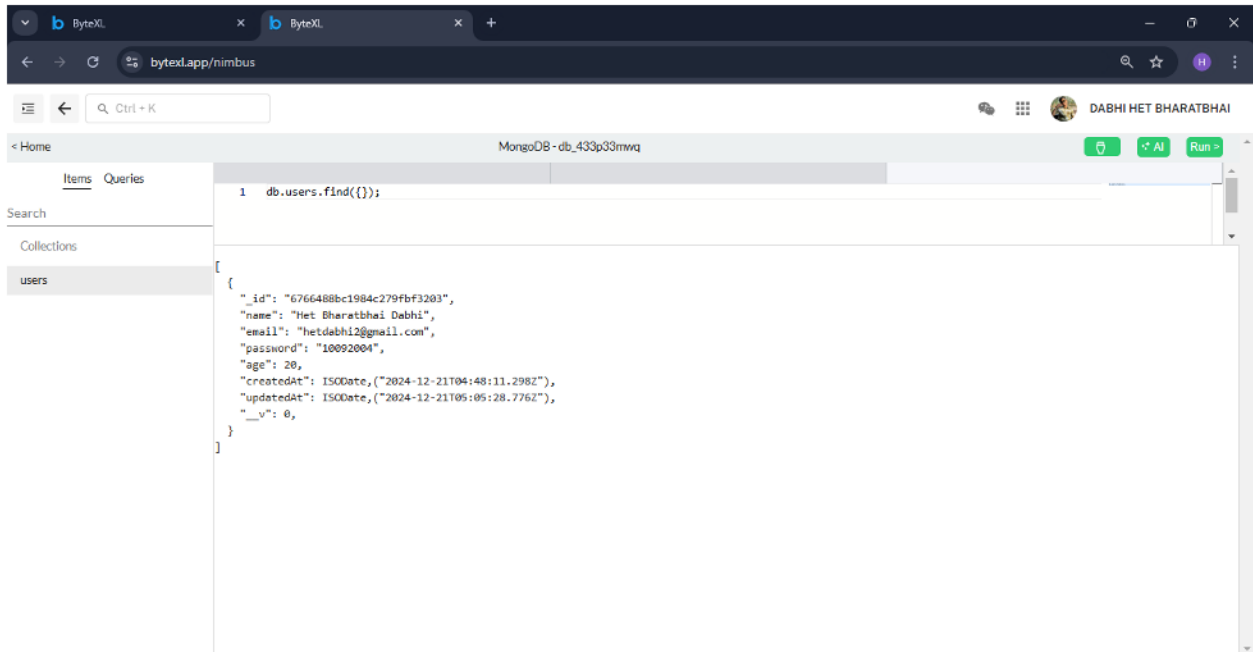


The screenshot shows the MongoDB Compass web interface. The browser address bar displays `bytexl.app/nimbus`. The interface includes a sidebar with 'Items' and 'Queries' tabs, a search bar, and a 'Collections' list with 'users' selected. The main area shows a query `1 db.users.find({});` and its results. The results are an array of two user documents. The first document has an `_id` of `"67664866c1984c279fbf3201"`, a `name` of `"Dhruv Dabhi"`, an `email` of `"dhruvdabhi@gmail.com"`, a `password` of `"29032000"`, an `age` of `24`, and `createdAt` and `updatedAt` timestamps. The second document has an `_id` of `"6766488bc1984c279fbf3203"`, a `name` of `"Het Dabhi"`, an `email` of `"hetdabhi2@gmail.com"`, a `password` of `"10092004"`, an `age` of `20`, and `createdAt` and `updatedAt` timestamps.

```
1 db.users.find({});
```

```
[
  {
    "_id": "67664866c1984c279fbf3201",
    "name": "Dhruv Dabhi",
    "email": "dhruvdabhi@gmail.com",
    "password": "29032000",
    "age": 24,
    "createdAt": ISODate("2024-12-21T04:47:34.091Z"),
    "updatedAt": ISODate("2024-12-21T04:47:34.091Z"),
    "__v": 0,
  },
  {
    "_id": "6766488bc1984c279fbf3203",
    "name": "Het Dabhi",
    "email": "hetdabhi2@gmail.com",
    "password": "10092004",
    "age": 20,
    "createdAt": ISODate("2024-12-21T04:48:11.298Z"),
    "updatedAt": ISODate("2024-12-21T04:48:11.298Z"),
    "__v": 0,
  }
]
```

### After:



The screenshot shows the MongoDB Compass web interface after a change. The browser address bar displays `bytexl.app/nimbus`. The interface is similar to the 'Before' screenshot, but the results array now contains only one document. This document has an `_id` of `"6766488bc1984c279fbf3203"`, a `name` of `"Het Bharatbhai Dabhi"`, an `email` of `"hetdabhi2@gmail.com"`, a `password` of `"10092004"`, an `age` of `20`, and `createdAt` and `updatedAt` timestamps.

```
1 db.users.find({});
```

```
[
  {
    "_id": "6766488bc1984c279fbf3203",
    "name": "Het Bharatbhai Dabhi",
    "email": "hetdabhi2@gmail.com",
    "password": "10092004",
    "age": 20,
    "createdAt": ISODate("2024-12-21T04:48:11.298Z"),
    "updatedAt": ISODate("2024-12-21T05:05:28.776Z"),
    "__v": 0,
  }
]
```