

Code in notesController.js:

```
import Note from "../models>Note.js"

export async function getAllNotes(req, res) {
  try {
    // Note.find(), to get every single note
    const notes = await Note.find()
    // Send status, send notes as json
    res.status(200).json(notes)
  } catch (error) {
    // For debugging purpose
    console.error("Error in getAllNotes controller", error)
    res.status(500).json({message: "Internal Server Error"})
  }
}

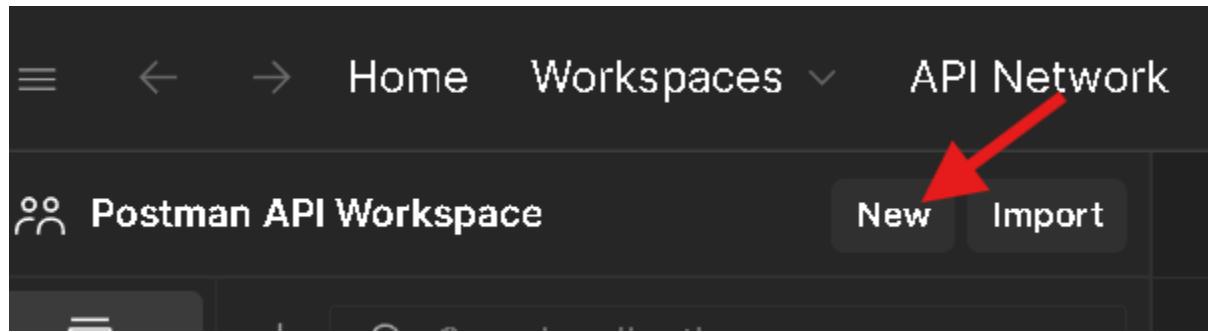
export function createNote(req, res) {
  // Gets data in form of json
  res.status(201).json({message: "Note created successfully!"})
}

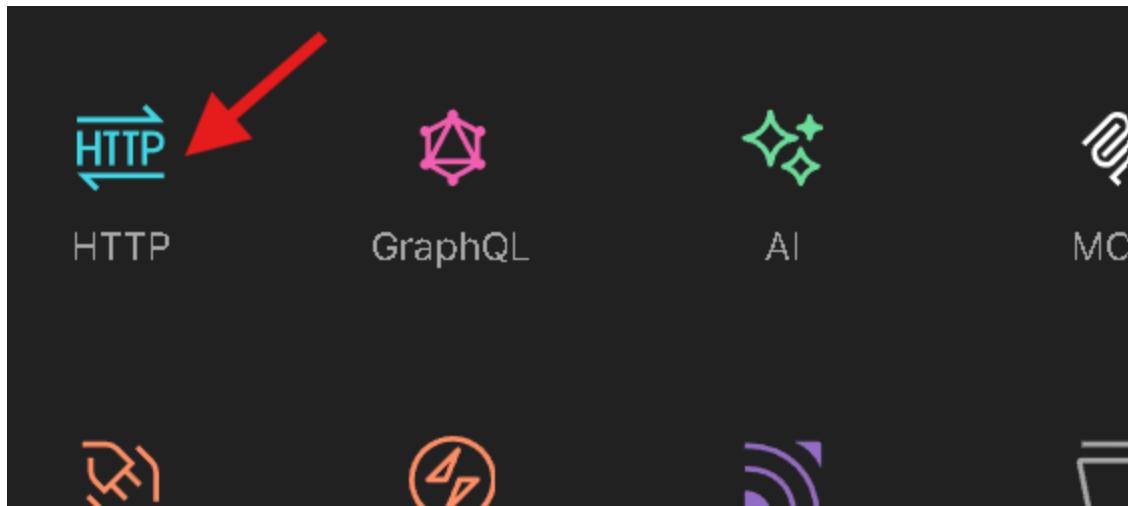
export function updateNote(req, res) {
  res.status(200).json({message: "Note updated successfully!"})
}

export function deleteNote(req, res) {
  res.status(200).json({message: "Note deleted successfully!"})
}
```

Start using Postman for HTTP Requests:

To test the http request:





Copy the URL:

A screenshot of the Postman interface. The address bar at the top shows the URL 'localhost:5001/api/notes'. A red arrow points to the URL field. Below the address bar, the text 'Pretty print' has a checkbox next to it. The main body area displays the JSON response: '[]'.

Paste it:

A screenshot of the Postman interface. The address bar at the top shows 'http://localhost:5001/api/notes'. Below it, the request configuration panel shows 'GET' selected and the URL 'http://localhost:5001/api/notes' in the 'URL' field. A red arrow points to the URL field. The 'Send' button is visible on the right.

Send request:

A screenshot of the Postman interface. The address bar at the top shows 'http://localhost:5001/api/notes'. Below it, the request configuration panel shows 'GET' selected and the URL 'http://localhost:5001/api/notes' in the 'URL' field. A red arrow points to the 'Send' button. The 'Send' button is highlighted in blue.

Got response:

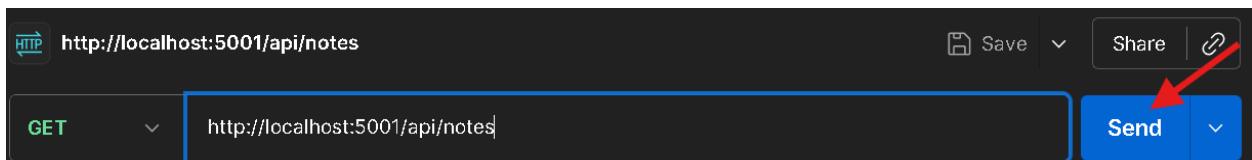
A screenshot of the Postman interface showing the response details. At the top right, the status '200 OK' is displayed in green. Below it, the 'Body' tab is selected, showing a JSON response: '1 []'. A red arrow points to the status code '200 OK'.

If the code breaks at try block:

Changed from find to findx

```
try {
    // Note.find(), to get every single note
    const notes = await Note.findx()
    // Send status, send notes as json
    res.status(200).json(notes)
```

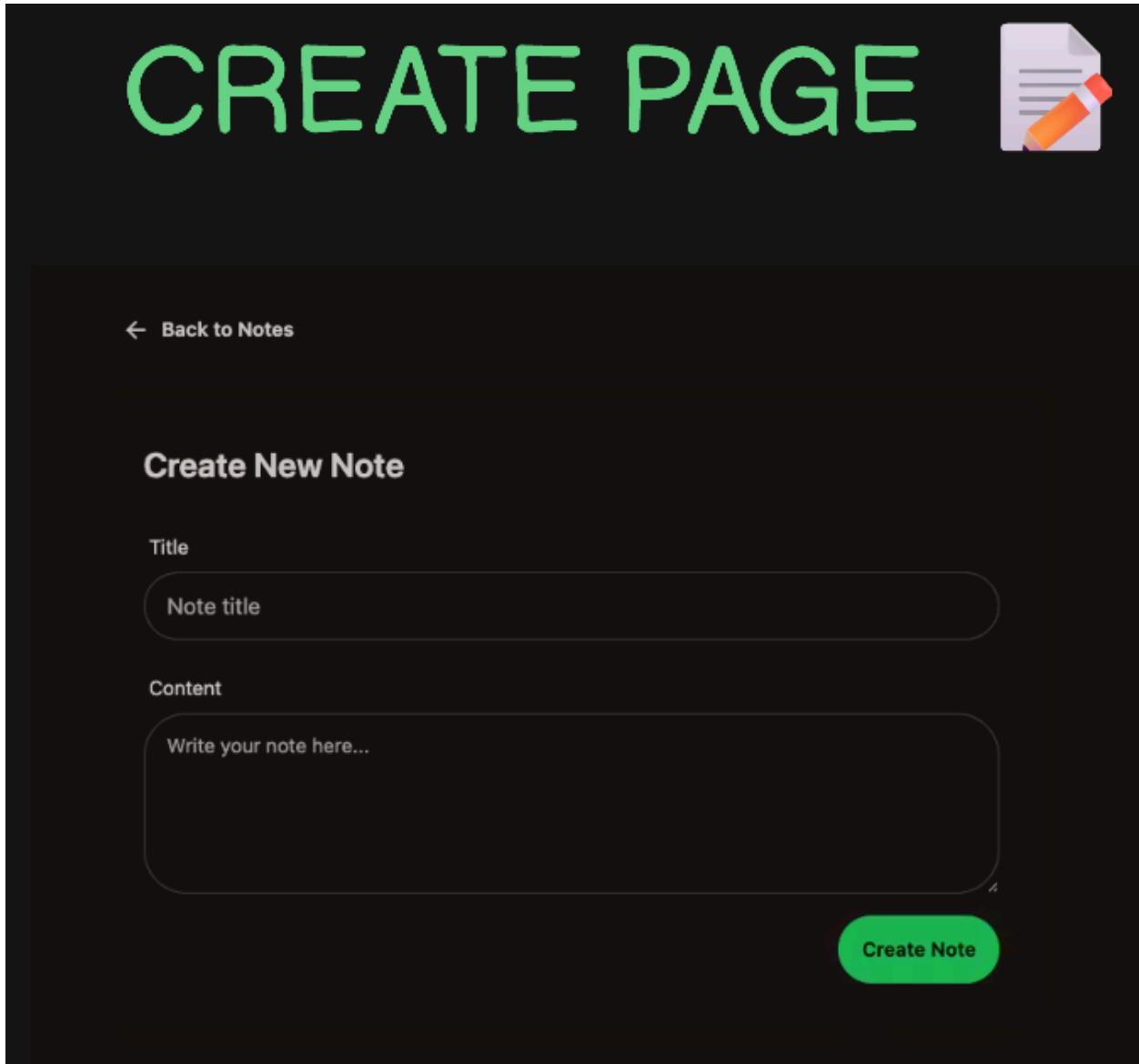
Send request:



Got response:



For creating Notes:



```
export async function createNote(req, res) {  
    // If user want to create a notes:  
    // includes title, content  
    try {  
        // title and content comes from req.body  
        const {title, content} = req.body  
        // by default we can't access this value  
        // To access them(or console it),  
        // Go to server.js, just before the routes  
        // add the code -> app.use(express.json())  
        // which is a middleware that we add  
        console.log(title, content)  
    } catch (error) {  
        res.status(500).json({  
            error: "An error occurred while creating the note."  
        })  
    }  
}
```

```
    } catch (error) {  
        }  
    }  
}
```

In server.js:

```
connectDB()
```

```
// middleware  
app.use(express.json())
```

```
app.use("/api/notes", notesRoutes)
```

Create note function:

```
export async function createNote(req, res) {  
    // If user want to create a notes:  
    // includes title, content  
    try {  
        // title and content comes from req.body  
        const {title, content} = req.body  
        // by default we can't access this value  
        // To access them(or console it),  
        // Go to server.js, just before the routes  
        // add the code -> app.use(express.json())  
        // which is a middleware that we add  
        // console.log(title, content)  
  
        // const newNote = new Note({title:title, content: content})  
        // Since key and value are the same, so the above code can  
        // be replaced as  
        const newNote = new Note({title, content})  
  
        await newNote.save()  
        res.status(201).json({message: "Note created Successfully"})  
    } catch (error) {  
        console.error("Error in createNote controller", error)  
        res.status(500).json({message: "Internal Server Error"})  
    }  
}
```

}

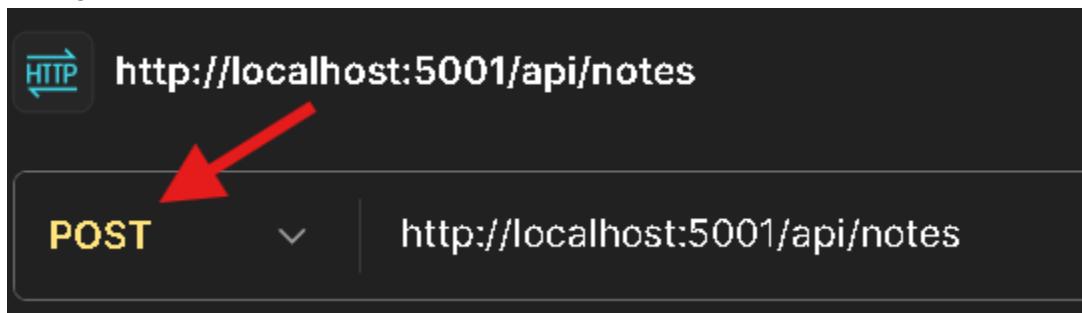
Since `createNotes` is the post request:

```
router.get("/", getAllNotes)
router.post("/", createNote)
router.put("/:id", updateNote)
router.delete("/:id", deleteNote)
```

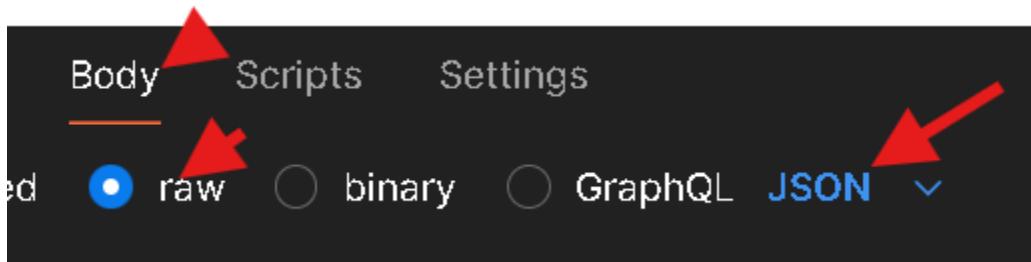
Lets see it in action:

In postman:

Change it to `post`:



Go to body, make to raw, change to JSON:



Write title and content:

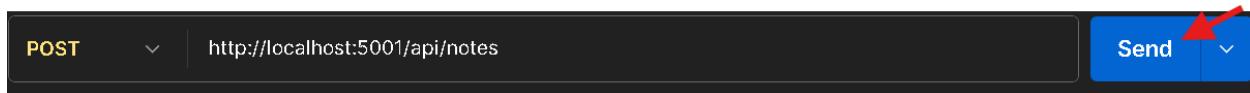
```
{
  "title": "my first note",
  "content": "some content"
}
```

POST http://localhost:5001/api/notes

Body (raw) JSON

```
{ "title": "my first note", "content": "some content" }
```

Send post request:



Got response:

201 Created

{ } JSON

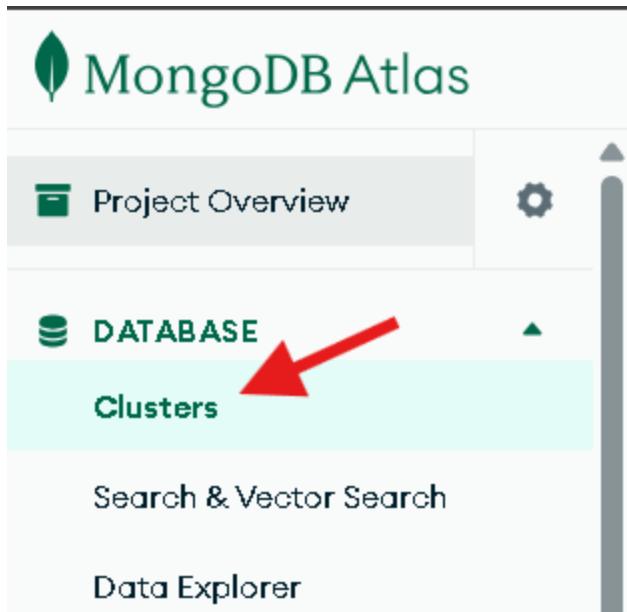
```
{ "message": "Note created Successfully" }
```

As per the code:

```
async function createNote(req, res) {
  await newNote.save()
  res.status(201).json({ message: "Note created Successfully" })
}
```

Check database:

Login using google.



Clusters

The screenshot shows the "Cluster0" cluster details page. At the top, there's a search bar with the placeholder "Find a database deployment...". Below it, the cluster status is shown as "● Cluster0" (green dot) followed by "Connect", "View Monitoring", "Browse Collections" (which has a red arrow pointing to it), and "...".

Below this, a message says "Wait to connect to Cluster0". The cluster name is "Cluster0".

The main content area shows the database structure:

- Database name**: admin
- Collection name**: notes
- Databases**: local
- Collections**: notes_db

Red arrows point to the "Clusters" button in the sidebar, the "Browse Collections" button, the "notes_db" collection, and the "notes" collection.

Then in the **notes** collection has:

Cluster0 > notes_db > notes

Documents 1 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#) *

 ADD DATA

 UPDATE

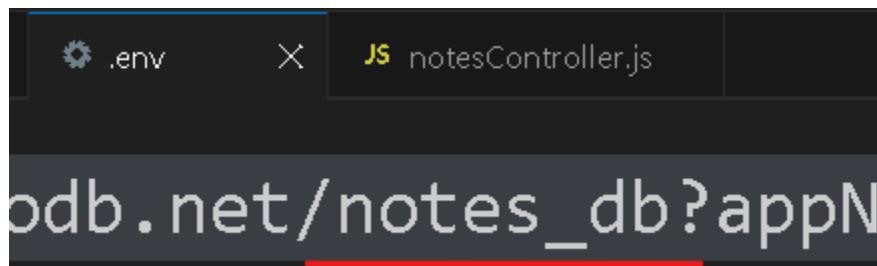
 DELETE

 EXPORT CODE

```
_id: ObjectId('698b417b5b1032b4854da5f2')
title: "my first note"
content: "some content"
createdAt: 2026-02-10T14:32:27.504+00:00
updatedAt: 2026-02-10T14:32:27.504+00:00
__v: 0
```

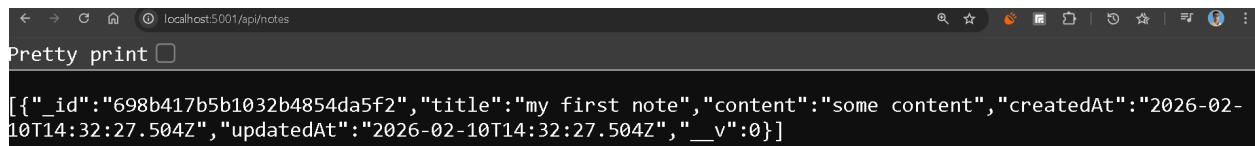
MongoDB has created **_id, createdAt and updatedAt**.

notes_db is actually comes from **.env** file, when setting up



If you have not provided, then database is named as **test**.

In the URL: <http://localhost:5001/api/notes>



Using postman, Get request:

Click send to get response

The screenshot shows a Postman interface with a GET request to `http://localhost:5001/api/notes`. The response body is a JSON array containing one note object:

```
1  [
2    {
3      "_id": "698b417b5b1032b4854da5f2",
4      "title": "my first note",
5      "content": "some content",
6      "createdAt": "2026-02-10T14:32:27.504Z",
7      "updatedAt": "2026-02-10T14:32:27.504Z",
8      "__v": 0
9    }
10 ]
```

Code for createNote function:

```
export async function createNote(req, res) {
  // If user want to create a notes:
  // includes title, content
  try {
    // title and content comes from req.body
    const {title, content} = req.body
    // by default we can't access this value
    // To access them(or console it),
    // Go to server.js, just before the routes
    // add the code -> app.use(express.json())
    // which is a middleware that we add
    // console.log(title, content)

    // const newNote = new Note({title:title, content: content})
```

```

// Since key and value are the same, so the above code can
// be replaced as
// const newNote = new Note({title, content})
// await newNote.save()

const newNote = new Note({title, content})
await newNote.save()
res.status(201).json({message: "Note Created successfully!"})

//res.status(201).json({message: "Note created Successfully"})
} catch (error) {
  console.error("Error in createNote controller", error)
  res.status(500).json({message: "Internal Server Error"})
}
}

```

In postman:

POST Send

Docs Params Authorization Headers (9) Body Scripts Settings Cookies

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

```

1  {
2    "title": "my first note",
3    "content": "some content 1"
4  }

```

Response:

Body 201 Created

{ } JSON Preview Visualize

```

1  {
2    "message": "Note Created successfully!"
3  }

```

As per the code:

```

async function createNote(req, res) {
  const newNote = new Note({title, content})
  await newNote.save()
  res.status(201).json({message: "Note Created successfully!"})
}

```

If we want informational response from postman instead of:

The screenshot shows a Postman request response. At the top right, it says "201 Created". Below that, under "Body", there is a JSON response:


```

    {
      "message": "Note Created successfully!"
    }
    
```

Which is not providing more information.

Code for *createNote* function:

```

export async function createNote(req, res) {
  // If user want to create a notes:
  // includes title, content
  try {
    // title and content comes from req.body
    const {title, content} = req.body
    // by default we can't access this value
    // To access them(or console it),
    // Go to server.js, just before the routes
    // add the code -> app.use(express.json())
    // which is a middleware that we add
    // console.log(title, content)

    // const newNote = new Note({title:title, content: content})
    // Since key and value are the same, so the above code can
    // be replaced as
    // const newNote = new Note({title, content})
    // await newNote.save()

    // const newNote = new Note({title, content})
    // await newNote.save()
    // res.status(201).json({message: "Note Created successfully!"})

    const note = new Note({title, content})
    const savedNote = await note.save()
    res.status(201).json(savedNote)

  } catch (error) {
    console.error("Error in createNote controller", error)
    res.status(500).json({message: "Internal Server Error"})
  }
}
    
```

In postman:

POST ▼ http://localhost:5001/api/notes

Docs Params Authorization Headers (9) Body ● Scripts Settings Cookies

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

```

1  {
2    "title": "my second note",
3    "content": "some content 2"
4 }
```

Response:

Body ▼ ⟳ 201 Created

{ } JSON ▼ Preview Visualize ▼

```

1  {
2    "title": "my second note",
3    "content": "some content 2",
4    "_id": "698c0c048123dba253ad879c",
5    "createdAt": "2026-02-11T04:56:36.167Z",
6    "updatedAt": "2026-02-11T04:56:36.167Z",
7    "__v": 0
8 }
```

In terminal:

```
[nodemon] restarting due to changes...
[nodemon] starting `node src/server.js`
Server started on PORT: 5001
MongoDB connected Successfully...
[ ]
```

In MongoDB(refresh it):

notes +

Cluster0 > notes_db > notes View monitoring

Documents 0 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#) Explain Rese

ADD DATA UPDATE DELETE EXPORT CODE 25 ▾ 1 - 0 of 0 ⟳ < >

Then:

```
_id: ObjectId('698c0c048123dba253ad879c')
title: "my second note"
content: "some content 2"
createdAt: 2026-02-11T04:56:36.167+00:00
updatedAt: 2026-02-11T04:56:36.167+00:00
__v: 0
```

Concept of Updating:

In [server.js](#):

```
app.use("/api/notes", notesRoutes)
```

In [notesRoutes.js](#):

```
7 router.put("/:id", updateNote)
```

Example for updating, Content:

```
_id: ObjectId('698c0eca8123dba253ad87a0')
title: "my third note"
content: "some content 3"
createdAt: 2026-02-11T05:08:26.101+00:00
updatedAt: 2026-02-11T05:08:26.101+00:00
__v: 0
```

In postman, it must be like

```
PUT http://localhost:5001/api/notes/698c0eca8123dba253ad87a0
```

Code in [notesRoutes.js](#):

```
7 router.put("/:id", updateNote)
```

and in [notesController.js](#) update function:

```
await Note.findByIdAndUpdate(req.params.id)
```

Must have same word.

Code for updateNote Function:

```
export async function updateNote(req, res) {
```

```

try {
  const {title, content} = req.body
  // How do we know id that user sends, so that we can update
  // based on id
  // {title, content} <- things to update
  await Note.findByIdAndUpdate(req.params.id, {title, content})
  res.status(200).json({message: "Note updated successfully."})

} catch (error) {
  console.error("Error in updateNote controller", error)
  res.status(500).json({message: "Internal Server Error"})
}
}

```

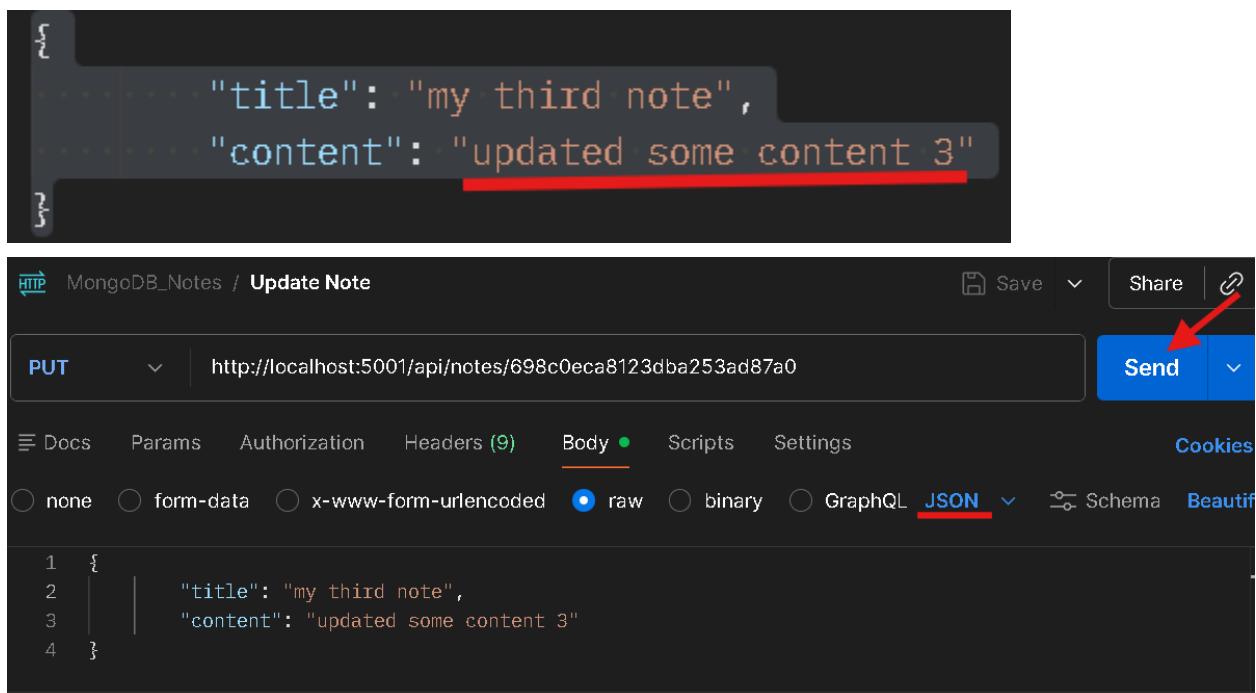
Initially in MongoDB:

```

_id: ObjectId('698c0eca8123dba253ad87a0')
title: "my third note"
content: "some content 3"
createdAt: 2026-02-11T05:08:26.101+00:00
updatedAt: 2026-02-11T05:08:26.101+00:00
__v: 0

```

In Postman:



The screenshot shows the Postman interface for a MongoDB Notes API. The request method is PUT, and the URL is `http://localhost:5001/api/notes/698c0eca8123dba253ad87a0`. The body is set to raw JSON, containing the following data:

```

1  {
2    |   "title": "my third note",
3    |   "content": "updated some content 3"
4  }

```

A red arrow points to the "Send" button in the top right corner of the request panel.

Response:

Body 200 OK

{ } JSON ▾ Preview Visualize | ▾

```

1  {
2   |   "message": "Note updated successfully."
3   }

```

Refresh DB:



In MongoDB:

```

_id: ObjectId('698c0eca8123dba253ad87a0')
title: "my third note"
content: "updated some content 3"
createdAt: 2026-02-11T05:08:26.101+00:00
updatedAt: 2026-02-11T06:18:38.733+00:00
__v: 0

```

If user provides invalid ID, during Updating:

To get fields after update.

was **before** `update` was applied. If you set `new: true`, `findOneAndUpdate()` will instead give you the object after `update` was applied.

Code for updateNote Function:

```

export async function updateNote(req, res) {
  try {
    const {title, content} = req.body
    // How do we know id that user sends, so that we can update
    // based on id
    // {title, content} <- things to update
  }
}

```

```

// {new: true} <- to get fields
const updatedNote = await Note.findByIdAndUpdate(req.params.id, {title, content}, {new: true})
// For false value, 404=not found,
if(!updatedNote) return res.status(404).json({message: "Note not found!"})
res.status(200).json({message: "Note updated successfully."})

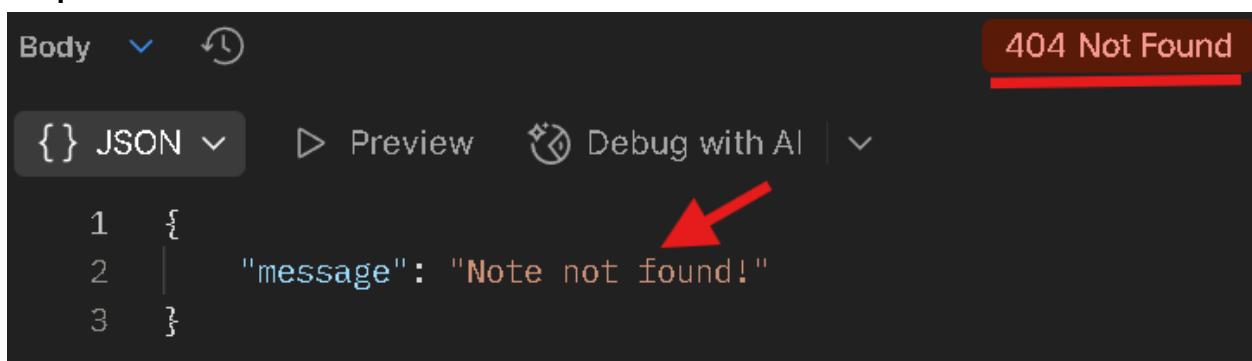
} catch (error) {
  console.error("Error in updateNote controller", error)
  res.status(500).json({message: "Internal Server Error"})
}
}

```

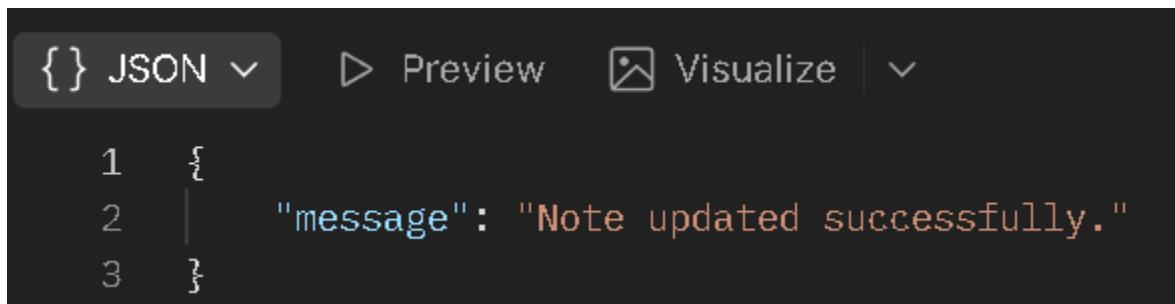
Now in postman:



Response:



For successful update:



To change the response:

Code for updateNote Function:

```

export async function updateNote(req, res) {
  try {
    const {title, content} = req.body
    // How do we know id that user sends, so that we can update
  }
}

```

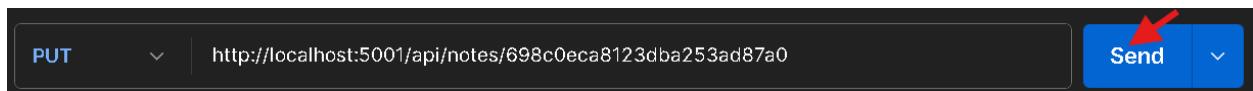
```

// based on id
// {title, content} <- things to update
// {new: true} <- to get fields
const updatedNote = await Note.findByIdAndUpdate(req.params.id, {title, content}, {new: true})
  // For false value, 404=not found,
  if(!updatedNote) return res.status(404).json({message: "Note not found!"})
  // res.status(200).json({message: "Note updated successfully."})
res.status(200).json(updatedNote)

} catch (error) {
  console.error("Error in updateNote controller", error)
  res.status(500).json({message: "Internal Server Error"})
}
}

```

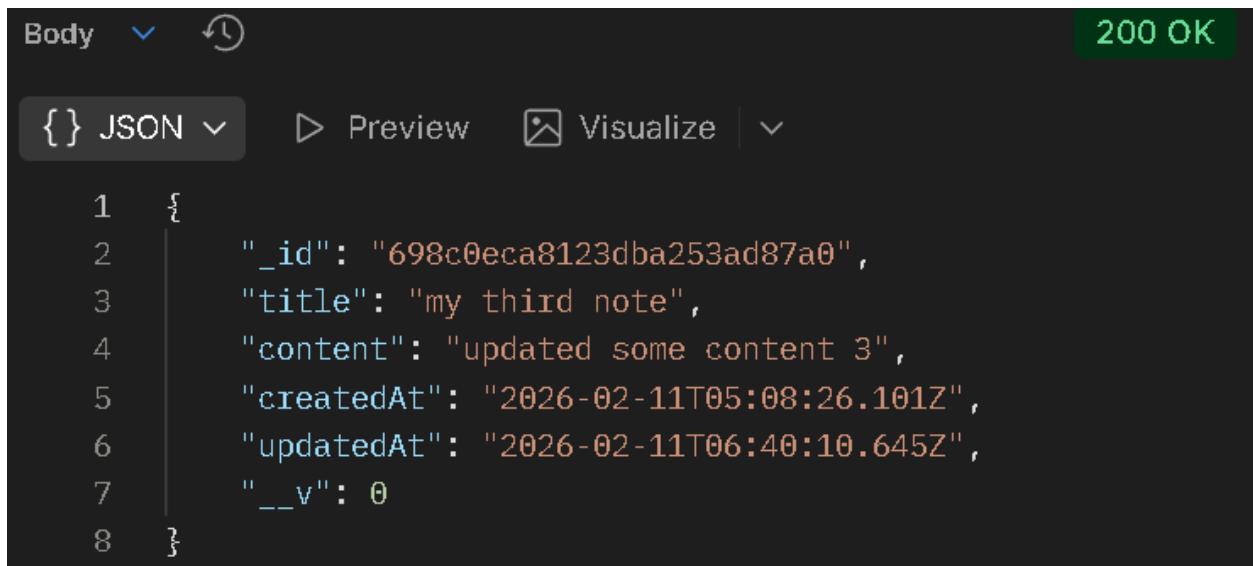
In postman:



Body, json, raw having:

```
{
  "title": "my third note",
  "content": "updated some content 3"
}
```

Response:



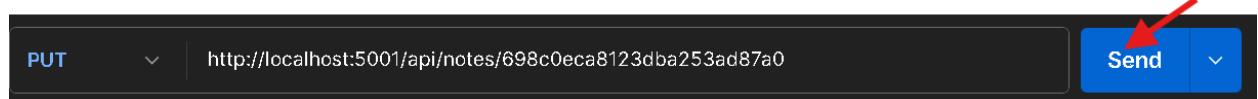
What if title is not provided(JSON Type):

Docs Params Authorization Headers (9) Body Scripts Settings

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

```
1 {  
2 |   "content": "updated some content 123"  
3 }
```

Send it:



Response:

Body JSON Preview Visualize 200 OK

```
{ } JSON ▾
```

```
1 {  
2   "_id": "698c0eca8123dba253ad87a0",  
3   "title": "my third note",  
4   "content": "updated some content 123",  
5   "createdAt": "2026-02-11T05:08:26.101Z",  
6   "updatedAt": "2026-02-11T06:44:47.394Z",  
7   "__v": 0  
8 }
```

Refresh in MongoDB:



```
_id: ObjectId('698c0eca8123dba253ad87a0')  
title: "my third note"  
content: "updated some content 123"  
createdAt: 2026-02-11T05:08:26.101+00:00  
updatedAt: 2026-02-11T06:44:47.394+00:00  
__v: 0
```

For Deleting the Notes:

In server.js:



```
JS db.js JS notesRoutes.js JS server.js X JS .env JS notes
JS server.js > ...
app.use("/api/notes", notesRoutes)
```

In notesRoutes.js:



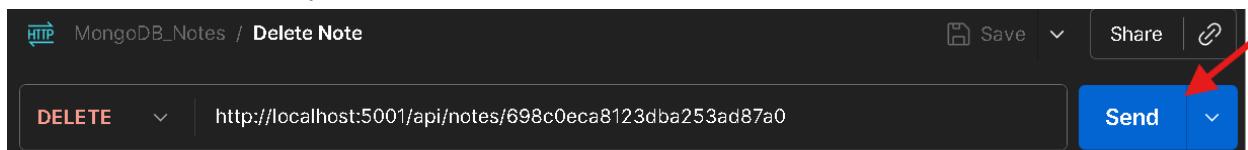
```
JS db.js JS notesRoutes.js X JS server.js JS .env JS notes
routes > JS notesRoutes.js > ...
router.delete("/:id", deleteNote)
```

Code in deleteNote:

```
export async function deleteNote(req, res) {
  try {
    const deletedNote = await Note.findByIdAndDelete(req.params.id)
    if (!deletedNote) return res.status(404).json({message: "Note not found!"})
    // By default status=200
    res.status(200).json({message: "Note deleted successfully!"})
  } catch (error) {
    console.error("Error in deleteNote controller", error)
    res.status(500).json({message: "Internal Server Error"})
  }
}
```

In postman:

No need to provide body



Response:

Body   200 OK

{ } JSON  Preview  Visualize 

```
1 {  
2   |   "message": "Note deleted successfully!"  
3 }
```

In MongoDB:

Refresh:

3 of 3



notes +

Cluster0 > notes_db > notes

Documents 3 Aggregations Schema Indexes 1

Type a query: { field: 'value' } or [Generate query](#)

[ADD DATA](#) [UPDATE](#) [DELETE](#) [EXPORT CODE](#)

```
_id: ObjectId('698c0ab3f540242a5abb0379')
title: "my first note"
content: "some content 1"
createdAt: 2026-02-11T04:50:59,240+00:00
updatedAt: 2026-02-11T04:50:59,240+00:00
__v: 0
```

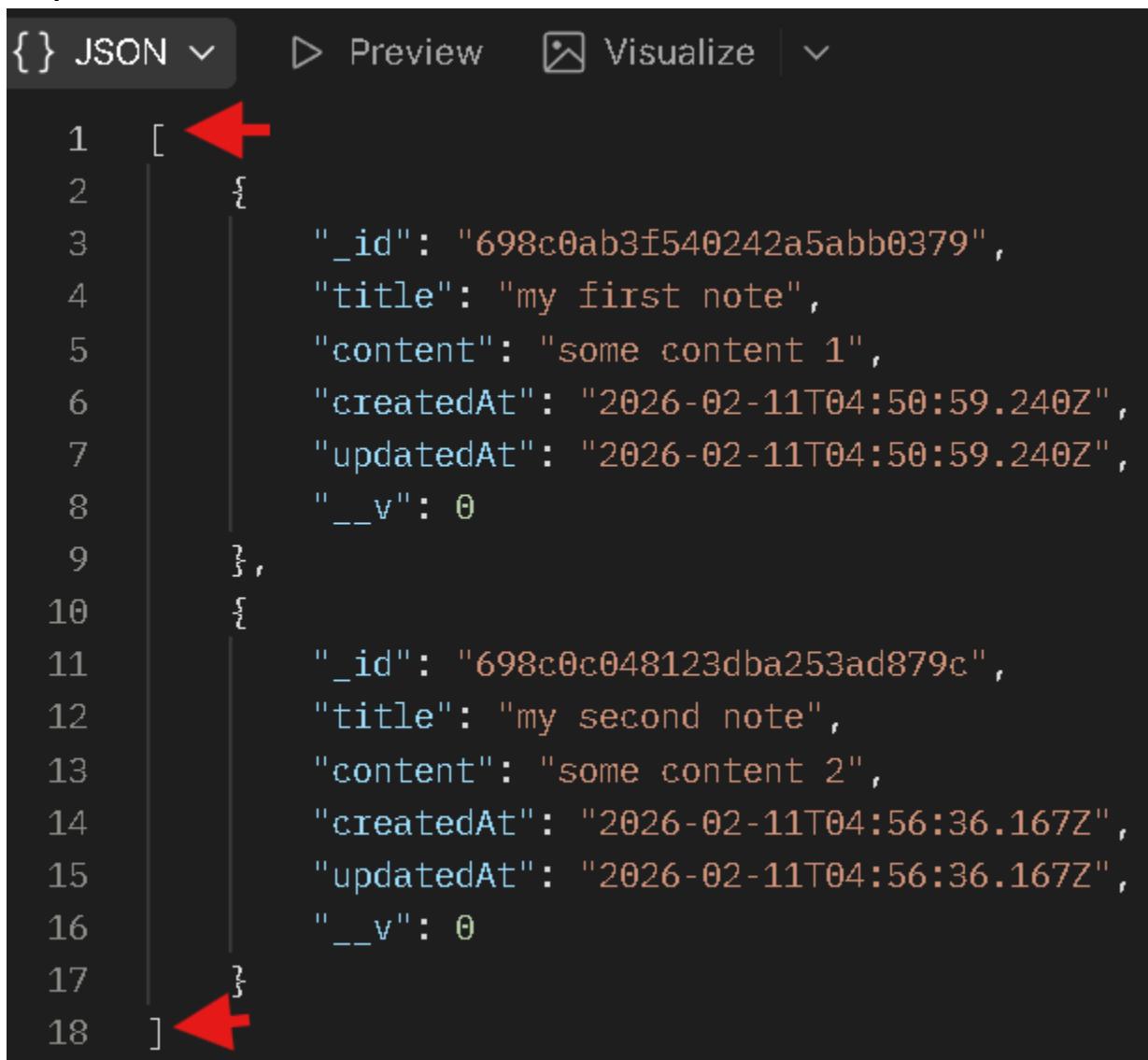
```
_id: ObjectId('698c0c048123dba253ad879c')
title: "my second note"
content: "some content 2"
createdAt: 2026-02-11T04:56:36,167+00:00
updatedAt: 2026-02-11T04:56:36,167+00:00
__v: 0
```

Third not visible.

In postman, get request:



Response:



```
{ } JSON ▾ ▶ Preview ⚡ Visualize ▾
```

```
1 [ ←
2 { ←
3     "_id": "698c0ab3f540242a5abb0379",
4     "title": "my first note",
5     "content": "some content 1",
6     "createdAt": "2026-02-11T04:50:59.240Z",
7     "updatedAt": "2026-02-11T04:50:59.240Z",
8     "__v": 0
9 },
10 {
11     "_id": "698c0c048123dba253ad879c",
12     "title": "my second note",
13     "content": "some content 2",
14     "createdAt": "2026-02-11T04:56:36.167Z",
15     "updatedAt": "2026-02-11T04:56:36.167Z",
16     "__v": 0
17 }
18 ] ←
```

To fetch the user based on ID:

In notesRoutes.js:

```
ge.json      JS db.js      JS notesRoutes.js X      JS server.js      .env      J
> src > routes > JS notesRoutes.js > ...
5 router.get("/", getAllNotes)
6 router.get("/:id", getNoteById)
7 router.post("/", createNote)
```

Also import it:

So finally [notesRouters.js](#) has:

```
import express from "express"
import { getAllNotes, getNoteById, createNote, updateNote, deleteNote } from
"../controllers/notesController.js"
const router = express.Router()
```

```
router.get("/", getAllNotes)
router.get("/:id", getNoteById)
router.post("/", createNote)
router.put("/:id", updateNote)
router.delete("/:id", deleteNote)
```

```
export default router;
```

Now let's create its controller.

Code for `getNoteById` function:

```
export async function getNoteById(req, res) {
  try {
    const note = await Note.findById(req.params.id)
    if(!note) return res.status(404).json({message: "Note not Found!"})
    res.json(note)
  } catch (error) {
    console.error("Error in getNoteById controller", error)
    res.status(500).json({message: "Internal Server Error"})
  }
}
```

If in postman, if you get error:



Could not send request

! Error: connect ECONNREFUSED 127.0.0.1:5001

Debug with AI

Solution is: run the server

```
PS C:\Users\kiran\OneDrive\Desktop\mern-thinkboard> cd .\backend\  
PS C:\Users\kiran\OneDrive\Desktop\mern-thinkboard\backend> npm run dev  
  
> backend@1.0.0 dev  
> nodemon src/server.js  
  
[nodemon] 3.1.11  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,cjs,json  
[nodemon] starting `node src/server.js`  
Server started on PORT: 5001  
MongoDB connected Successfully...  
□
```

Copy the id, from **get Request from notes**.

HTTP MongoDB_Notes / **Read Notes**

GET http://localhost:5001/api/notes

Docs Params Authorization Headers (7) Body S

Body { } JSON ▶ Preview Visualize

```
1 [  
2 {  
3   "_id": "698c0ab3f540242a5abb0379",  
4   "title": "my first note",  
5   "content": "some content 1",  
6   "createdAt": "2026-02-11T04:50:59.240Z",  
7   "updatedAt": "2026-02-11T04:50:59.240Z",  
8   "__v": 0  
9 },  
10 {  
11   "_id": "698c0c048123dba253ad879c",  
12   "title": "my second note",  
13   "content": "some content 2",  
14   "createdAt": "2026-02-11T04:56:36.167Z",  
15   "updatedAt": "2026-02-11T04:56:36.167Z",  
16   "__v": 0  
17 }  
18 ]
```

Paste to get note by ID:



Response:

```
{  
  "_id": "698c0ab3f540242a5abb0379",  
  "title": "my first note",  
  "content": "some content 1",  
  "createdAt": "2026-02-11T04:50:59.240Z",  
  "updatedAt": "2026-02-11T04:50:59.240Z",  
  "__v": 0  
}
```

To sort all the notes:

Like newest first.

```
Ion getAllNotes(req, res) {  
  
  nd(), to get every single note  
  eatedAt: -1}) <= newest first  
  s = await Note.find().sort({createdAt: -1})
```

Then in postman:

HTTP MongoDB_Notes / **Read Notes**

GET http://localhost:5001/api/notes

Docs Params Authorization Headers (7) Body Sc

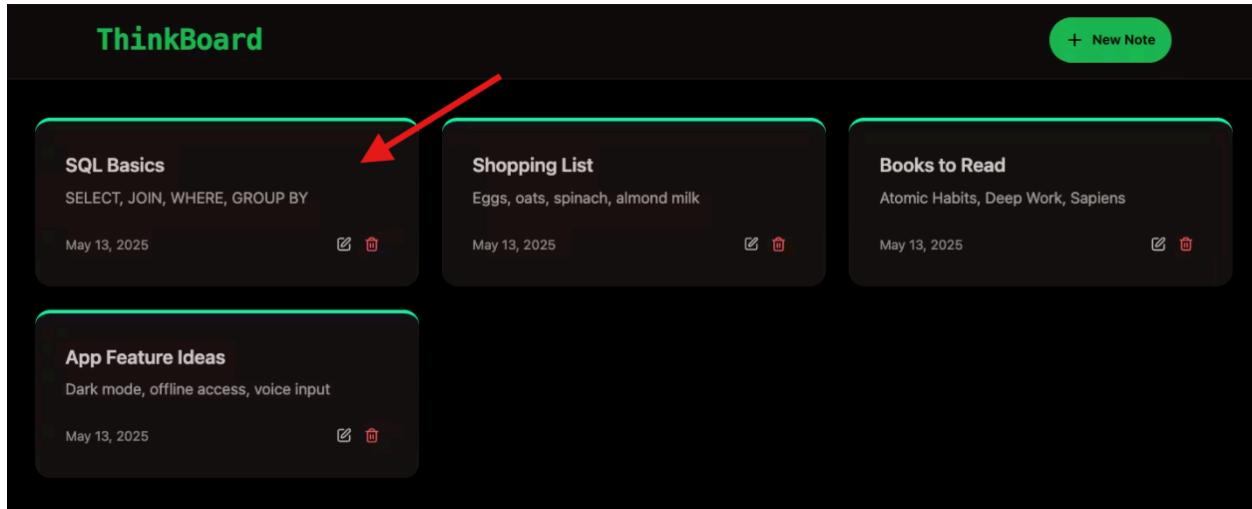
Body ↻

{ } JSON ▾ Preview Visualize ▾

```
1   [
2     {
3       "_id": "698c0c048123dba253ad879c",
4       "title": "my second note",
5       "content": "some content 2",
6       "createdAt": "2026-02-11T04:56:36.167Z",
7       "updatedAt": "2026-02-11T04:56:36.167Z",
8       "__v": 0
9     },
10    {
11      "_id": "698c0ab3f540242a5abb0379",
12      "title": "my first note",
13      "content": "some content 1",
14      "createdAt": "2026-02-11T04:50:59.240Z",
15      "updatedAt": "2026-02-11T04:50:59.240Z",
16      "__v": 0
17    }
18  ]
```

So **my second note(title)** has come to the first place.

So that I can get the latest note at the beginning:



Code in server.js:

```
import express from "express"
import notesRoutes from "./routes/notesRoutes.js"
import { connectDB } from "./config/db.js"

import dotenv from "dotenv"
dotenv.config()

//console.log(process.env.MONGO_URI)

const app = express()
// if process.env.PORT is undefined then PORT = 5001(by default value)
const PORT = process.env.PORT || 5001

connectDB()

// middleware: are used to control title and content
// instead of providing it from postman
app.use(express.json())

app.use("/api/notes", notesRoutes)

app.listen(PORT, () => {
  console.log("Server started on PORT:", PORT)
})
```

Code in notesRoutes.js:

```
import express from "express"
import { getAllNotes, getNoteById, createNote, updateNote, deleteNote } from
"../controllers/notesController.js"
const router = express.Router()

router.get("/", getAllNotes)
router.get("/:id", getNoteById)
router.post("/", createNote)
router.put("/:id", updateNote)
router.delete("/:id", deleteNote)

export default router;
```

Code in notesController.js:

```
import Note from "../models>Note.js"

export async function getAllNotes(req, res) {
  try {
    // Note.find(), to get every single note
    // sort({createdAt: -1}) <= newest first
    // by default createdAt: 1
    const notes = await Note.find().sort({createdAt: -1})
    // Send status, send notes as json
    res.status(200).json(notes)
  } catch (error) {
    // For debugging purpose
    console.error("Error in getAllNotes controller", error)
    res.status(500).json({message: "Internal Server Error"})
  }
}

export async function getNoteById(req, res) {
  try {
    const note = await Note.findById(req.params.id)
    if(!note) return res.status(404).json({message: "Note not Found!"})
    res.json(note)
  } catch (error) {
    console.error("Error in getNoteById controller", error)
    res.status(500).json({message: "Internal Server Error"})
  }
}
```

```

export async function createNote(req, res) {
    // If user want to create a notes:
    // includes title, content
    try {
        // title and content comes from req.body
        const {title, content} = req.body
        // by default we can't access this value
        // To access them(or console it),
        // Go to server.js, just before the routes
        // add the code -> app.use(express.json())
        // which is a middleware that we add
        // console.log(title, content)

        // const newNote = new Note({title:title, content: content})
        // Since key and value are the same, so the above code can
        // be replaced as
        // const newNote = new Note({title, content})
        // await newNote.save()

        // const newNote = new Note({title, content})
        // await newNote.save()
        // res.status(201).json({message: "Note Created successfully!"})

        const note = new Note({title, content})
        const savedNote = await note.save()
        res.status(201).json(savedNote)

    } catch (error) {
        console.error("Error in createNote controller", error)
        res.status(500).json({message: "Internal Server Error"})
    }
}

export async function updateNote(req, res) {
    try {
        const {title, content} = req.body
        // How do we know id that user sends, so that we can update
        // based on id
        // {title, content} <- things to update
        // {new: true} <- to get fields
        const updatedNote = await Note.findByIdAndUpdate(req.params.id, {title, content}, {new: true})
        // For false value, 404=not found,
    }
}

```

```
    if(!updatedNote) return res.status(404).json({message: "Note not found!"})
    // res.status(200).json({message: "Note updated successfully."})
    res.status(200).json(updatedNote)

} catch (error) {
    console.error("Error in updateNote controller", error)
    res.status(500).json({message: "Internal Server Error"})
}
}

export async function deleteNote(req, res) {
    try {
        const deletedNote = await Note.findByIdAndDelete(req.params.id)
        if (!deletedNote) return res.status(404).json({message: "Note not found!"})
        res.status(200).json({message: "Note deleted successfully!"})
    } catch (error) {
        console.error("Error in deleteNote controller", error)
        res.status(500).json({message: "Internal Server Error"})
    }
}
```