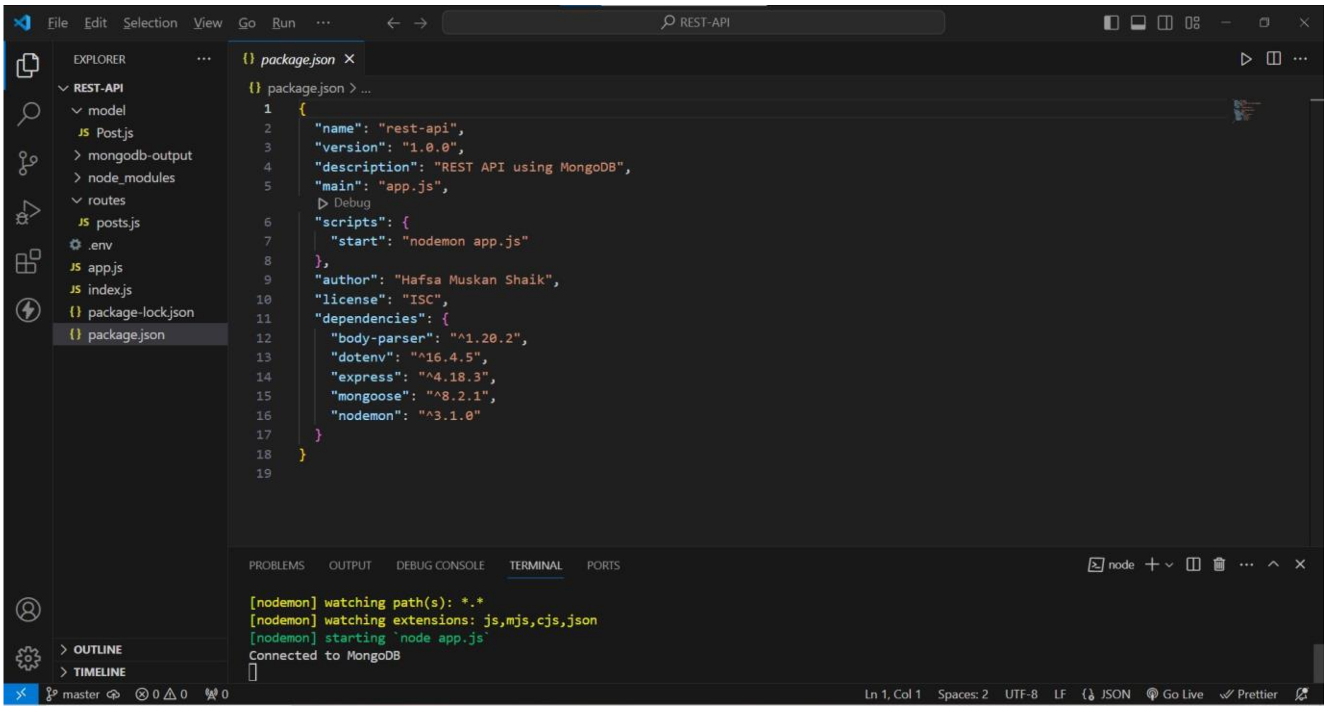
Creating a RESTful API using express.js and creating a database and index in MongoDB.

Name	: Shaik. Hafsa Muskan
Email	: 208x1a0587@khitguntur.ac.in
Phone no	: 9030628268
Roll No	: 208x1a0587
College	: Kallam Haranadhareddy Institute of Technology

Source code:

Package.json file:

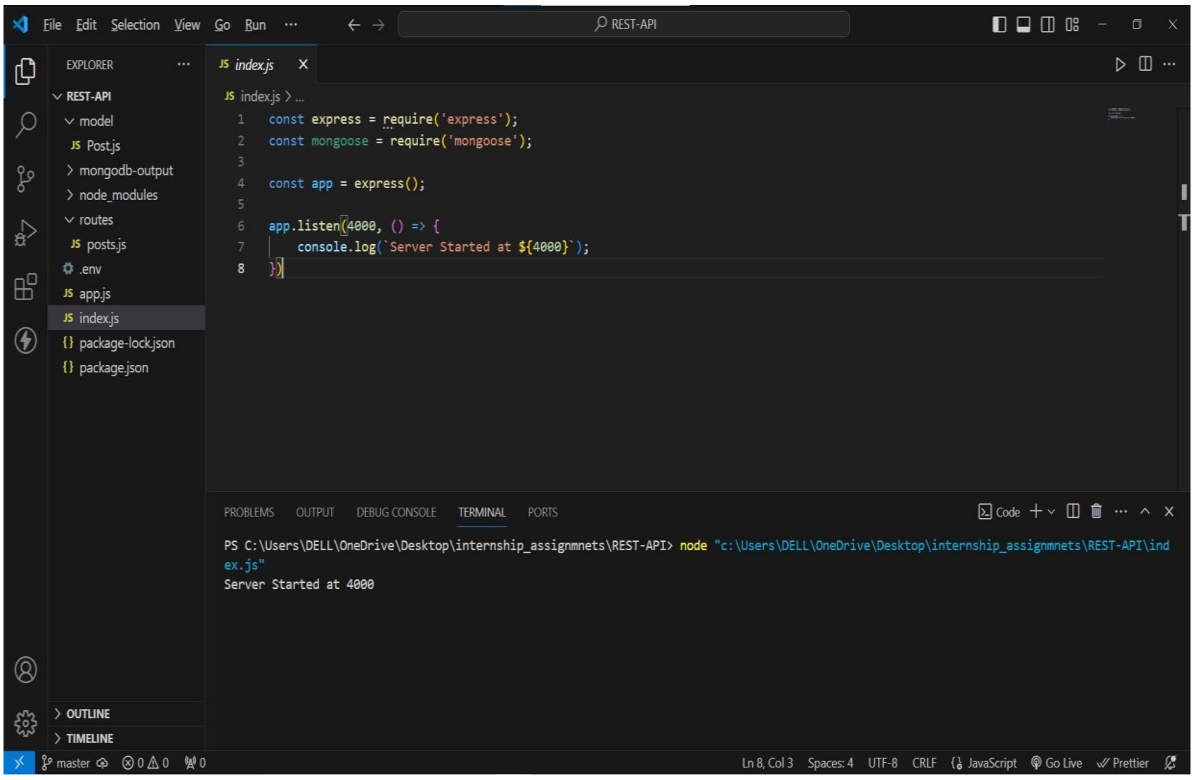
In this file change the scripts to start the nodemon server. It is used to automate the process such as saving and restarting the server.



```json
{
  "name": "rest-api",
  "version": "1.0.0",
  "description": "REST API using MongoDB",
  "main": "app.js",
  "scripts": {
    "start": "nodemon app.js"
  },
  "author": "Hafsa Muskan Shaik",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.20.2",
    "dotenv": "^16.4.5",
    "express": "^4.18.3",
    "mongoose": "^8.2.1",
    "nodemon": "^3.1.0"
  }
}
```

```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node app.js`
Connected to MongoDB
```

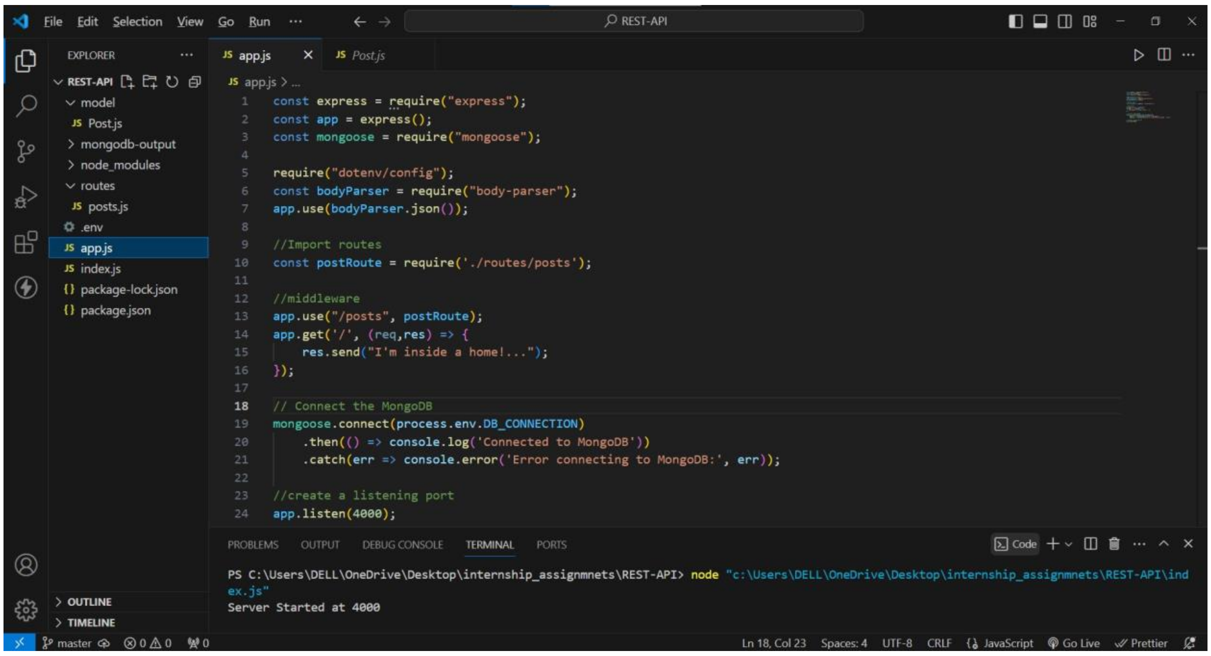
index.js:



```
const express = require('express');
const mongoose = require('mongoose');

const app = express();

app.listen(4000, () => {
    console.log(`Server Started at ${4000}`);
})
```

```
PS C:\Users\DELL\OneDrive\Desktop\internship_assignmnets\REST-API> node "c:\Users\DELL\OneDrive\Desktop\internship_assignmnets\REST-API\index.js"
Server Started at 4000
```

app.js:

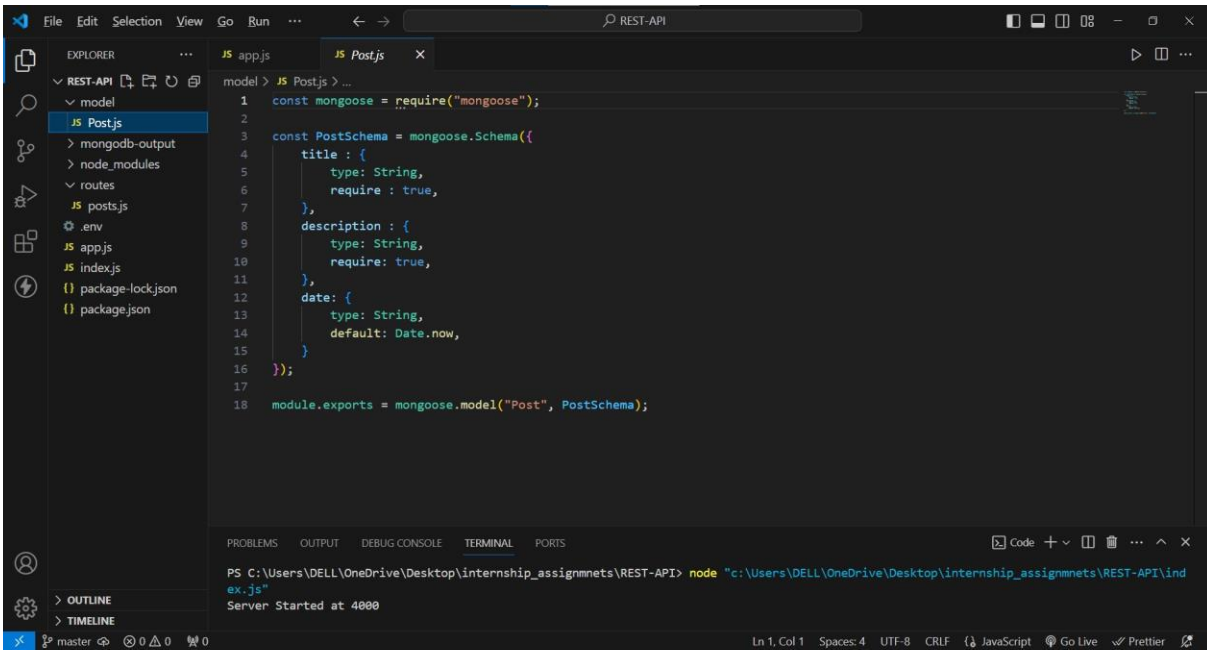


```
const express = require("express");
const app = express();
const mongoose = require("mongoose");

require("dotenv/config");
const bodyParser = require("body-parser");
app.use(bodyParser.json());

//Import routes
const postRoute = require('./routes/posts');

//middleware
app.use("/posts", postRoute);
app.get('/', (req,res) => {
    res.send("I'm inside a home!...");
});

// Connect the MongoDB
mongoose.connect(process.env.DB_CONNECTION)
    .then(() => console.log('Connected to MongoDB'))
    .catch(err => console.error('Error connecting to MongoDB:', err));

//create a listening port
app.listen(4000);
```

```
PS C:\Users\DELL\OneDrive\Desktop\internship_assignmnets\REST-API> node "c:\Users\DELL\OneDrive\Desktop\internship_assignmnets\REST-API\index.js"
Server Started at 4000
```

Model:

post.js file:

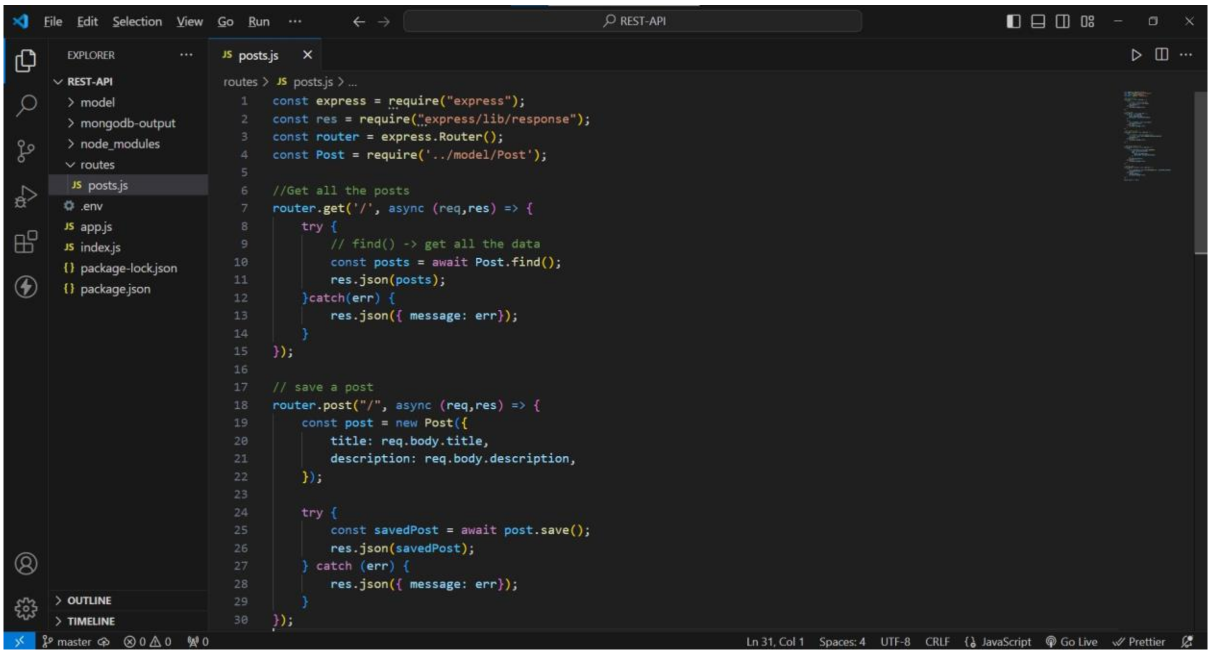


```
const mongoose = require("mongoose");

const PostSchema = mongoose.Schema({
    title : {
        type: String,
        require : true,
    },
    description : {
        type: String,
        require: true,
    },
    date: {
        type: String,
        default: Date.now,
    }
});

module.exports = mongoose.model("Post", PostSchema);
```

```
PS C:\Users\DELL\OneDrive\Desktop\internship_assignmnets\REST-API> node "c:\Users\DELL\OneDrive\Desktop\internship_assignmnets\REST-API\index.js"
Server Started at 4000
```

Routes:

posts.js file:



```
const express = require("express");
const res = require("express/lib/response");
const router = express.Router();
const Post = require('../model/Post');

//Get all the posts
router.get('/', async (req,res) => {
    try {
        // find() -> get all the data
        const posts = await Post.find();
        res.json(posts);
    }catch(err) {
        res.json({ message: err});
    }
});

// save a post
router.post("/", async (req,res) => {
    const post = new Post({
        title: req.body.title,
        description: req.body.description,
    });

    try {
        const savedPost = await post.save();
        res.json(savedPost);
    } catch (err) {
        res.json({ message: err});
    }
});
```

Steps to run locally:

1. Create a folder as any name.
2. Open that folder in any IDE (vs code).
3. Create a root folder: Here Restful-API is root folder.

4. In root folder create .env file and create a DB_CONNECITON variable and assign a value to it.

   - DB_CONNECITON = your mongodb_connection_string
   - Example: DB_CONNECTION="mongodb+srv://admin:admin@cluster0.xez5xvg.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0"

5. Open the terminal (ctrl + ~) in code editor.
6. Type the following commands to install dependencies and to run the server.

   - npm init: to initialize the project
   - npm i: to install all the dependencies.
   - npm start: to run server

   Note: Here I used nodemon to automate the process such as saving and restarting the server.

7. If you get the message without any errors in terminal then your server was running successfully.

Route and its functionality:

For this use any API client like Postman or Thunder Client.

Here I used Postman client API.

Step 1: Create Route:

1. This route is used to create a post in database with the following fields. title & description.

2. In postman create a collection and add new requests and select methods based on operation.

   - **get**: to get all the data: http://localhost:4000/posts/

   - **post:** to insert the data: http://localhost:4000/posts

3. Use post method to pass the following json data in a body as your required value.

```
{
        "title": "First post",
        "description": "my first post"
  }
```

4. After inserting the values click on send button and then use get method to get the data which is posted.

5. The data which is posted can automatically loaded into the MongoDB. Note: In this project I used cloud mongoDB (MongoDB Atlas).

Step 2: Perform CRUD Operations:

READONE:

1. This route is used to read specific post by passing the specific post id as a param.

use GET:URL: http://localhost:4000/posts/65ee0c657944e3ca6e3e1305

2. After sending you will get the specific post details as response.

READALL:

1. Read all route is used to get all the post data existing in the mongodb database.

   use GET(): URL: http://localhost:4000/posts/

2. After sending you will get all the post details as response.

UPDATE:

1. This route is used to update the post by passing the specific post id as a param.

   use PUT:
   URL: http://localhost:4000/posts/65ee0c657944e3ca6e3e1305

2. After sending you will get updated post details as response.

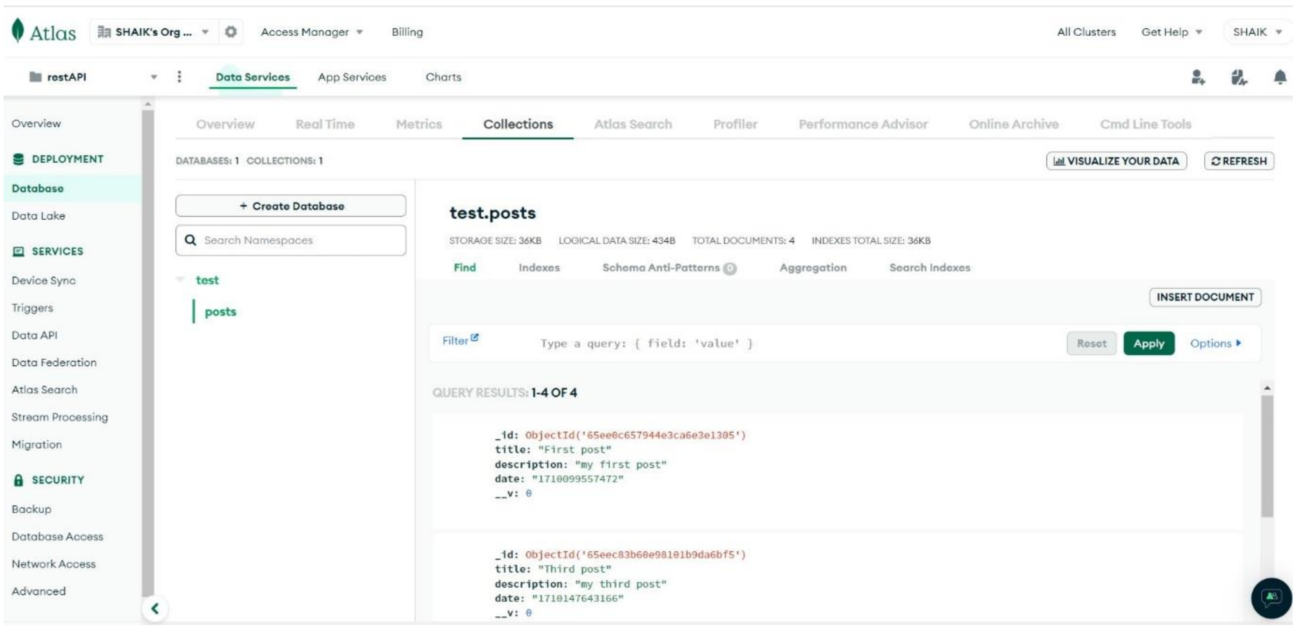
DELETE:

1. This route is used to delete specific post by passing the post id as a param.

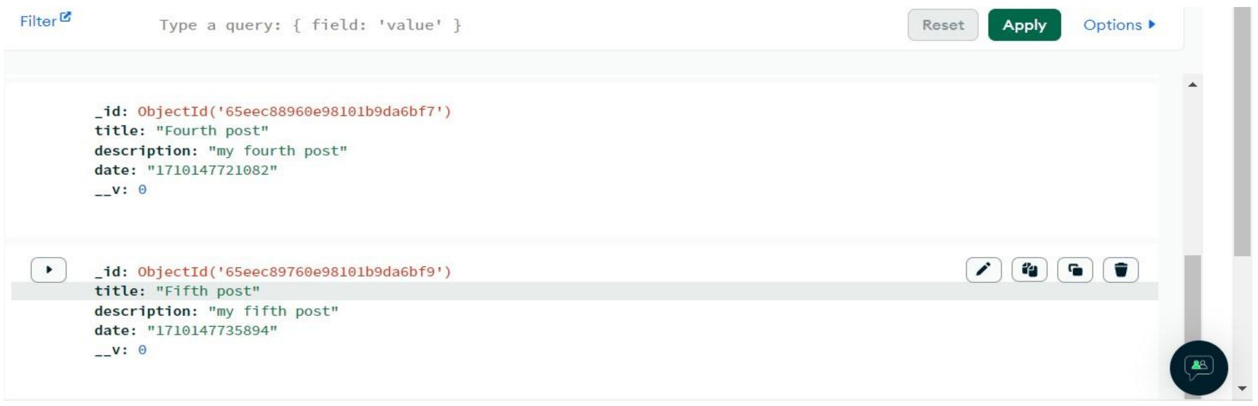
   use DELETE:
   URL: http://localhost:4000/posts/65ee0c657944e3ca6e3e1305

2. After sending the specific post will be deleted from the database.

Output:



```
_id: ObjectId('65ee0c657944e3ca6e3e1305')
title: "First post"
description: "my first post"
date: "1710099557472"
__v: 0

_id: ObjectId('65eec83b60e98101b9da6bf5')
title: "Third post"
description: "my third post"
date: "1710147643166"
__v: 0
```



```
_id: ObjectId('65eec88960e98101b9da6bf7')
title: "Fourth post"
description: "my fourth post"
date: "1710147721082"
__v: 0

_id: ObjectId('65eec89760e98101b9da6bf9')
title: "Fifth post"
description: "my fifth post"
date: "1710147735894"
__v: 0
```