Index.js

import dotenv from 'dotenv'

import './data/index.js'

import express from "express";

import articlesRouter from './routes/article.js';

import indexRouter from './routes/pages/index.js'

import articlesPageRouter from './routes/pages/article.js';

import HttpError from './utils/HttpError.js';

import fs from 'fs';

import path from 'path';

const app = express();

app.set( 'views', path.join( process.cwd(), 'src', 'views' ) );

app.set( 'view engine', 'ejs' );

// custom middleware

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

    const logMessage = `${req.method} ${req.url} at time ${new Date().toTimeString()}\n`;

    fs.writeFile( path.join( process.cwd(), 'src', 'server.log' ), logMessage, { encoding: 'utf-8', flag: 'a+' }, ( err ) => {

        if( err ) {

            console.log( err.message );

        }

        next(); // pass control to the next middleware

    });

});

app.use(express.static(path.join(process.cwd(),'src','public','bootstrap-5.1.3-dist')));

app.use(express.json({extended:true}));

app.use(express.urlencoded({extended:true}));

app.use('/articles',articlesRouter);

app.use( '/', indexRouter );

//app.use('/article/id',(req,res)=>console.log(req.params.id));

app.use( '/page/articles', articlesPageRouter );

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

    const httpError=new HttpError('Resource not found',404);

    next(httpError);

})

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

    res.status(httpError.statusCode||500).json({

        message:httpError.message

    })

})

dotenv.config();

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

app.listen(PORT, err => {

    if (err) {

        console.log(err.message);

        return;

    }

    console.log(`sucessfully started http://localhost:${PORT}`);

}

)

Controllers/article.js

import { fetchArticles, fetchArticlesById, addArticle, updateArticle, removeArticle,addComment ,fetchComments} from "../services/article.js";

import HttpError from '../utils/HttpError.js'

const getArticles = (req, res,next) => {

    let { sort, order, page, q } = req.query;

    let pageInt = parseInt(page);

    if (isNaN(pageInt)) {

        pageInt = 1;

    }

    if (!sort) {

        sort = 'title';

    }

    if (!order) {

        order = 'asc';

    }

    fetchArticles(sort, order,page, q)

        .then(articles=> {

            res.json(articles);

        })

        .catch(err => {

            const httpError = new HttpError(err.message, 500);

            next(httpError)

        });

}

const getArticlesById = (req, res, next) => {

    const { \_id } = req.params;

    fetchArticlesById(\_id)

        .then((article) => {

            if (!article) {

                const httpError = new HttpError('article with gievn id does not exists', 404)

                next(httpError);

                return;

            }

            res.json(article);

        })

        .catch((err) => {

            if (err.kind === 'ObjectId') {

                const httpError = new HttpError('Invalid article id', 400)

                next(httpError);

            } else {

                const httpError = new HttpError(err.message, 500)

                next(httpError)

            }

        });

}

const postArticle = (req, res, next) => {

    const { body } = req;

    // const { \_id } = req.params;

    console.log(Object.keys(body));

    if (Object.keys(body).length === 0) {

        const httpError = new HttpError('request body is empty.product details are missing', 400);

        next(httpError);

        return;

    }

    addArticle(body)

        .then(article => {

               res.status(201).json(article);

        })

        .catch(err => {

            const httpError=new HttpError(err.message,400);

            next(httpError);

        });

}

const putArticle = (req, res, next) => {

    const { body } = req;

    const { \_id } = req.params;

    if (Object.keys(body).length === 0) {

        const httpError = new HttpError('request body is empty.product details are missing', 400);

        next(httpError);

        return;

    }

    updateArticle(\_id, body)

        .then(article => {

            res.json(article);

        })

        .catch(err => {

            res.json(err.message);

        })

}

const deleteArticle = (req, res, next) => {

    const { \_id } = req.params;

    removeArticle(\_id)

        .then(() => {

            res.status(204).json()

        })

        .catch((err) => {

            if (err.kind === 'ObjectId') {

                const httpError = new HttpError('Invalid product id', 400)

                next(httpError);

            } else {

                const httpError = new HttpError(err.message, 500)

                next(httpError)

            }

        })

}

const postComment = (req, res, next) => {

    const { \_id } = req.params;

    const { body } = req;

    // check if the body is an empty object

    if( Object.keys( body ).length === 0 ) {

      const httpError = new HttpError( 'Request body is empty. Review details are missing.', 400 );

      next( httpError );

      return;

    }

    addComment( \_id, body )

      .then(article => {

        res.status( 201 ).json( article );

      })

      .catch(err => {

        const httpError = new HttpError( err.message, 400 );

        next( httpError );

      });

  };

  const getComments = (req, res, next) => {

    const { \_id } = req.params;

    fetchComments( \_id )

      .then((comments) => {

          res.json(comments);

      })

      .catch((err) => {

          const httpError = new HttpError( err.message, 500 );

          next( httpError );

      });

  }

export {

    getArticles,

    getArticlesById,

    postArticle,

    putArticle,

    deleteArticle,

    postComment,

    getComments

}

Controllers/pages/article.js

import { fetchArticles } from "../../services/article.js";

const getArticlePage=(req,res,next)=>{

    fetchArticles()

    .then(articles=>{

    res.render('articles',{

        articles:articles

    })

})

}

export{

    getArticlePage

}

Controllers/pages/index.js

const getHomePage=(req,res,next)=>{

    res.render('index');

}

export{

    getHomePage

}

Data/article.json

[

    {

      "author": "Prakitha",

      "title": "The lentin Salab",

      "abstract": "Let's talk about lentils: The best kind to use in  salads are those that do not fall apart when cooked. My favorites are mall green (often sold as lentils du Puy), small brown lentils (sometimes called castelluccio), or even black lentils. Cook them in salted water according to their package directions, or whatever your favorite way to cook lentils is, then cool them before using. ",

      "body": "Place onion, jalapeño, and garlic in a blender or food processor [see Note at end] and run the machine in short bursts until they're in smaller bits. Add two-thirds (just eyeball it) cilantro, drained tomatoes, salt, and juice of one lime and run the machine until the tomatoes are well-chopped but not totally smooth. When making salsa for scooping up with chips, I will add some of the reserved tomato juices, a tablespoon at a time, if the mixture is very thick, but to dress this lentil salad, we don't want it very thin. Adjust seasoning to ensure it's robust enough to dress the lentils, adding salt, pepper, and more lime juice, if needed to taste.",

      "imageUrl": "https://i0.wp.com/live.staticflickr.com/65535/51815628180\_bac90e84bf\_z.jpg?resize=640%2C427&ssl=1",

      "createdAt":"2021-12-12",

      "comments": [

        {

          "commenter": "hellobib",

          "title": "Must try!",

          "comment":"I can't wait to try this!",

          "commentedAt": "2021-12-14"

        }

      ]

    },{

      "author": "Samrudh",

      "title": "Short-nib onion soup",

      "abstract": "A couple months ago, I was out with friends and we stopped briefly back at a friend's place (hi Jocelyn!). It smelled amazing and it turned out she had chicken chili going in the crockpot. Despite not planning to stay, we inhaled a bowl in her yard before heading back out again and I have not stopped thinking about it since, hospitality on a you-never-know level. Stews and hearty soups are already wired with this energy — they keep well, are easily reheated, and if nobody else eats it, you're happy to have it for yourself.",

      "body": "There are three components here: 1. Oven-braised short ribs, cooking the meat and creating the dreamiest broth. 2. Caramelized onions on the stove, building the soup's base. 3. Cheese croutons to sink into the final soup bowls. Because this is rich, I like to serve it on the smaller side, in 12-ounce (or 1.5-cup) bowls. It's definitely closer to a stew in consistency, but if you'd like it more soup-like, you can add more broth to thin it. You can also, if you wish, broil the cheese toasts on top of the soup, as onion soup is more traditionally served. I used this short rib onion soup as guidance here, but ended up needing more short ribs and using my own, preferred, onion soup method.",

      "imageUrl": "https://i0.wp.com/smittenkitchen.com/wp-content/uploads/2021/12/short-rib-onion-soup-13-scaled.jpg?resize=1536%2C1024&ssl=1",

      "createdAt":"Mar 19, 2021",

      "comments": [

        {

          "commenter": "hiiiindb",

          "title": "Wow!",

          "comment":"I tried it and its yummy",

          "commentedAt": "2021-05-10"

        }

      ]

    },{

      "author": "Manoritha",

      "title": "Checkboard cookies",

      "abstract": " I imagined checkerboard cookies would be a hideous amount of work for something that looked cute but probably didn't taste like much, the dark portions chocolate in color, not flavor.",

      "body": "I used my favorite cookie methods outlined in my Unfussy Sugar Cookies for ease: starting with cold butter and rolling out the dough right away with no floured counter to clean up. A couple quick trips to the freezer firms up the dough, and from there you stack and slice to make the cutest checkerboards I've ever or tasted with no sweat or stress.",

      "imageUrl": "https://i2.wp.com/smittenkitchen.com/wp-content/uploads/2021/12/checkerboard-cookies-1-scaled.jpg?resize=1536%2C1024&ssl=1",

      "createdAt":"Apr 19, 2021",

      "comments": [

        {

          "commenter": "sjkbdcb2tgi",

          "title": "Go for it!",

          "comment":"My kids loved it soo much thank you for the recepie",

          "commentedAt": "2022-01-10"

        }

      ]

    },{

      "author": "Butterscotch",

      "title": "Almond horn cookies",

      "abstract": "The only thing my mother ever asked me to bring home from the bakery where I worked in high school where almond horn cookies, or Mandelhörnchen, probably no surprise as we are a family of established marzipan fiends, most especially when dark chocolate is also involved.",

      "body": "If you have no soft spot for almond paste or almond extract, you should turn away now. It’s almost all they’re made of. They're also naturally flourless, gluten-, leavener- and dairy-free (if you use a dairy-free chocolate); the last time we had a cookie that checked all of these boxes it was all I could talk about for the next six months.      ",

      "imageUrl": "https://i1.wp.com/smittenkitchen.com/wp-content/uploads//2017/04/almond-horns-mandelhoernchen.jpg?fit=750%2C500&ssl=1",

      "createdAt":"May 19, 2021",

      "comments": [

        {

          "commenter": "hetholo",

          "title": "Must try!",

          "comment":"The texture is super",

          "commentedAt": "2021-10-15"

        }

      ]

    }

  ]

Data/index.js

import mongoose from 'mongoose';

import '../models/comment.js';

const connectionStr = `mongodb://localhost:27017/food\_blog\_app`;

mongoose.connect(connectionStr)

    .then(() => {

        console.log('connected to database');

    })

    .catch(err => {

        console.log(err.message);

    })

Models/comment.js

import mongoose from 'mongoose';

import commentsSchema from './commenters.js';

const articleSchema = new mongoose.Schema({

    author: {

        type: String,

        required: true,

    },

    title: {

        type: String,

        required: true,

    },

    abstract: {

        type: String,

        required: true,

    },

    body: {

        type: String,

        required: false

    },

    imageUrl: {

        type: String,

        required: false

    },

    createdAt: {

        type: Date,

        required: true,

        default: Date.now

    },

    comments: {

        type: [ commentsSchema ],

    },

});

// register the model

mongoose.model('article', articleSchema);

models/commenters.js

import mongoose from 'mongoose';

const commentsSchema = new mongoose.Schema({

    commenter: {

        type: String,

        required: true,

    },

    title: {

        type: String

    },

    comment: {

        type: String,

        required: true

    },

    createdAt: {

        type: Date,

        required: true,

        default: Date.now

    }

});

export default commentsSchema;

routes/article.js

import express from 'express';

import { getArticles,getArticlesById, postArticle, putArticle ,deleteArticle,postComment,getComments} from '../controllers/article.js';

// create a router

const router = express.Router();

router.get('/', getArticles);

router.get('/:\_id', getArticlesById);

router.post('/', postArticle);

router.put('/:\_id', putArticle);

router.delete('/:\_id',deleteArticle);

router.post(   '/:\_id/comments', getComments );

router.post(   '/:\_id/comments', postComment );

export default router;

routes/pages/article.js

import express from 'express';

import { getArticlePage } from "../../controllers/pages/article.js";

const router = express.Router();

router.get( '/', getArticlePage);

export default router;

routes/pages/index.js

import express from "express";

import { getHomePage } from "../../controllers/pages/index.js";

const router =express.Router();

router.get('/',(req,res,next)=>{

    res.render('index');

})

router.get('/',getHomePage)

export default router;

services/article.cjs

const articles =require('../data/article.json')

const fetchArticles=()=>articles;

module.exports={

    fetchArticles

}

Services/article.js

/\*\*

 \* Service talks to the database (data layer)

 \*/

import mongoose from "mongoose";

//import normalize from "normalize\_mongoose";

import config from '../utils/config.js'

const Article = mongoose.model('article');

const fetchArticles = (sort, order, page, q) => {

  const skipClause = (page - 1) \* config.PAGE\_SIZE;

  const filterClause = {}

  if (q) {

    filterClause.description = {

      $regex: q,

      $options: 'i'// ignore case

    }

  }

  console.log(filterClause);

  const sortClause = {

    [sort]: order === 'desc' ? -1 : 1

  }

  return Article.find(filterClause).

    select('author title abstract body imageUrl  createdAt')

    .sort(sortClause).skip(skipClause).limit(config.PAGE\_SIZE);

};

const fetchArticlesById = (\_id) => {

  return Article.findById(\_id);

};

const addArticle = (article) => {

  return Article.create(article);

}

const updateArticle = (\_id, newArticleDetails) => {

  return Article.findByIdAndUpdate(\_id, newArticleDetails, { new: true, runValidators: true });

}

const removeArticle = (\_id) => {

  return Article.findByIdAndRemove(\_id);

}

const addComment = (\_id, comments) => {

  return Article.findByIdAndUpdate(

    \_id,

    {

      $push: {

        comments: comments

      }

    },

    {

      new: true,

      runValidators: true

    }

  );

};

const fetchComments = (\_id) => {

  return Article

    .findById(\_id)

    .select('comments')

    .then(articleComment=> articleComment.comments)

}

export {

  fetchArticles,

  fetchArticlesById,

  addArticle,

  updateArticle,

  removeArticle,

  addComment,

  fetchComments

};

Utils/config.js

const config={

    PAGE\_SIZE:5

}

export default config;

utils/HttpError.js

class HttpError extends Error{

    constructor(message,statusCode){

        super(message);

        this.statusCode=statusCode;

    }

}

export default HttpError

views/articles.ejs

<%- include( "header" ) -%>

    <h1 class="my-5">

        List of Recipie

        <br />

        <small class="h4">The BLOG SPOT</small>

    </h1>

    <div class="row">

        <% for( let i = 0; i < articles.length; i++ ) { %>

            <div class="col-12 col-lg-3">

                <div class="card" style="width: 18rem;">

                    <img src="<%=articles[i].imageUrl%>" class="card-img-top" style="width: 40%; margin: 0 auto;">

                    <div class="card-body">

                      <h5 class="card-title">

                        <%=articles[i].title%>

                      </h5>

                      <div class="card-text">

                        <%=articles[i].abstract%>

                      </div>

                      <br/>

                      <div class="card-text">

                        <%=articles[i].body%>

                      </div>

                      <br/>

                      <div class="card-text">

                        <%=articles[i].comments%>

                      </div>

                      <a href="#" class="btn btn-primary">click the like button</a>

                    </div>

                  </div>

            </div>

        <% } %>

    </div>

    <%- include( "footer" ) -%>

Views/footer.ejs

<script src="/bootstrap-5.1.3-dist/css/bootstrap.min.js"></script>

</body>

</html>

Views/header.ejs

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Foodies SPOT</title>

    <link rel="stylesheet" href="/bootstrap-5.1.3-dist/css/bootstrap.min.css">

</head>

<body class="container">

    <%- include( "navigation-menu" ) -%>

Views/index.ejs

<%-include("header")-%>

    <h1 class="my=4">Amazin</h1><br/>

    <small class="h4">The Amazin online store</small>

    <p>Welcome to BLOG SPOT. You can find many interesting articles.

        The one stop destination for many articles.

    </p>

        <p>Author:Prakitha</p>

        <p>

            Tech stack:

            <ul>

                <li>Prakitha</li>

                <li>Samrudh</li>

            </ul>

        </p>

<%-include("footer")-%>

Views/navigation-menu.js

<nav class="navbar navbar-expand-lg navbar-dark bg-primary">

    <div class="container-fluid">

      <a class="navbar-brand" href="/">Foodies SPOT</a>

      <button

        class="navbar-toggler"

        type="button"

        data-bs-toggle="collapse"

        data-bs-target="#navbarNav"

        aria-controls="navbarNav"

        aria-expanded="false"

        aria-label="Toggle navigation"

      >

        <span class="navbar-toggler-icon"></span>

      </button>

      <div class="collapse navbar-collapse" id="navbarNav">

        <ul class="navbar-nav">

          <li class="nav-item">

            <a class="nav-link" aria-current="page" href="/page/articles">Recipies</a>

          </li>

        </ul>

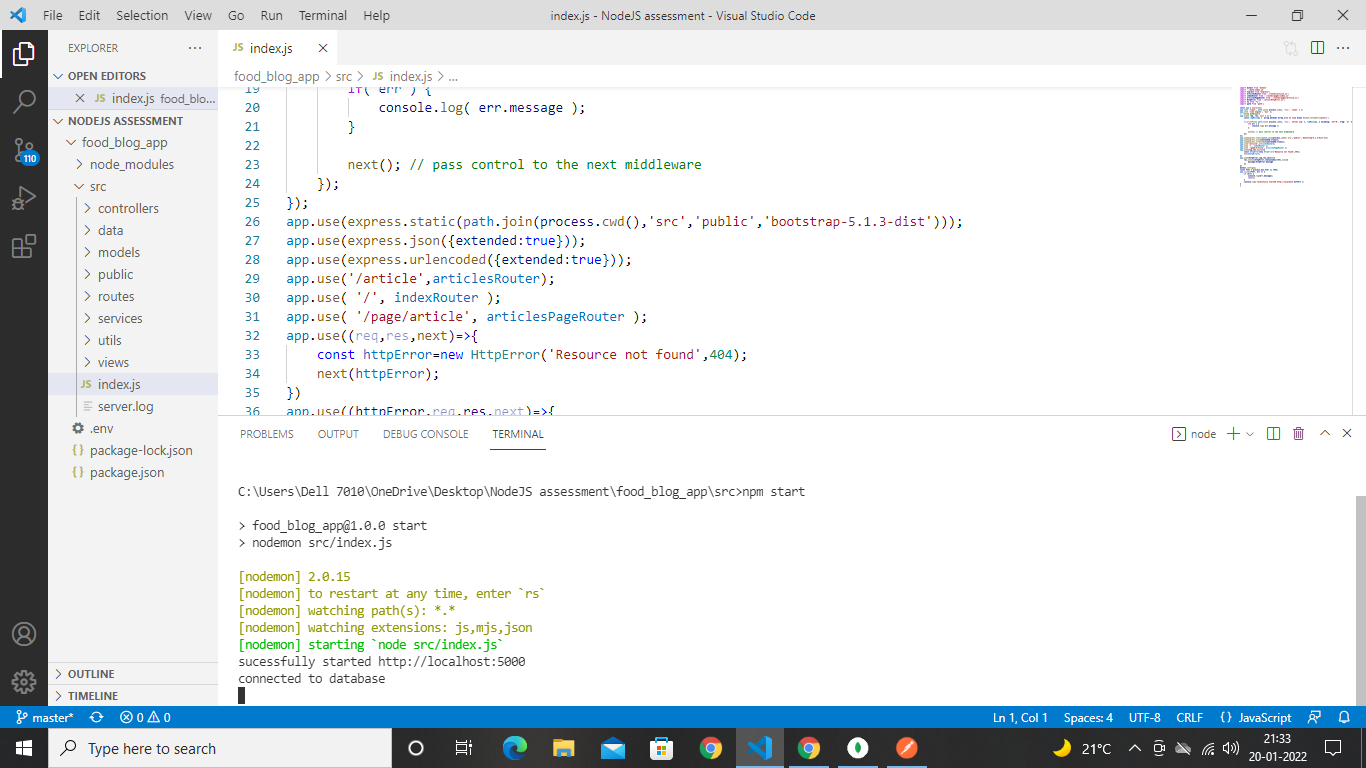
      </div>

    </div>

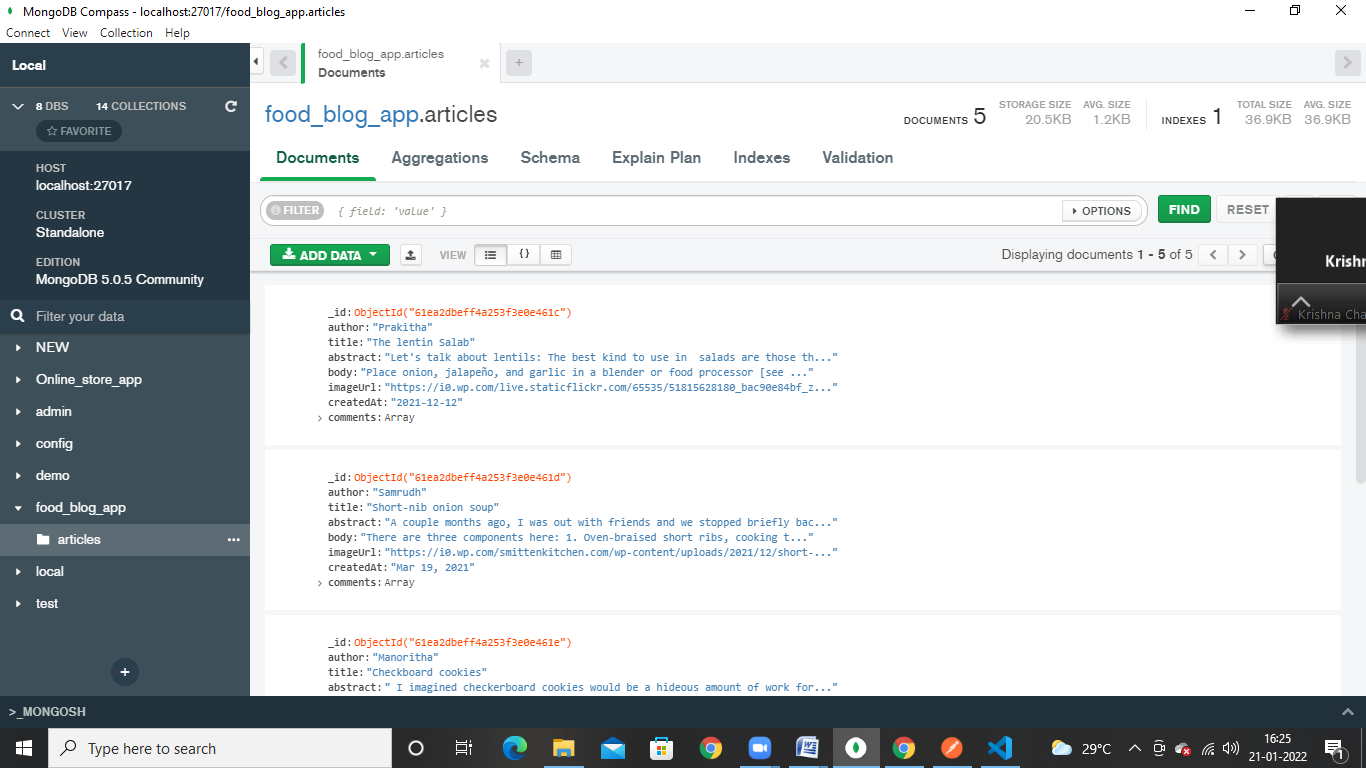
  </nav>

Output pictures:

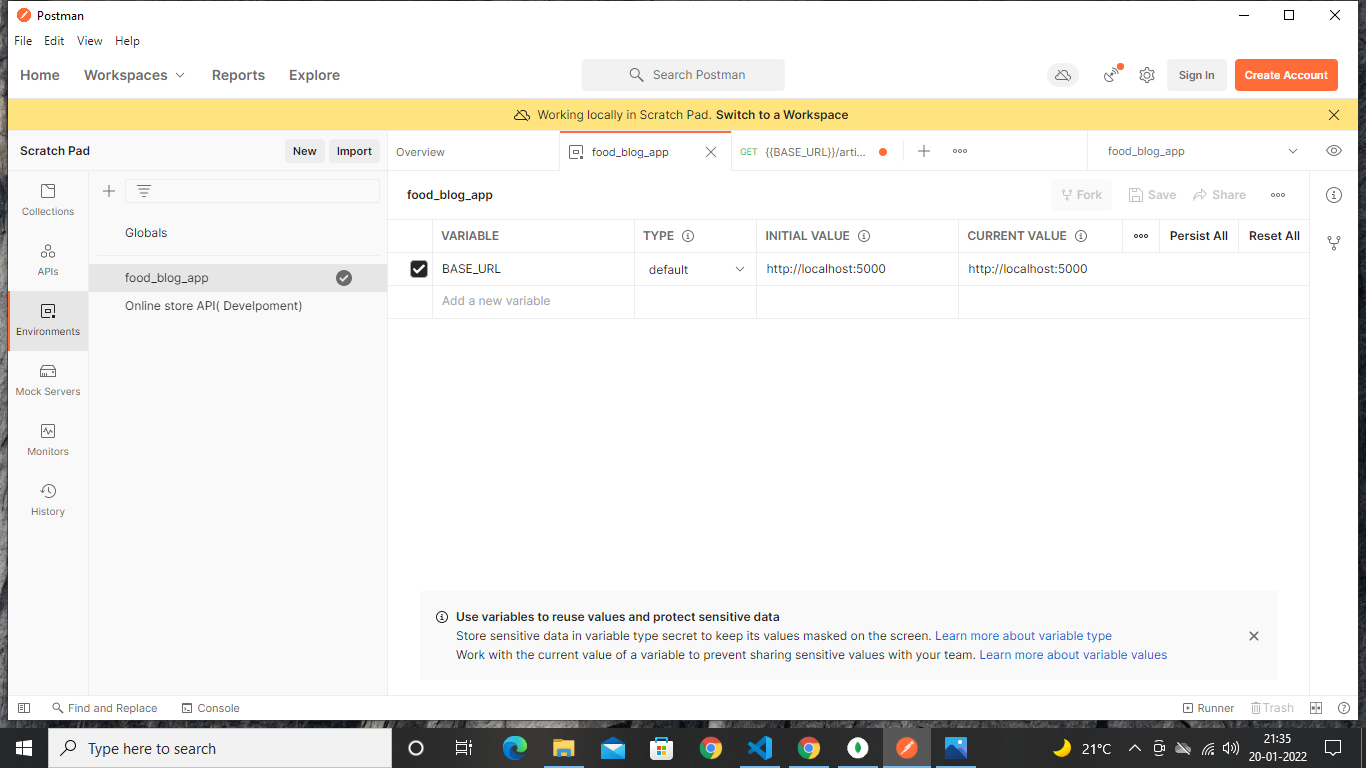
1.Program started running



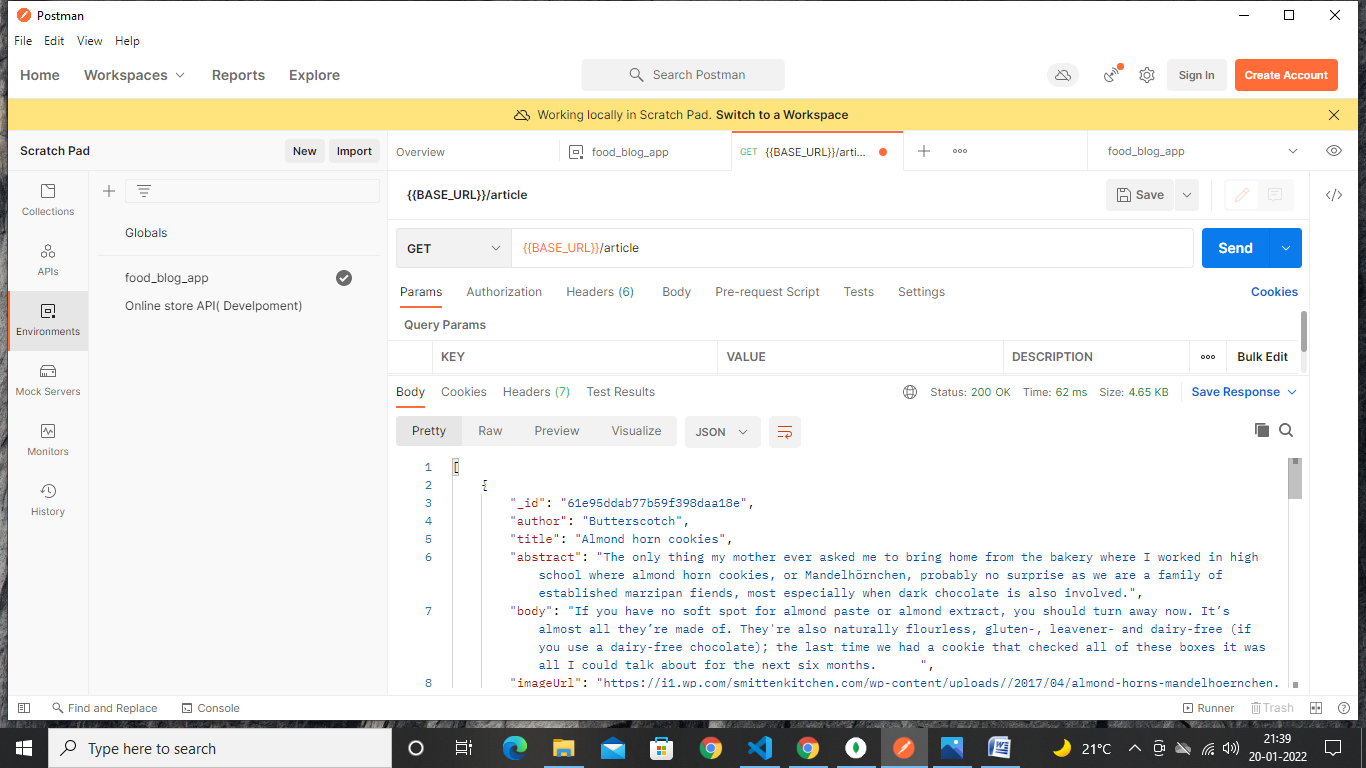
2.connected the data to mongo



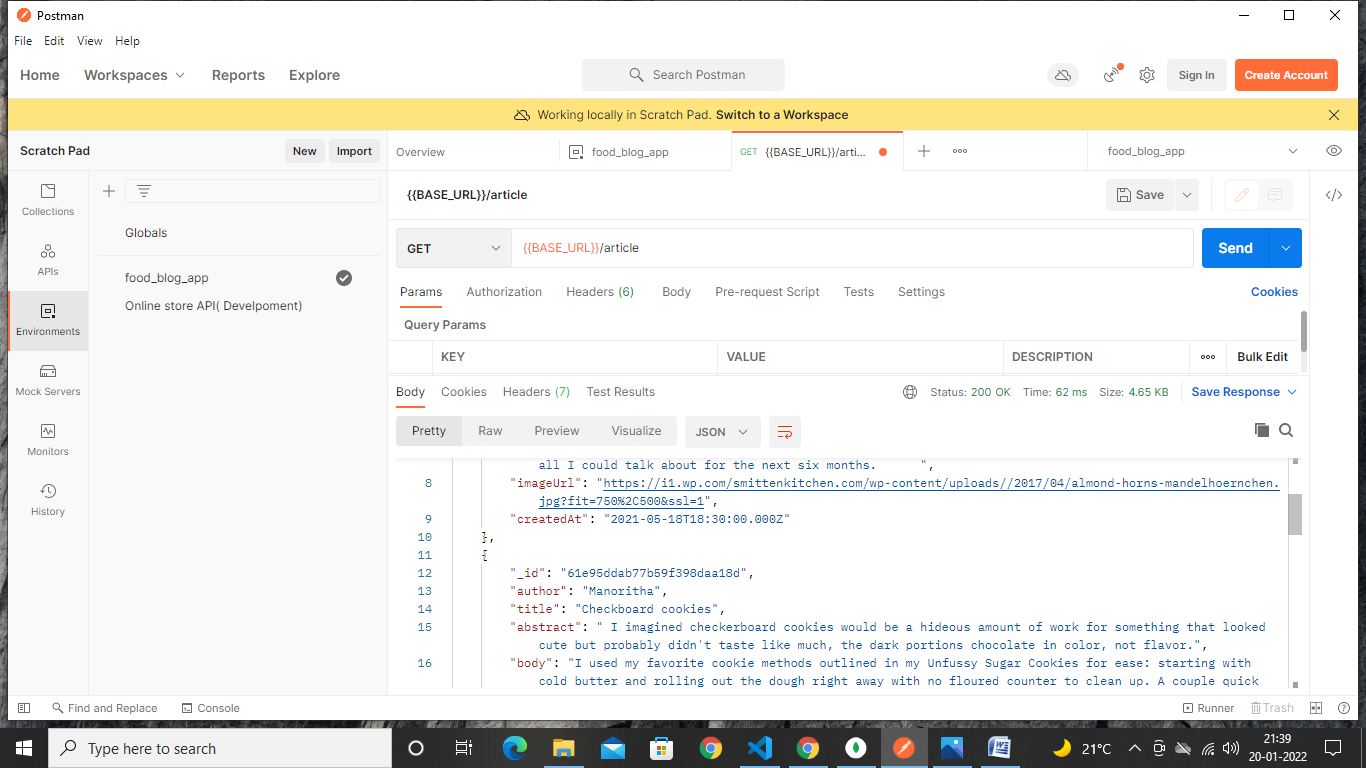
3. created enviroment in post man

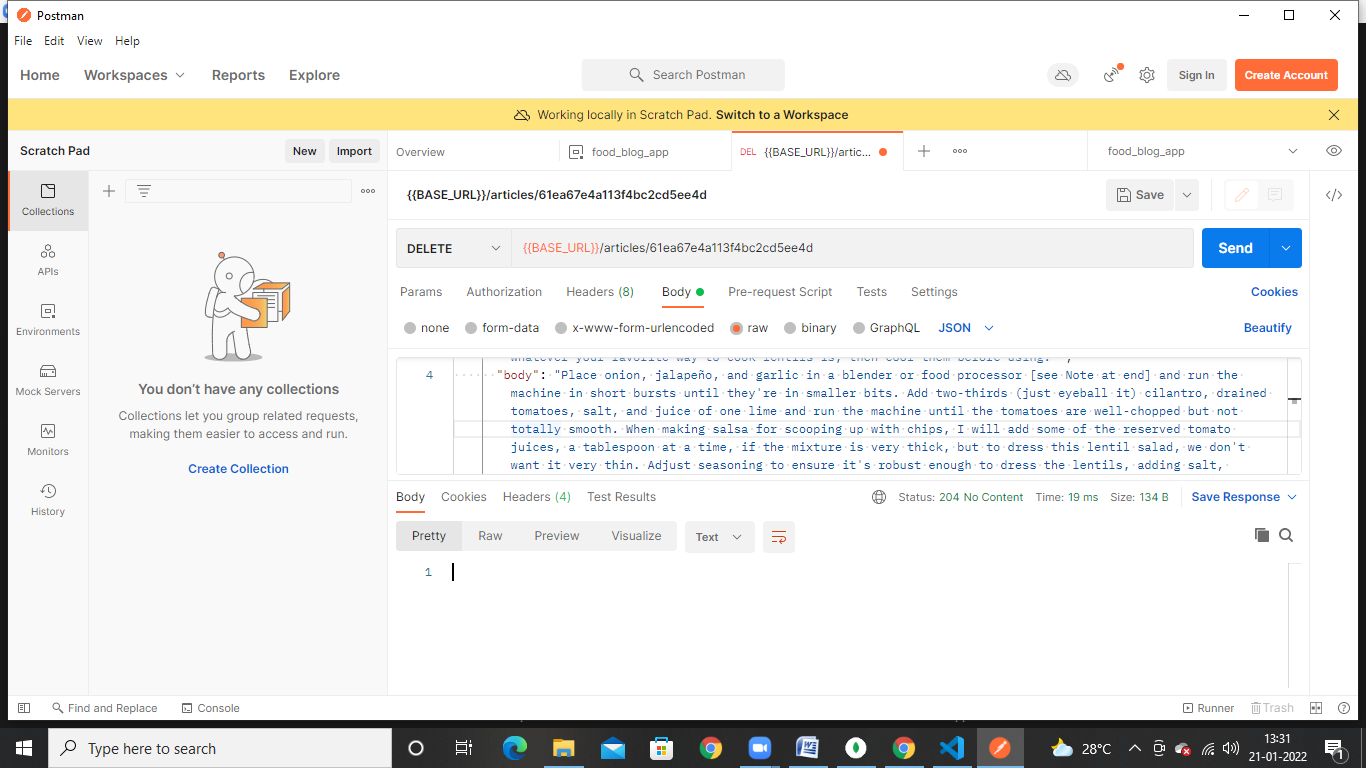


4.get articles

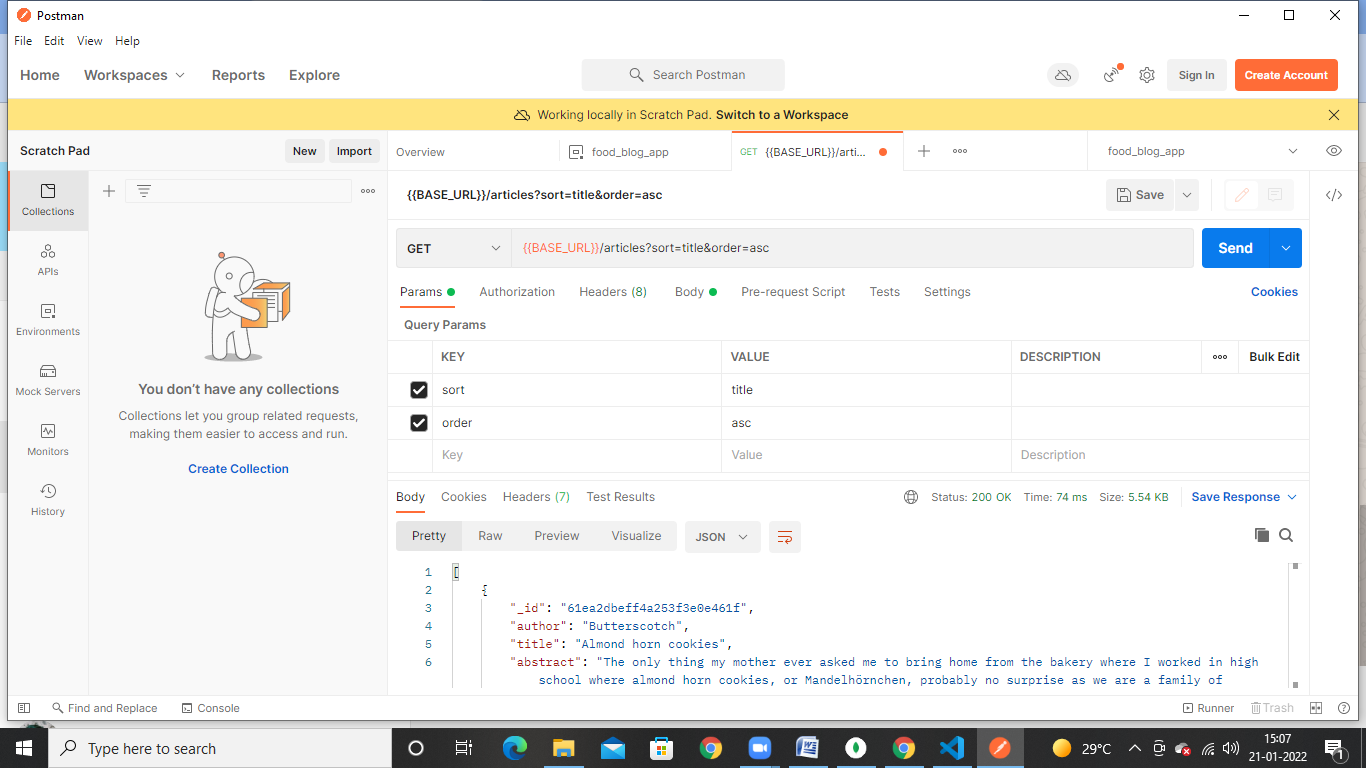


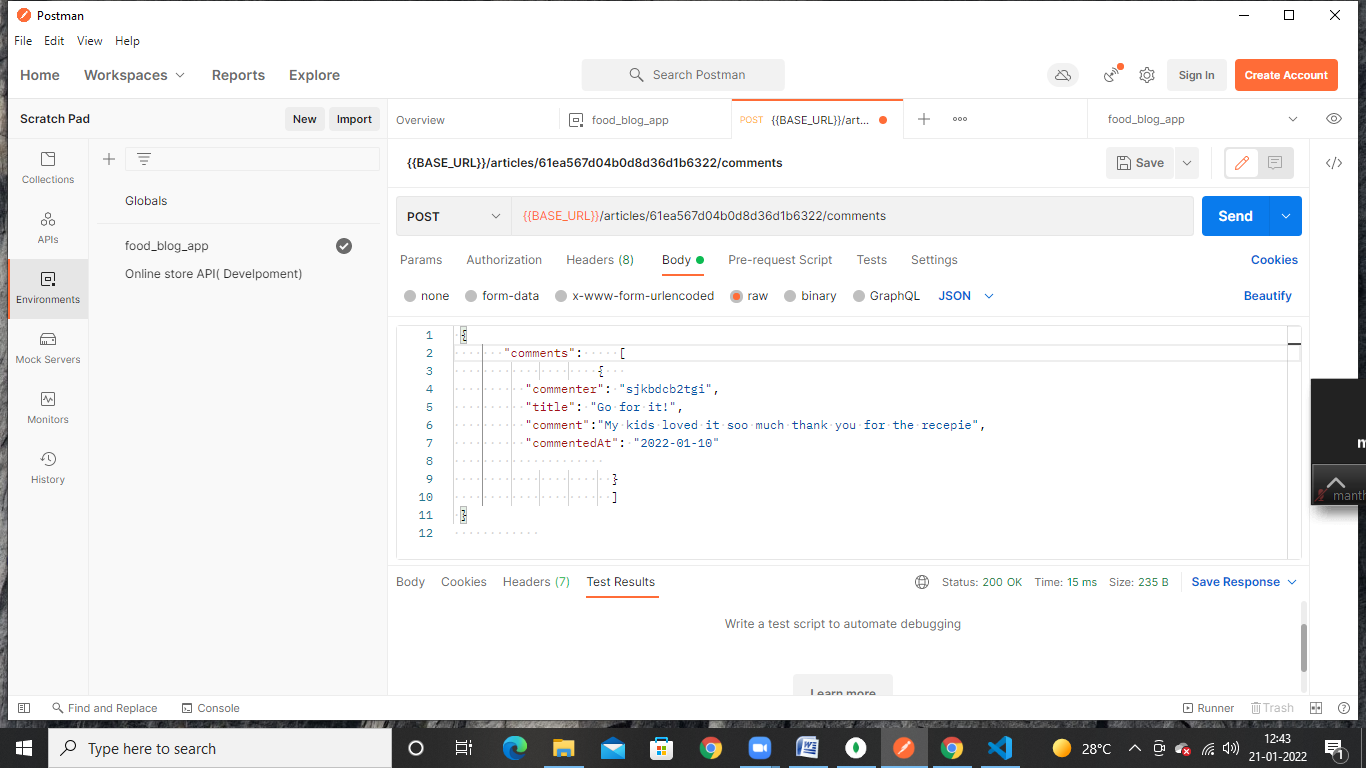
5.get all articles



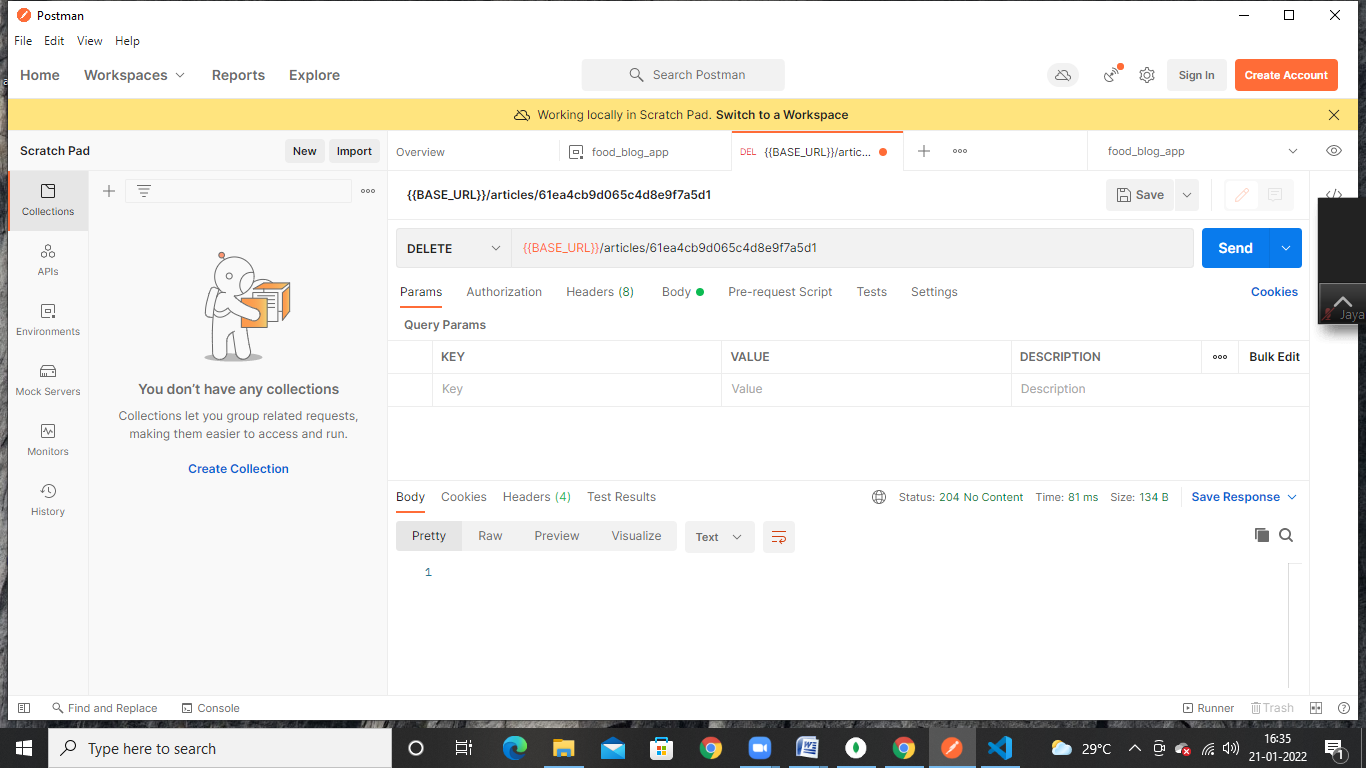
6.get articles by id

7.using praser



8.post 

9. delete



Page url response

