Advanced Database Topics

Prepared By

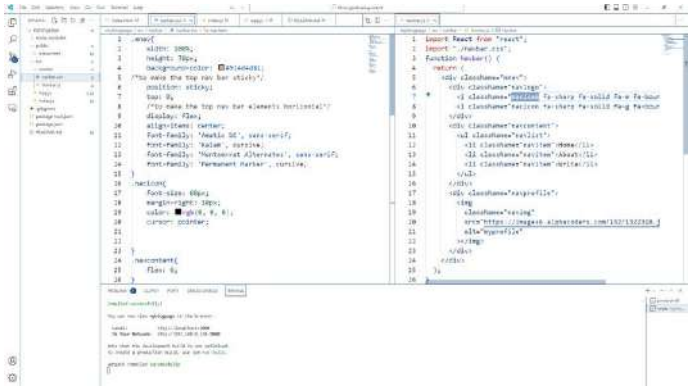
Monika Govindaraj (110122262)

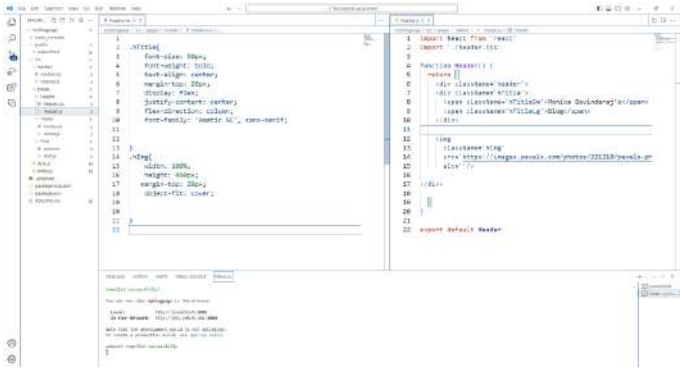
Project Hierarchy

Client	Server
1.nav bar a.navbar.js 2.pages a.Login.js b.Register.js c.Home.js d.Header.js e.newPost.js f.Post.js g.DetailPage.js h.RowPostContainer.js i.Comments.js 3.App.js 4.Index.js	1.Index.js 2.env file 3.Models a.Category.js b.Comments.js c.Post.js d.User.js 4.routes/Ser a.category.js b.comments.js c.post.js d.user.js

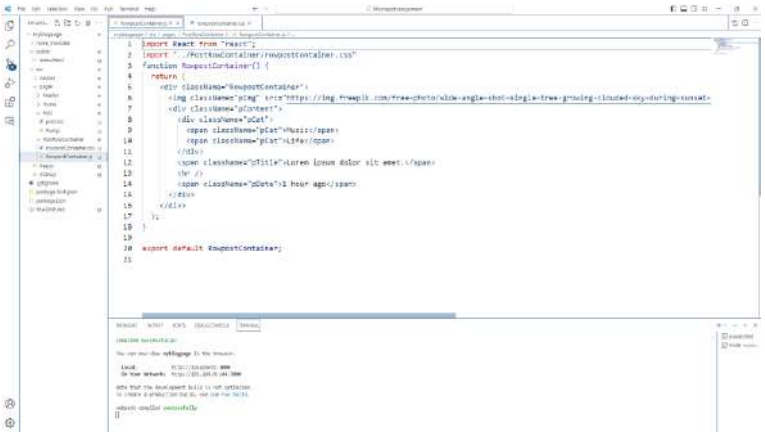
---

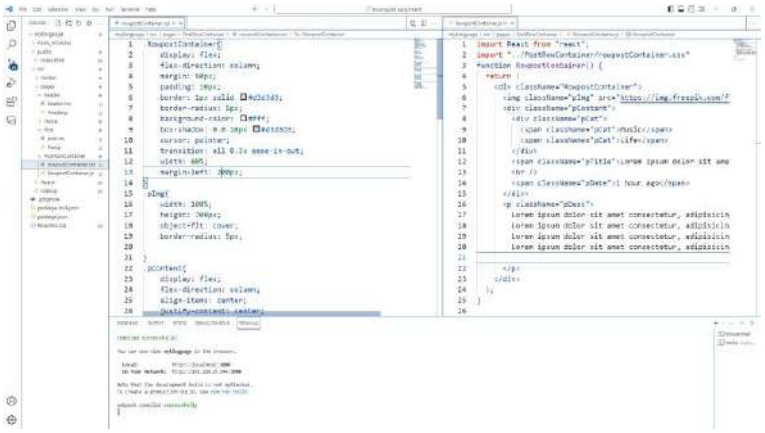
1. Create Blogging platform Frontend Page

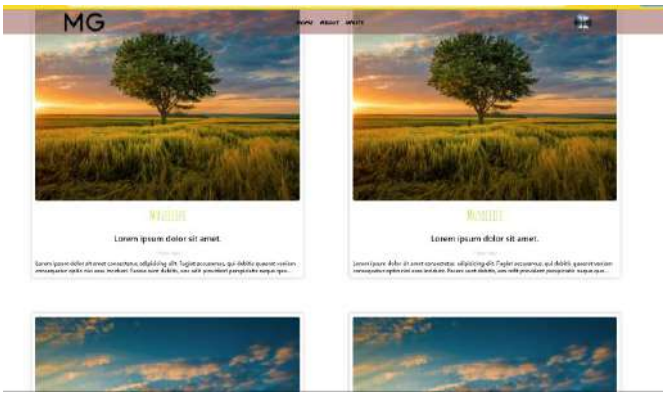


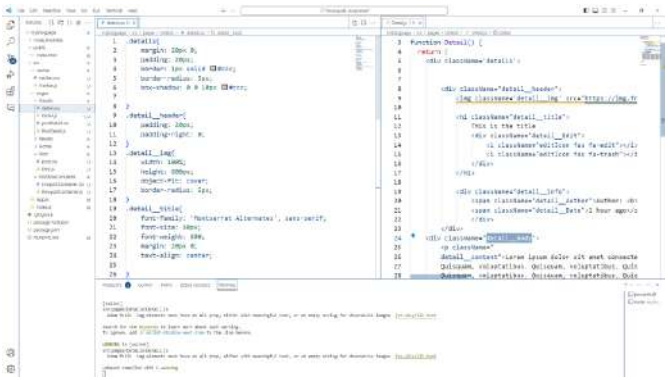


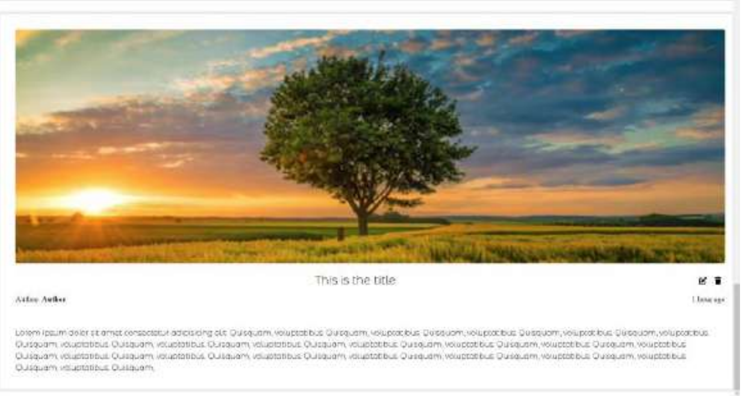

MG
HOME ABOUT WRITE
Monika Govindaraj's
Blog






MG
Lorem ipsum dolor sit amet.
Lorem ipsum dolor sit amet.





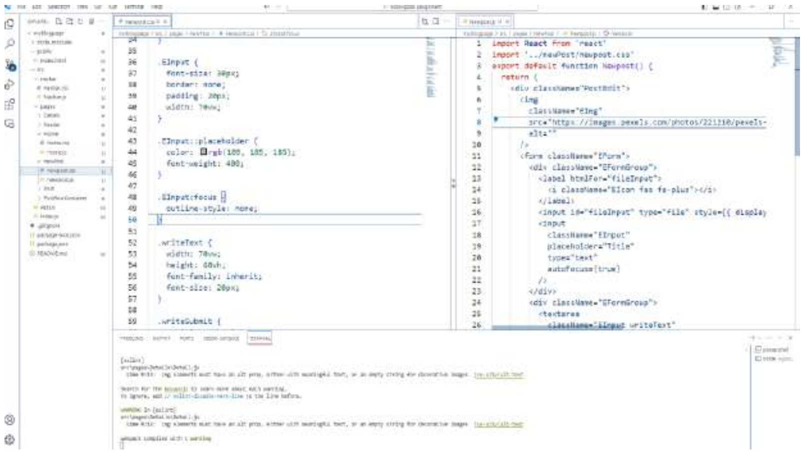
This is the title

Author: Author

1 hour ago

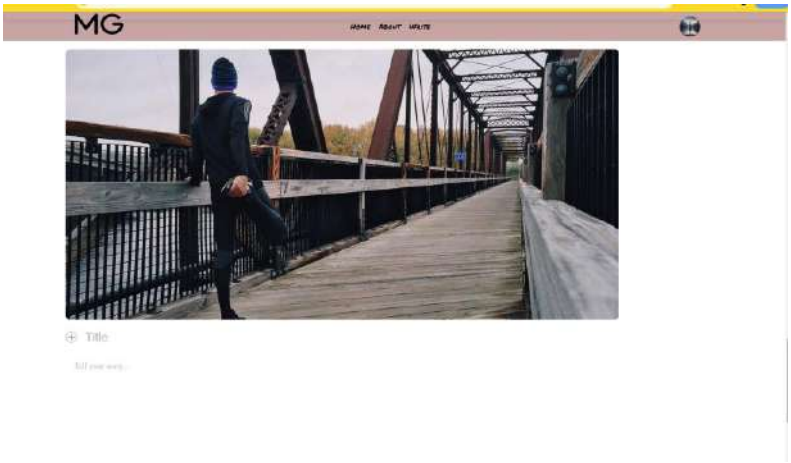
Lorem ipsum dolor sit amet consectetur adipisicing elit. Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam, voluptatibus Quisquam.

Create a new post page

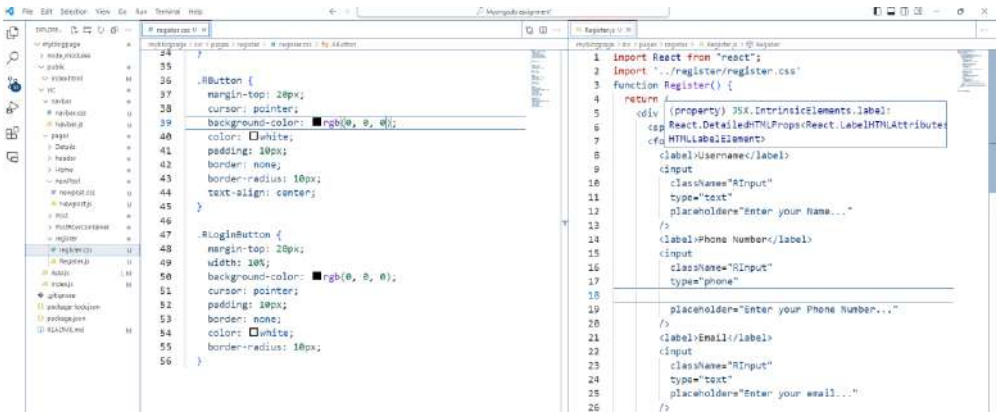


```
}

.EInput {
  font-size: 30px;
  border: none;
  padding: 20px;
  width: 70vw;
}

.EInput::placeholder {
  color: rgb(180, 185, 185);
  font-weight: 400;
}

.EInput:focus {
  outline-style: none;
}

.writeText {
  width: 70vw;
  height: 60vh;
  font-family: inherit;
  font-size: 20px;
}

.writeSubmit {
```

```
import React from 'react'
import '../newPost/newpost.css'
export default function Newpost() {
  return (
    <div className="PostEdit">
      <img
        className="EImg"
        src="https://images.pexels.com/photos/221210/pexels-
        alt=""
      />
      <form className="EForm">
        <div className="EFormGroup">
          <label htmlFor="fileInput">
            <i className="EIcon fas fa-plus"></i>
          </label>
          <input id="fileInput" type="file" style={{ display
          <input
            className="EInput"
            placeholder="Title"
            type="text"
            autoFocus={true}
          />
        </div>
        <div className="EFormGroup">
          <textarea
            className="EInput writeText"
```

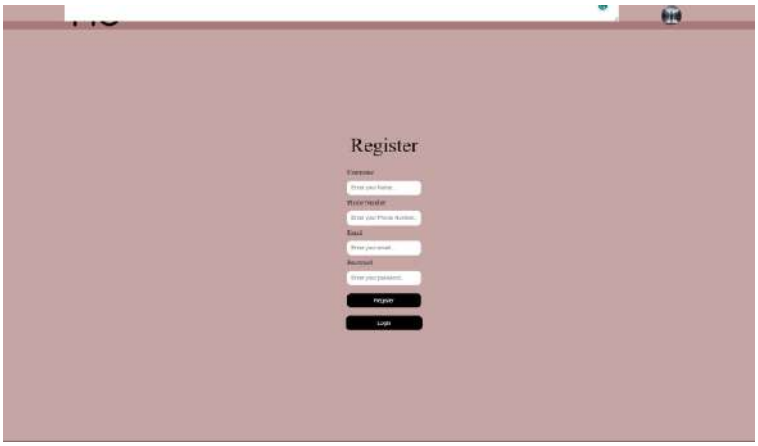


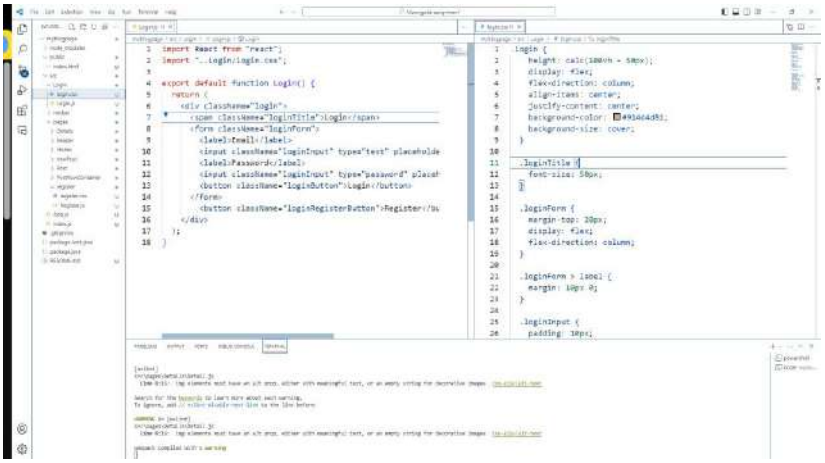
Create a register page and Login Page



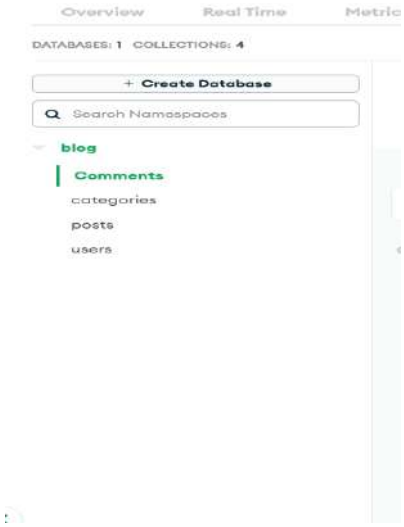
```
.RButton {
  margin-top: 20px;
  cursor: pointer;
  background-color: rgb(0, 0, 0);
  color: white;
  padding: 10px;
  border: none;
  border-radius: 10px;
  text-align: center;
}

.RLoginButton {
  margin-top: 20px;
  width: 10%;
  background-color: rgb(0, 0, 0);
  cursor: pointer;
  padding: 10px;
  border: none;
  color: white;
  border-radius: 10px;
}
```

```
import React from "react";
import '../register/register.css'
function Register() {
  return (
    <div
      <label>Username</label>
      <input
        className="RInput"
        type="text"
        placeholder="Enter your Name..."
      />
      <label>Phone Number</label>
      <input
        className="RInput"
        type="phone"

        placeholder="Enter your Phone Number..."
      />
      <label>Email</label>
      <input
        className="RInput"
        type="text"
        placeholder="Enter your email..."
      />
```





2. Next Step is to create a MongoDB schema



User Schema:

Categories schema:

```
_id: ObjectId('6537bf9e7445751c0bc12f52')
username: "moni"
title: "Coding For every one"
createdAt: "2023-10-23T08:49:38.659+00:00"
▾ tag: Array
  0: "coding"
  1: "Technology"
```

Comments schema:

```
_id: ObjectId('6537c09e7445751c0bc12f53')
username: "monika"
comment: "Nice Post"
postID: ""
createdAt: "2023-10-23T17:50:42.574+00:00"
__V: "0"
updatedAt: "2023-10-23T17:50:42.574+00:00"
```

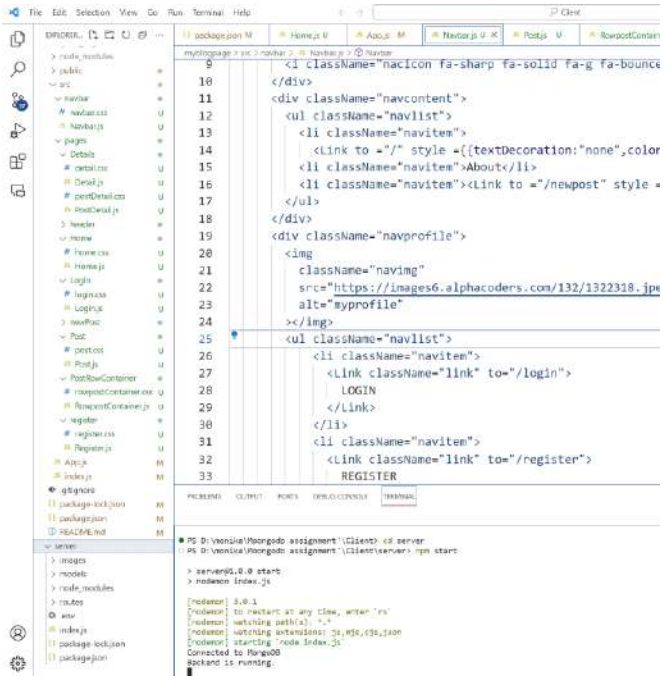
Blogposts schema:

```
_id: ObjectId('6536b73ff548721787d6a522')
title: "test"
desc: "testt"
username: "monika"
▸ categories: Array
createdAt: 2023-10-23T18:11:11.878+00:00
updatedAt: 2023-10-23T18:11:11.878+00:00
__v: 0
```

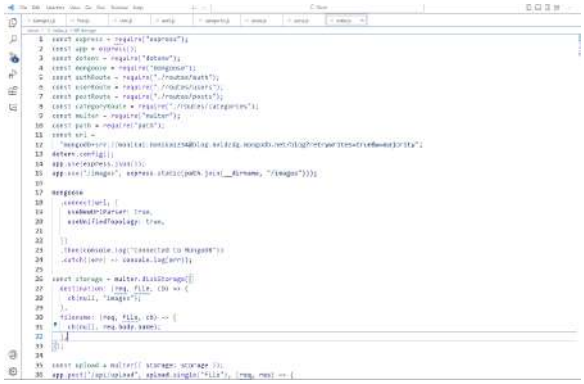
0

```
_id: ObjectId('6536e1d0770e8b6451e8a6db')
title: "My new Ideas"
desc: "my life has various ideas"
username: "monigov"
▸ categories: Array
createdAt: 2023-10-23T21:12:48.250+00:00
updatedAt: 2023-10-23T21:12:48.250+00:00
__v: 0
```

3. Project Structure



Index.js



Models

1. User.js

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

const UserSchema = new mongoose.Schema(
  {
    username: {
      type: String,
      required: true,
      unique: true,
    },
    email: {
      type: String,
      required: true,
      unique: true,
    },
    password: {
      type: String,
      required: true,
    },
    profilePic: {
      type: String,
      default: "",
    },
  },
  { timestamps: true }
);

module.exports = mongoose.model("User", UserSchema);
```

2.Post.js

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

const PostSchema = new mongoose.Schema(
  {
    title: {
      type: String,
      required: true,
      unique: true,
    },
    desc: {
      type: String,
      required: true,
    },
    photo: {
      type: String,
      required: false,
    },
    username: {
      type: String,
      required: true,
    },
    categories: {
      type: Array,
      required: false,
    },
  },
  { timestamps: true }
);

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

3.Categories.js

JS Category.js × | JS Post.js | JS User.js | JS auth.js | JS categories.js | JS posts.js | JS users.js | JS index.js

server > models > JS Category.js > ...

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

const CategorySchema = new mongoose.Schema(
  {
    name: {
      type: String,
      required: true,
      unique: true,
    },
  },
  { timestamps: true }
);

module.exports = mongoose.model("Category", CategorySchema);
```

4.Comments.js

Category.js | Comments.js | Post.js | User.js | auth.js | categories.js | posts.js | users.js | index.js

server > models > Comments.js > CommentsSchema > comment

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

const CommentsSchema = new mongoose.Schema(
  {
    username: {
      type: String,
      required: true,
      unique: true,
    },
    title: {
      type: String,
      required: true,
    },
    comment: {
      type: String,
      required: true,
    },
    postid: {
      type: String,
      required: true,
    },
  },
  { timestamps: true }
);

module.exports = mongoose.model("Comments", CommentsSchema);
```

Controller

1.Auth.js

```
const router = require("express").Router();
const User = require("../models/User");
const bcrypt = require("bcrypt");


router.post("/register", async (req, res) => {
  try {
    const salt = await bcrypt.genSalt(10);
    const hashedPass = await bcrypt.hash(req.body.password, salt);
    const newUser = new User({
      username: req.body.username,
      email: req.body.email,
        phone: req.body.phone,
      password: hashedPass,
    });

    const user = await newUser.save();
    res.status(200).json(user);
  } catch (err) {
    res.status(500).json(err);
  }
});


router.post("/login", async (req, res) => {
  try {
    const user = await User.findOne({ username: req.body.username });
    !user && res.status(400).json("Wrong credentials!");

    const validated = await bcrypt.compare(req.body.password, user.password);
    !validated && res.status(400).json("Wrong credentials!");

    const { password, ...others } = user._doc;
    res.status(200).json(others);
  } catch (err) {
    res.status(500).json(err);
```

3.user.js

```
const router = require("express").Router();
const Category = require("../models/Category");

router.post("/", async (req, res) => {
  const newCat = new Category(req.body);
  try {
    const savedCat = await newCat.save();
    res.status(200).json(savedCat);
  } catch (err) {
    res.status(500).json(err);
  }
});

router.get("/", async (req, res) => {
    try {
      const cats = await Category.find();
      res.status(200).json(cats);
    } catch (err) {
      res.status(500).json(err);
    }
  });

module.exports = router;
```

4.post.js

```
ver > routes > JS post.js > ...
const router = require("express").Router();
const User = require("../models/User");
const Post = require("../models/Post");

//CREATE POST
router.post("/", async (req, res) => {
  const newPost = new Post(req.body);
  try {
    const savedPost = await newPost.save();
    res.status(200).json(savedPost);
  } catch (err) {
    res.status(500).json(err);
  }
});

//UPDATE POST
router.put("/:id", async (req, res) => {
  try {
    const post = await Post.findById(req.params.id);
    if (post.username === req.body.username) {
      try {
        const updatedPost = await Post.findByIdAndUpdate(
          req.params.id,
          {
            $set: req.body,
          },
          { new: true }
        );
        res.status(200).json(updatedPost);
      } catch (err) {
        res.status(500).json(err);
      }
    } else {
      res.status(401).json("You can update only your post!");
    }
  } catch (err) {
```

5.Category.js

```
const router = require("express").Router();
const Category = require("../models/Category");

router.post("/", async (req, res) => {
  const newCat = new Category(req.body);
  try {
    const savedCat = await newCat.save();
    res.status(200).json(savedCat);
  } catch (err) {
    res.status(500).json(err);
  }
});

router.get("/", async (req, res) => {
    try {
      const cats = await Category.find();
      res.status(200).json(cats);
    } catch (err) {
      res.status(500).json(err);
    }
  });

module.exports = router;
```

5.Comments.js

```
const router = require("express").Router();
const Comments = require("../models/Comments");

router.post("/", async (req, res) => {
  const newCom = new Comments(req.body);
  try {
    const savedCom = await newCom.save();
    res.status(200).json(savedCom);
  } catch (err) {
    res.status(500).json(err);
  }
});

router.get("/", async (req, res) => {
    try {
      const Coms = await Comegory.find();
      res.status(200).json(Coms);
    } catch (err) {
      res.status(500).json(err);
    }
  });

module.exports = router;
```

Mongo database connected

```
const path = require('path');
const url =
  "mongodb+srv://monikai:monikai234@blog.ovldzdg.mongodb.net/blog?retryWrites=true&w=majority";
dotenv.config();
app.use(express.json());
app.use("/images", express.static(path.join(__dirname, "/images")));

mongoose
  .connect(url, {
    useNewUrlParser: true,
    useUnifiedTopology: true,

  })
  .then(console.log("Connected to MongoDB"))
  .catch((err) => console.log(err));
```

4. Populate data to mongo db.

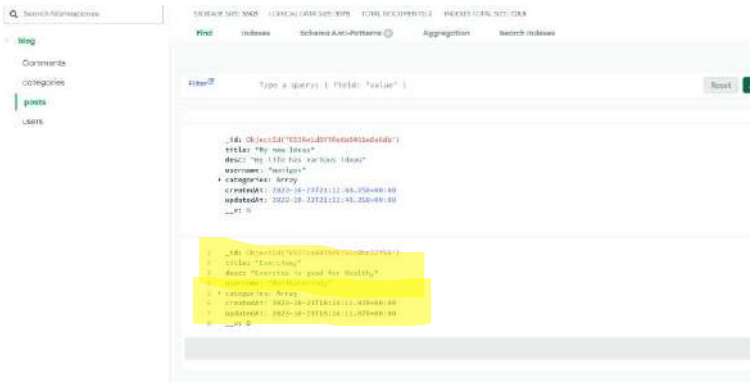
```
username: 'moni',
email: 'moni@gmail.com',
password: bcrypt.hashSync('pass1', 10),
},
{
username: 'moni1',
email: 'moni1@gmail.com',
password: bcrypt.hashSync('pass2', 10),
},
{
username: 'moni2',
email: 'moni2@gmail.com',
password: bcrypt.hashSync('pass3', 10),
},
];
const PostData = [
{
title: 'Post 1',
desc: 'Coding For every one',
username: 'moni',
categories: ['Coding', 'Technology'],
},
{
title: 'test',
desc: 'testt',
username: 'monika',
categories: ['music',Life],
},
];
const commentsData = [
{
commenterName: 'monika',
commentText: 'Nice Post',
blogPost: '6537c09e7445751c0bc12f53',
},
{
```

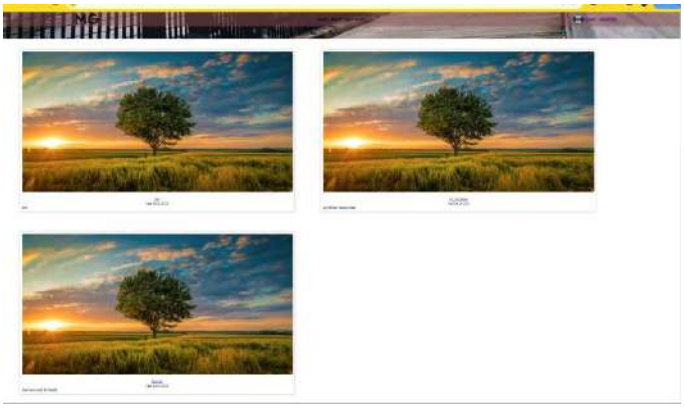
```
];
const commentsData = [
{
commenterName: 'monika',
commentText: 'Nice Post',
blogPost: '6537c09e7445751c0bc12f53',
},
{
commenterName: 'moni2',
commentText: 'Post!',
blogPost: '6536e1d0770e8b6451e8a6db',
}
];
async function populateDatabase() {
try {
await User.deleteMany({});
await BlogPost.deleteMany({});
await Comment.deleteMany({});

const createdUsers = await User.insertMany(usersData);
const createdBlogPosts = await BlogPost.insertMany(blogPostsData);

const blogPostTitleToId = createdBlogPosts.reduce((acc, post) => {
acc[post.title] = post._id;
return acc;
}, {});

const commentsWithIds = commentsData.map((comment) => ({
...comment,
blogPost: blogPostTitleToId[comment.blogPostTitle],
```

Next step is to fetch the data from the database to the website

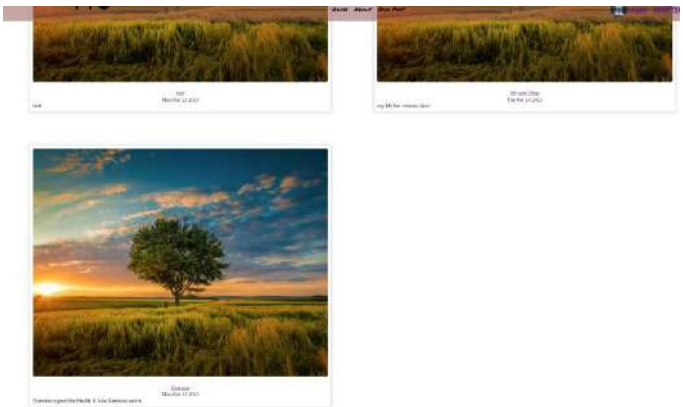
BLOG Post data is fetched from the database

Add New Post to MongoDB





Edit Post



Add Comments to Post