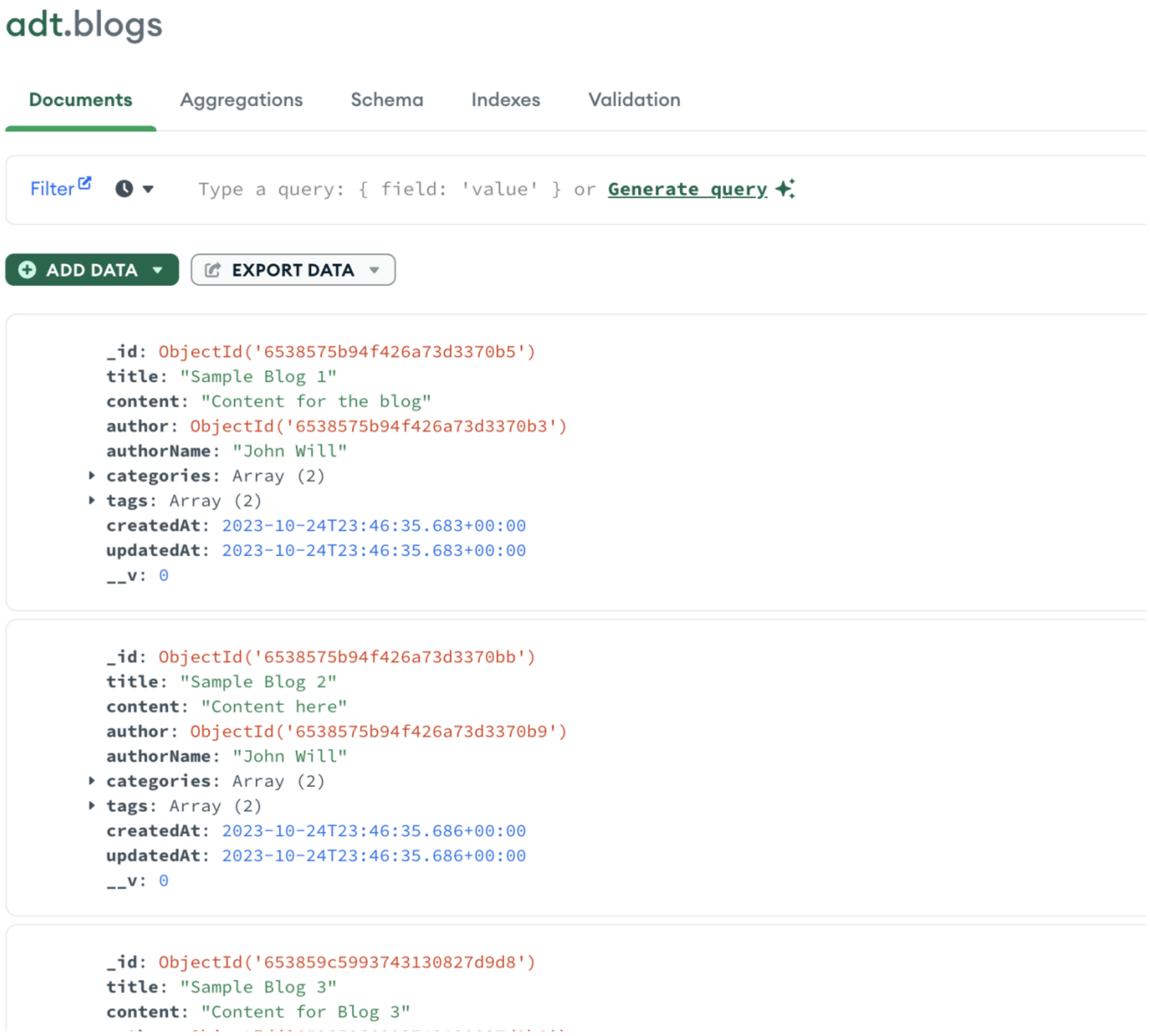
Advanced Database Topics (COMP-8157)

University of Windsor

Assignment -1

Name : Karan Mahajan
Section : 3
Student ID : 110119774

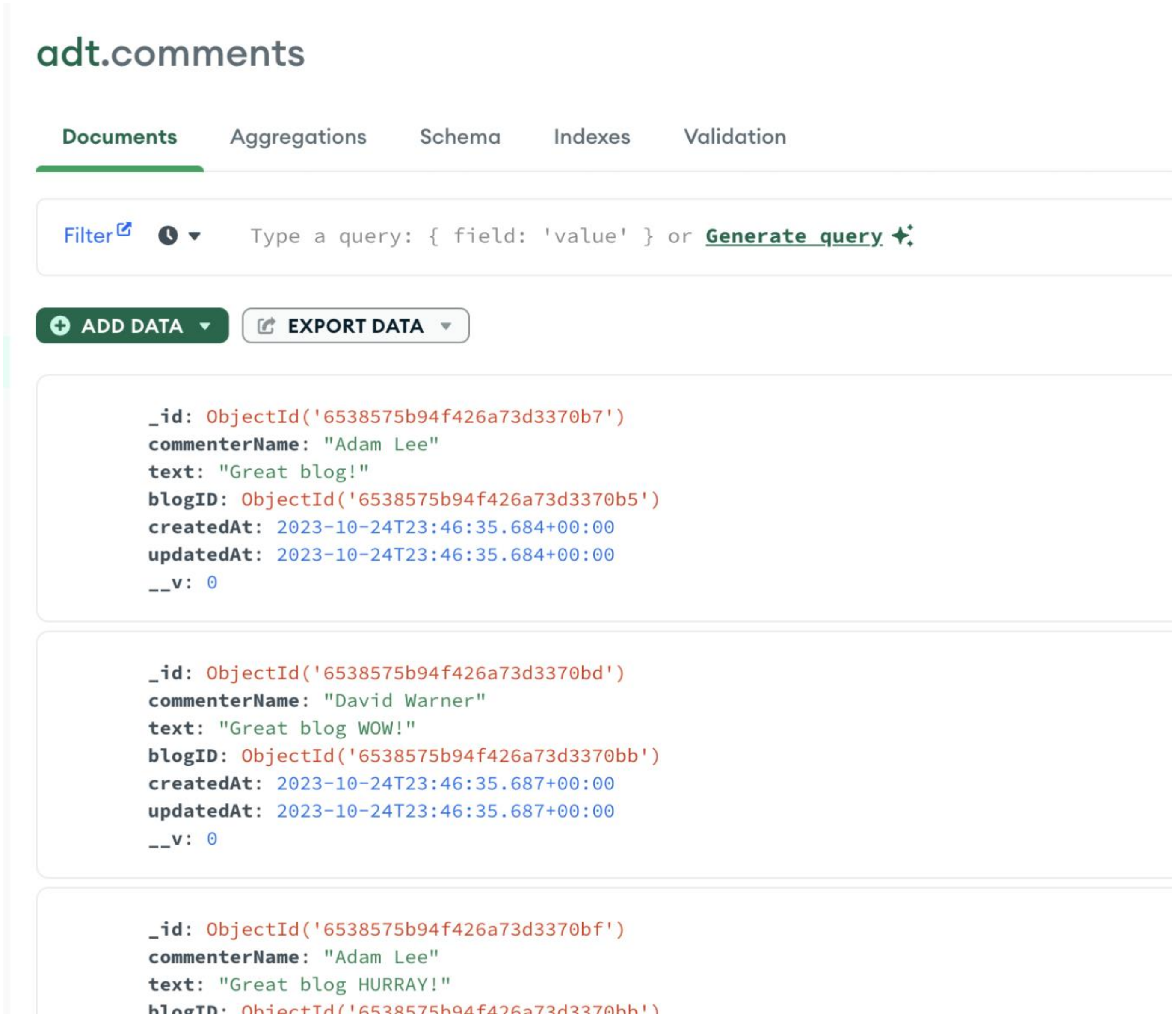
DB Schema



```
_id: ObjectId('6538575b94f426a73d3370b5')
title: "Sample Blog 1"
content: "Content for the blog"
author: ObjectId('6538575b94f426a73d3370b3')
authorName: "John Will"
categories: Array (2)
tags: Array (2)
createdAt: 2023-10-24T23:46:35.683+00:00
updatedAt: 2023-10-24T23:46:35.683+00:00
__v: 0
```

```
_id: ObjectId('6538575b94f426a73d3370bb')
title: "Sample Blog 2"
content: "Content here"
author: ObjectId('6538575b94f426a73d3370b9')
authorName: "John Will"
categories: Array (2)
tags: Array (2)
createdAt: 2023-10-24T23:46:35.686+00:00
updatedAt: 2023-10-24T23:46:35.686+00:00
__v: 0
```

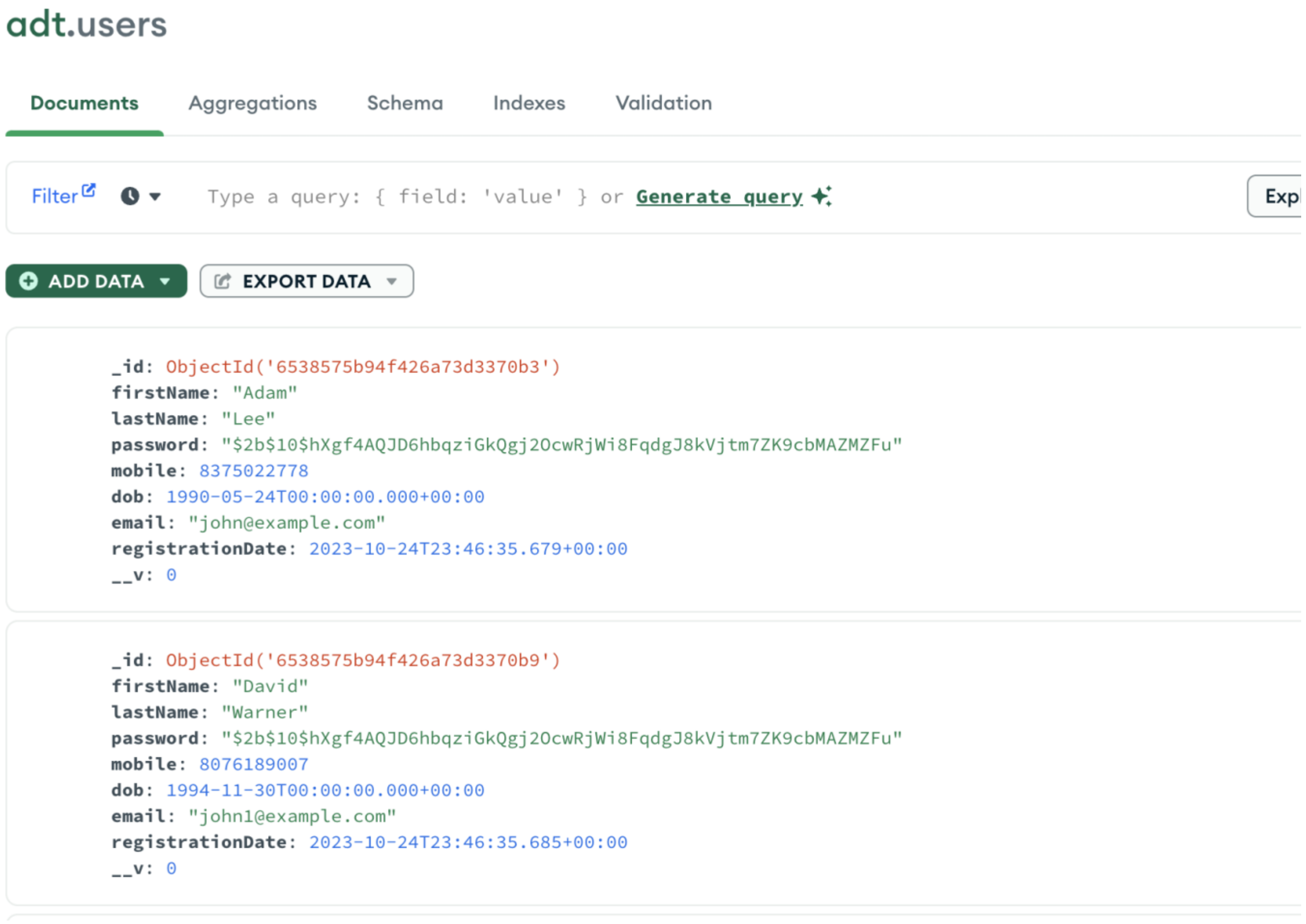
```
_id: ObjectId('653859c5993743130827d9d8')
title: "Sample Blog 3"
content: "Content for Blog 3"
```



```
_id: ObjectId('6538575b94f426a73d3370b7')
commenterName: "Adam Lee"
text: "Great blog!"
blogID: ObjectId('6538575b94f426a73d3370b5')
createdAt: 2023-10-24T23:46:35.684+00:00
updatedAt: 2023-10-24T23:46:35.684+00:00
__v: 0
```

```
_id: ObjectId('6538575b94f426a73d3370bd')
commenterName: "David Warner"
text: "Great blog WOW!"
blogID: ObjectId('6538575b94f426a73d3370bb')
createdAt: 2023-10-24T23:46:35.687+00:00
updatedAt: 2023-10-24T23:46:35.687+00:00
__v: 0
```

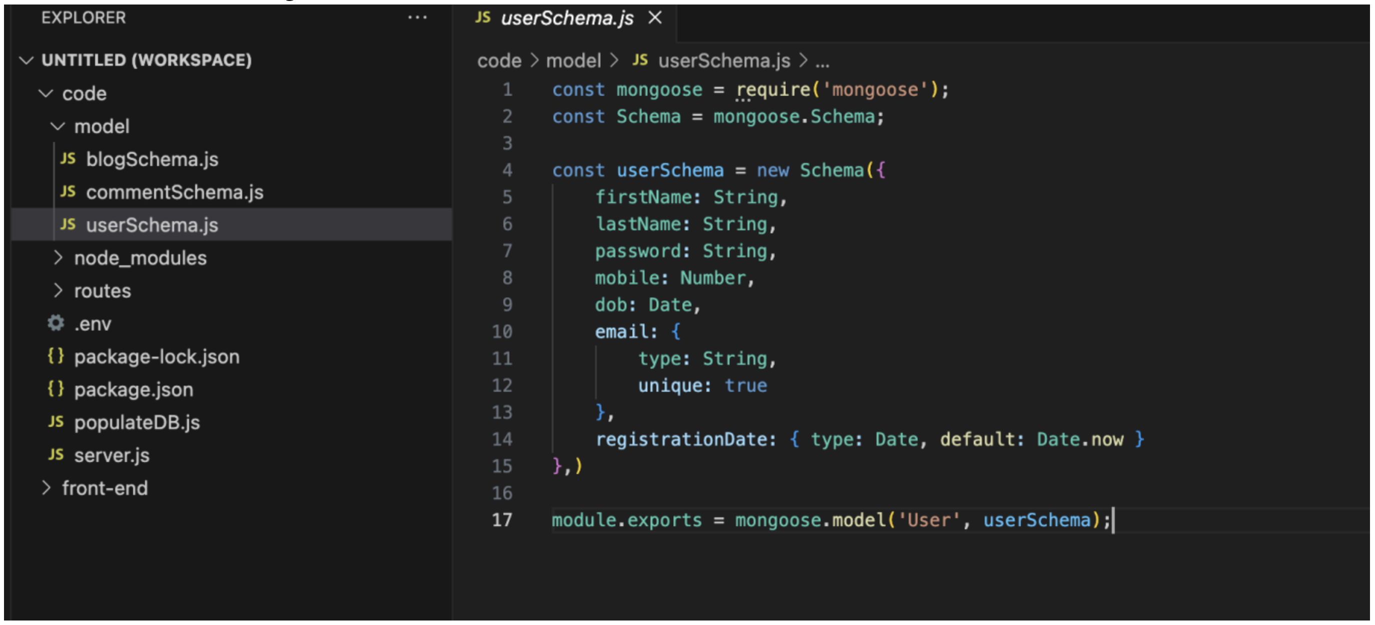
```
_id: ObjectId('6538575b94f426a73d3370bf')
commenterName: "Adam Lee"
text: "Great blog HURRAY!"
blogID: ObjectId('6538575b94f426a73d3370bb')
```



```
_id: ObjectId('6538575b94f426a73d3370b3')
firstName: "Adam"
lastName: "Lee"
password: "$2b$10$hXgf4AQJD6hbqziGkQgj2OcwRjWi8FqdgJ8kVjtm7ZK9cbMAZMZFu"
mobile: 8375022778
dob: 1990-05-24T00:00:00.000+00:00
email: "john@example.com"
registrationDate: 2023-10-24T23:46:35.679+00:00
__v: 0
```

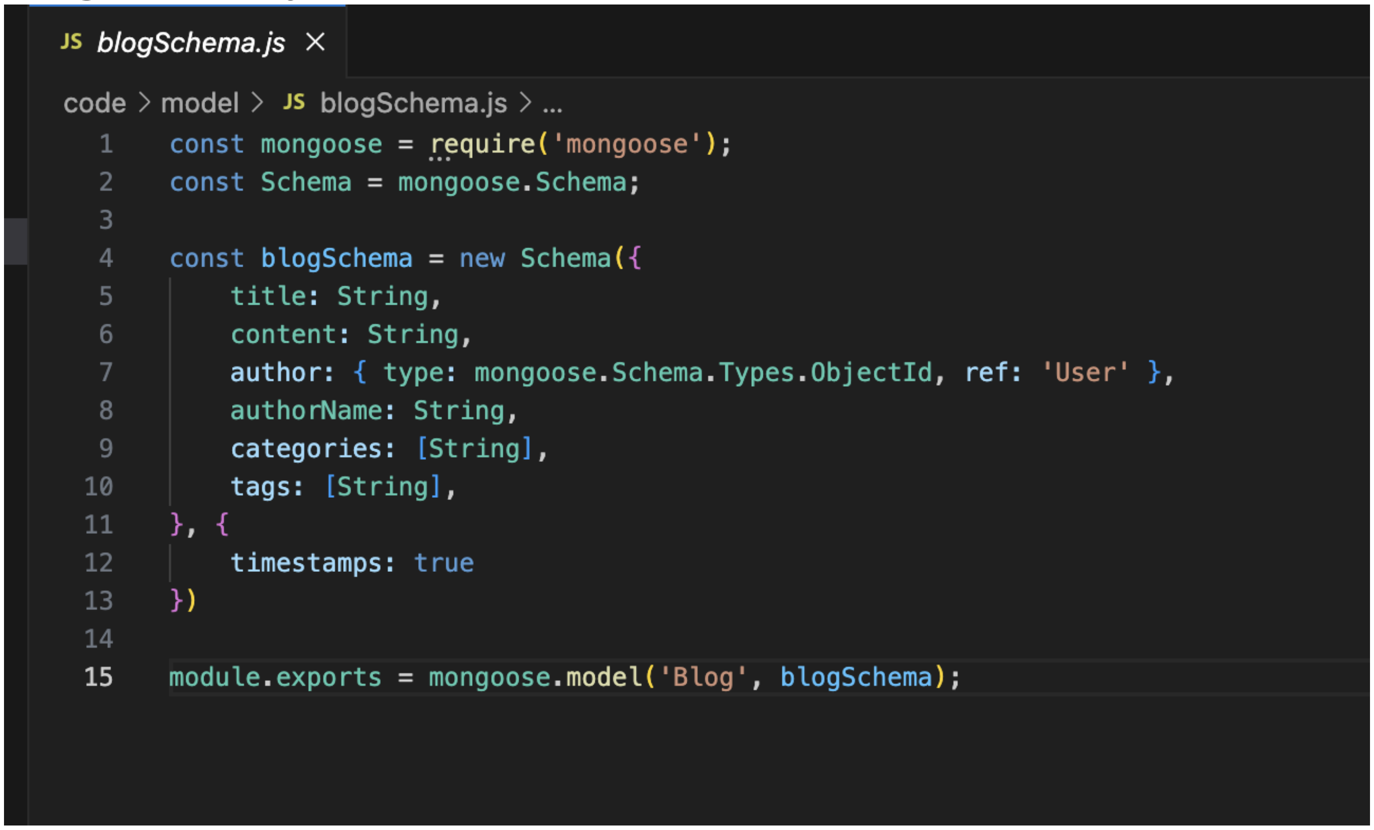
```
_id: ObjectId('6538575b94f426a73d3370b9')
firstName: "David"
lastName: "Warner"
password: "$2b$10$hXgf4AQJD6hbqziGkQgj2OcwRjWi8FqdgJ8kVjtm7ZK9cbMAZMZFu"
mobile: 8076189007
dob: 1994-11-30T00:00:00.000+00:00
email: "john1@example.com"
registrationDate: 2023-10-24T23:46:35.685+00:00
__v: 0
```

userSchema.js



```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const userSchema = new Schema({
    firstName: String,
    lastName: String,
    password: String,
    mobile: Number,
    dob: Date,
    email: {
        type: String,
        unique: true
    },
    registrationDate: { type: Date, default: Date.now }
},)

module.exports = mongoose.model('User', userSchema);
```

blogSchema.js

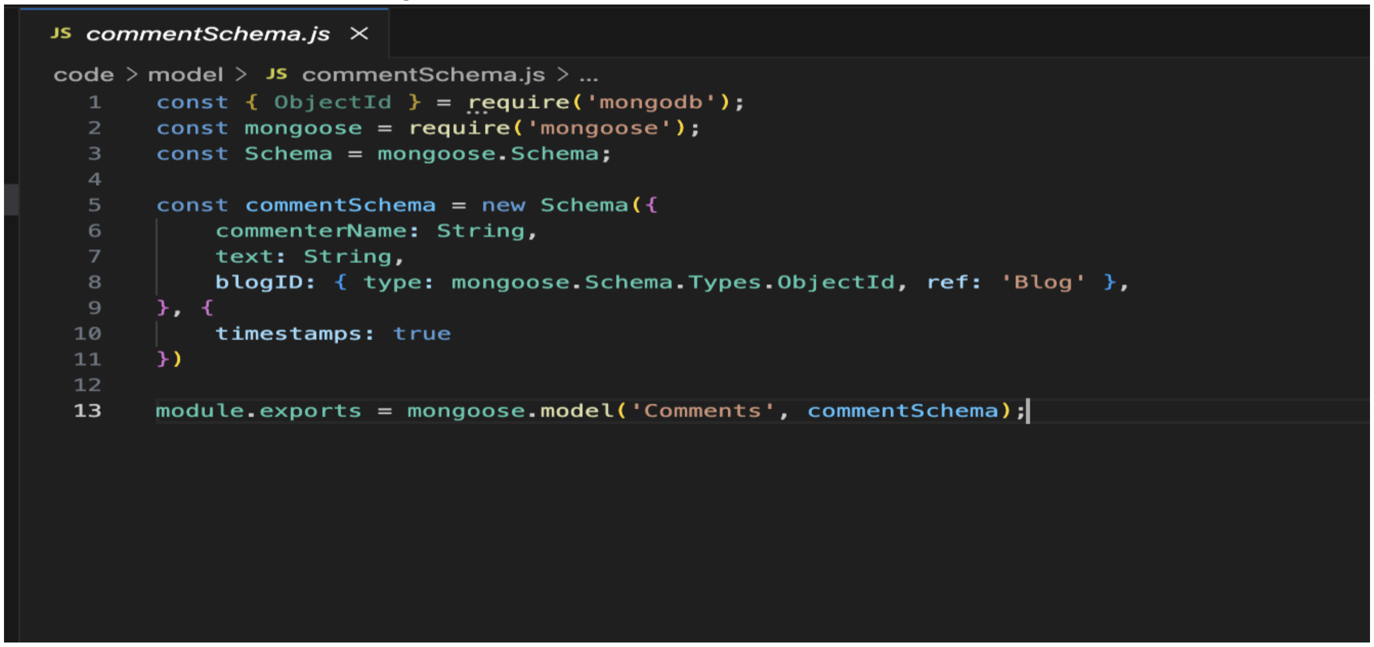


JS blogSchema.js ×

code > model > JS blogSchema.js > ...

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const blogSchema = new Schema({
    title: String,
    content: String,
    author: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
    authorName: String,
    categories: [String],
    tags: [String],
}, {
    timestamps: true
})

module.exports = mongoose.model('Blog', blogSchema);
```

commentSchema.js



JS commentSchema.js ×

code > model > JS commentSchema.js > ...

```
const { ObjectId } = require('mongodb');
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const commentSchema = new Schema({
    commenterName: String,
    text: String,
    blogID: { type: mongoose.Schema.Types.ObjectId, ref: 'Blog' },
}, {
    timestamps: true
})

module.exports = mongoose.model('Comments', commentSchema);
```

blog.js

```javascript
const express = require('express');
const router = express.Router();
const Blog = require('../model/blogSchema');
const User = require('../model/userSchema');
const Comment = require('../model/commentSchema');

// GET all blogs with optional filtering by tags and categories
router.get('/all', async (req, res) => {
    try {
        const { tags, categories } = req.query;
        const filter = {};

        if (tags) {
            filter.tags = { $in: tags.split(',') };
        }

        if (categories) {
            filter.categories = { $in: categories.split(',') };
        }

        const blogs = await Blog.find(filter);
        res.status(200).json(blogs);
    } catch (error) {
        console.error(error);
        res.status(500).json({ message: 'Unable to fetch the blogs' });
    }
});

// POST a new blog
router.post('/add', async (req, res) => {
    try {
        const { title, content, categories, tags } = req.body;
        const { user } = req.userData;

        const userDetails = await User.findOne({ email: user.email }).exec();

        const blogData = {
            title,
            content,
            categories,
            tags,
            authorName: `${userDetails.firstName} ${userDetails.lastName}`,
            author: userDetails.id,
        };

        const userBlog = await Blog.create(blogData);
        res.status(200).json(userBlog);
    } catch (error) {
        console.error(error);
        res.status(500).json({ message: 'Unable to upload the blog' });
    }
});

// DELETE a blog and its associated comments
router.delete('/delete', async (req, res) => {
    try {
        const id = req.query.id;

        await Blog.deleteOne({ _id: id }).exec();
        await Comment.deleteMany({ blogID: id }).exec();

        res.status(200).json({ message: 'Successfully deleted' });
    } catch (error) {
        console.error(error);
        res.status(500).json({ message: 'Unable to delete the blog' });
    }
});

// PUT (update) a blog
router.put('/update/:blogId', async (req, res) => {
    try {
        const { title, content, categories, tags } = req.body;
        const { blogId } = req.params;
        const { user } = req.userData;

        const userDetails = await User.findOne({ email: user.email }).exec();
        if (!userDetails) {
            return res.status(400).json({ message: 'User not found' });
        }

        const updatedData = {
            title,
            content,
            categories,
            tags,
            authorName: `${userDetails.firstName} ${userDetails.lastName}`,
            author: userDetails.id,
        };

        const updatedBlog = await Blog.findByIdAndUpdate(blogId, updatedData, { new: true }).exec();

        if (!updatedBlog) {
            return res.status(400).json({ message: 'Blog not found' });
        }

        res.status(200).json(updatedBlog);
    } catch (error) {
        console.error(error);
        res.status(500).json({ message: 'Unable to update the blog' });
    }
});

// POST a comment on a blog
router.post('/comment', async (req, res) => {
    try {
        const { comment, user, blogId } = req.body;
        const userDetails = await User.findOne({ email: user }).exec();

        const commentData = {
            text: comment,
            blogID: blogId,
            commenterName: `${userDetails.firstName} ${userDetails.lastName}`,
        };

        await Comment.create(commentData);
        res.status(200).json({ message: 'Successful' });
    } catch (error) {
        console.error(error);
        res.status(500).json({ message: 'Unable to add the comment' });
    }
});

// GET comments for a specific blog
router.get('/getcomments', async (req, res) => {
    try {
        const { id } = req.query;
        const commentsData = await Comment.find({ blogID: id });
        res.status(200).json(commentsData);
    } catch (error) {
        console.error(error);
        res.status(500).json({ message: 'Unable to fetch the comments' });
    }
});

module.exports = router;
```

user.js

```
const express = require('express');
const router = express.Router();
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const User = require('../model/userSchema');
const saltRounds = 10;

const ValidateEmail = (mail) => {
    // Use a regular expression to validate email.
    return /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/.test(mail);
};

const hashPassword = async (password) => {
    try {
        const hash = await bcrypt.hash(password, saltRounds);
        return hash;
    } catch (error) {
        console.error(error.message);
        throw error; // Rethrow the error for handling in the route handler.
    }
};

const compareHashPassword = async (password, hashPassword) => {
    try {
        const isValid = await bcrypt.compare(password, hashPassword);
        return isValid;
    } catch (error) {
        console.error(error.message);
        throw error; // Rethrow the error for handling in the route handler.
    }
};

router.post('/register', async (req, res) => {
    try {
        const { password, confirmPassword, firstName, lastName, email, mobile, DOB } = req.body;

        if (!email || !ValidateEmail(email)) {
            return res.status(400).json({ message: 'Invalid Email Address' });
        }

        if (!password || !confirmPassword || password !== confirmPassword) {
            return res.status(400).json({ message: 'Passwords do not match' });
        }

        if (!mobile || !Number.isInteger(mobile)) {
            return res.status(400).json({ message: 'Invalid Mobile Number' });
        }

        if (!firstName || !lastName) {
            return res.status(400).json({ message: 'First Name or Last Name missing' });
        }

        console.log(DOB);
        if (new Date(DOB) == 'Invalid Date') {
            return res.status(400).json({ message: 'Invalid Date of Birth' });
        }

        const existingUser = await User.findOne({ email }).exec();
        if (existingUser) {
            return res.status(409).json({ message: 'User Already Exists' });
        }

        const newPassword = await hashPassword(password);
        const newUserDetails = {
            firstName,
            lastName,
            mobile,
            password: newPassword,
            email,
            dob: DOB,
        };

        const user = await User.create(newUserDetails);
        res.status(200).json(user);
    } catch (error) {
        console.error(error.message);
        res.status(500).json({ message: 'Unable to register' });
    }
});

router.post('/login', async (req, res) => {
    try {
        const { password, email } = req.body;

        if (!email) {
            return res.status(401).json({ message: 'Email missing' });
        }

        if (!password) {
            return res.status(401).json({ message: 'Password missing' });
        }

        if (!ValidateEmail(email)) {
```

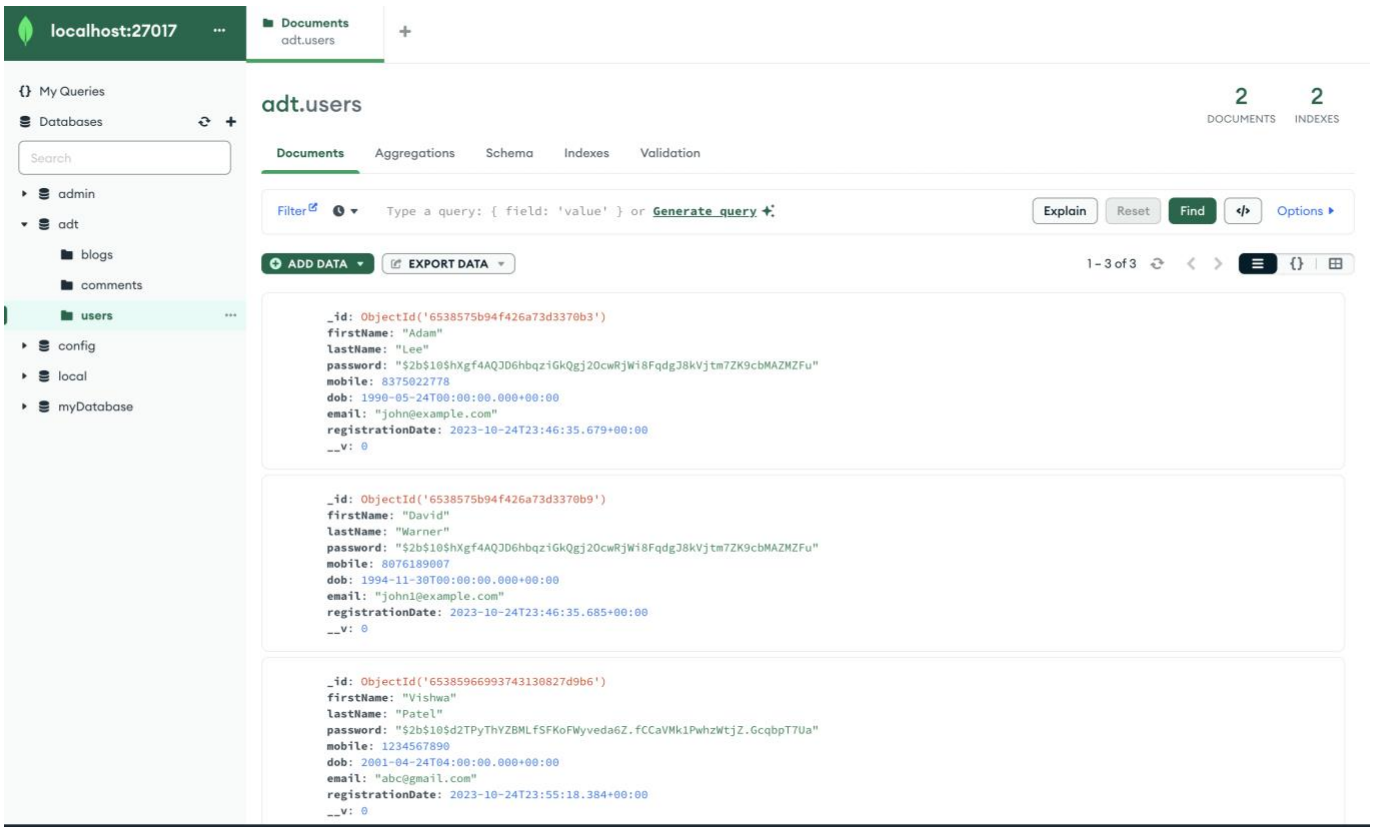
```
      return res.status(401).json({ message: 'Invalid Email Address' });
    }

    const existingUser = await User.findOne({ email }).exec();
    if (!existingUser) {
      return res.status(401).json({ message: 'User does not exist' });
    }

    const isValid = await compareHashPassword(password, existingUser.password);
    if (!isValid) {
      return res.status(401).json({ message: 'Wrong Password' });
    }

    const user = {
      email,
      password,
    };

    jwt.sign({ user }, process.env.JWT_SECRET, { expiresIn: '1h' }, (err, token) => {
      if (err) {
        console.error(err.message);
        res.status(401).json({ message: 'Unable to Login' });
      } else {
        res.status(200).json({ token });
      }
    });
  } catch (error) {
    console.error(error.message);
    res.status(401).json({ message: 'Unable to Login' });
  }
});

module.exports = router;
```

server.js

```
const express = require('express');
const mongoose = require('mongoose');
const userRoute = require('./routes/user');
const blogRoute = require('./routes/blog');
const jwt = require('jsonwebtoken');
const cors = require('cors');
const User = require('./model/userSchema');
const Blog = require('./model/blogSchema');
require('dotenv').config();

const app = express();

app.use(cors());  // Enable CORS for all routes
app.use(express.json());

const connect = async () => {
    try {
        mongoose.connect(process.env.MONGO_URL, {
            useUnifiedTopology: true,
            useNewUrlParser: true
        });
        console.log("Connected to mongodb");
    } catch (error) {
        console.log(`error while connecting mongodb : ${error}`);
    }
}

connect();

// Routes

const verifyToken = (req, res, next) => {
    const bearerHeader = req.headers['authorization'];
    if (typeof bearerHeader !== 'undefined') {
        const bearer = bearerHeader.includes(' ') ? bearerHeader.split(' ') : [null, bearerHeader];
        const bearerToken = bearer[1];
        jwt.verify(bearerToken, process.env.JWT_SECRET, (err, data) => {
            if (err) {
                res.sendStatus(403).json({ message: "user not logged in" });
            } else {
                req.userData = data;
                next();
            }
        });
    } else {
        res.sendStatus(403).json({ message: "user not logged in" });
    }
};

app.use('/', userRoute);
app.use('/blog', verifyToken, blogRoute);

app.post('/getuser', verifyToken, async (req, res) => {
    try {
        const { userId } = req.body;
        const userDetails = await User.findOne({ _id: userId }).exec();
        res.status(200).json({
            fullName: `${userDetails?.firstName} ${userDetails?.lastName}`
        });
    } catch (error) {
        console.log(error);
        res.status(500).json({ message: 'Unable to find the user' });
    }
})

app.get('/checkuser', verifyToken, async (req, res) => {
    try {
        const id = req.query.blogId;
        const blogDetails = await Blog.findOne({ _id: id }).exec();
        const userId = blogDetails.author;
        const userDetails = await User.findOne({ _id: userId }).exec();
        res.status(200).json({ email: userDetails.email });
    } catch (error) {
        console.log(error);
        res.status(500).json({ message: 'Unable to find the user details' });
    }
})

app.get('/', (req, res) => {
    res.send("Welcome to ADT Assignment");
```

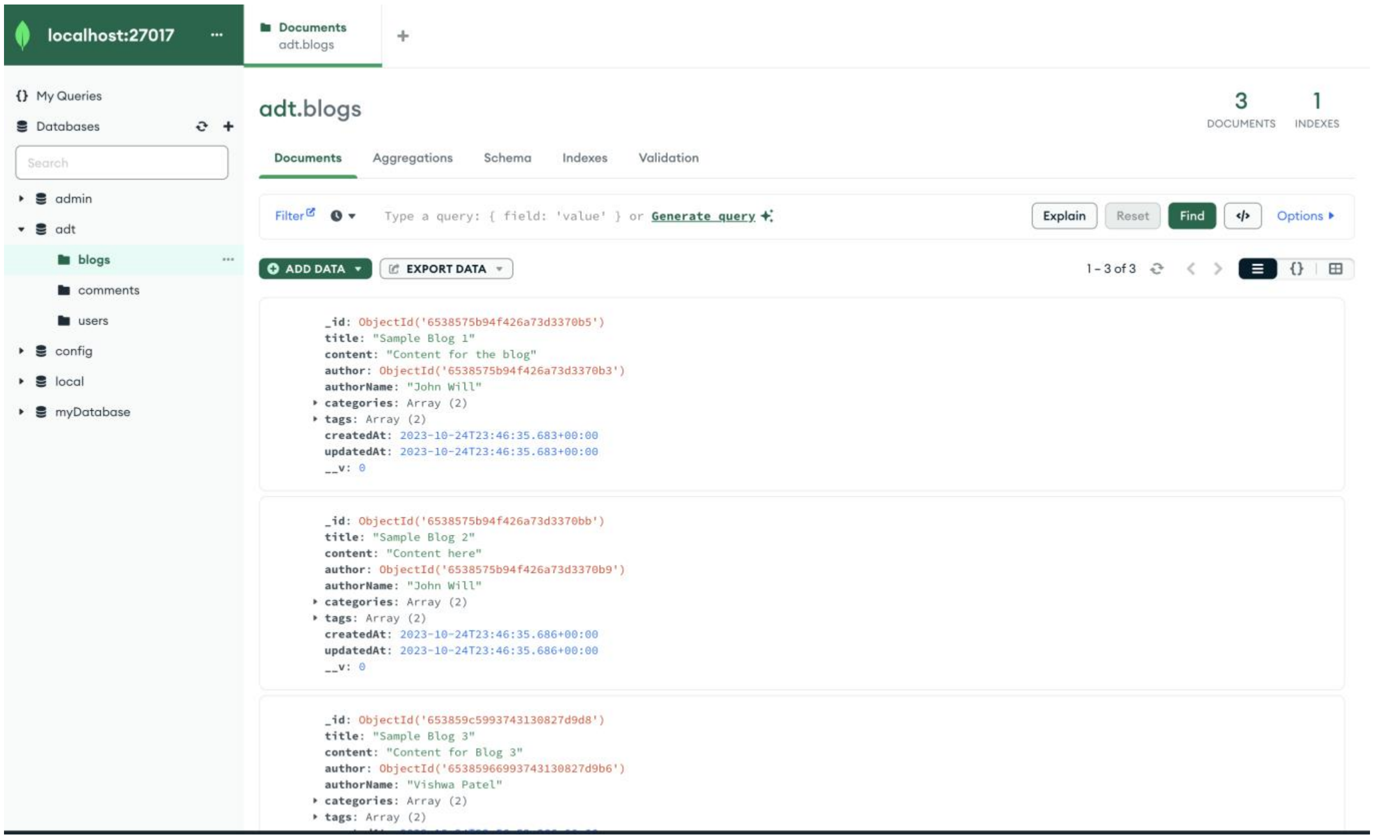
```
})



mongoose.connection.once('open', () => {
   app.listen(8000, () => {
      console.log("Server running on port 8000");
   });
})
```

Users collection



```
_id: ObjectId('6538575b94f426a73d3370b3')
firstName: "Adam"
lastName: "Lee"
password: "$2b$10$hXgf4AQJD6hbqziGkQgj2OcwRjWi8FqdgJ8kVjtm7ZK9cbMAZMZFu"
mobile: 8375022778
dob: 1990-05-24T00:00:00.000+00:00
email: "john@example.com"
registrationDate: 2023-10-24T23:46:35.679+00:00
__v: 0

_id: ObjectId('6538575b94f426a73d3370b9')
firstName: "David"
lastName: "Warner"
password: "$2b$10$hXgf4AQJD6hbqziGkQgj2OcwRjWi8FqdgJ8kVjtm7ZK9cbMAZMZFu"
mobile: 8076189007
dob: 1994-11-30T00:00:00.000+00:00
email: "john1@example.com"
registrationDate: 2023-10-24T23:46:35.685+00:00
__v: 0

_id: ObjectId('653859669937431308 27d9b6')
firstName: "Vishwa"
lastName: "Patel"
password: "$2b$10$d2TPyThYZBMLfSFKoFWyveda6Z.fCCaVMk1PwhzWtjZ.GcqbpT7Ua"
mobile: 1234567890
dob: 2001-04-24T04:00:00.000+00:00
email: "abc@gmail.com"
registrationDate: 2023-10-24T23:55:18.384+00:00
__v: 0
```

Blogs Collection

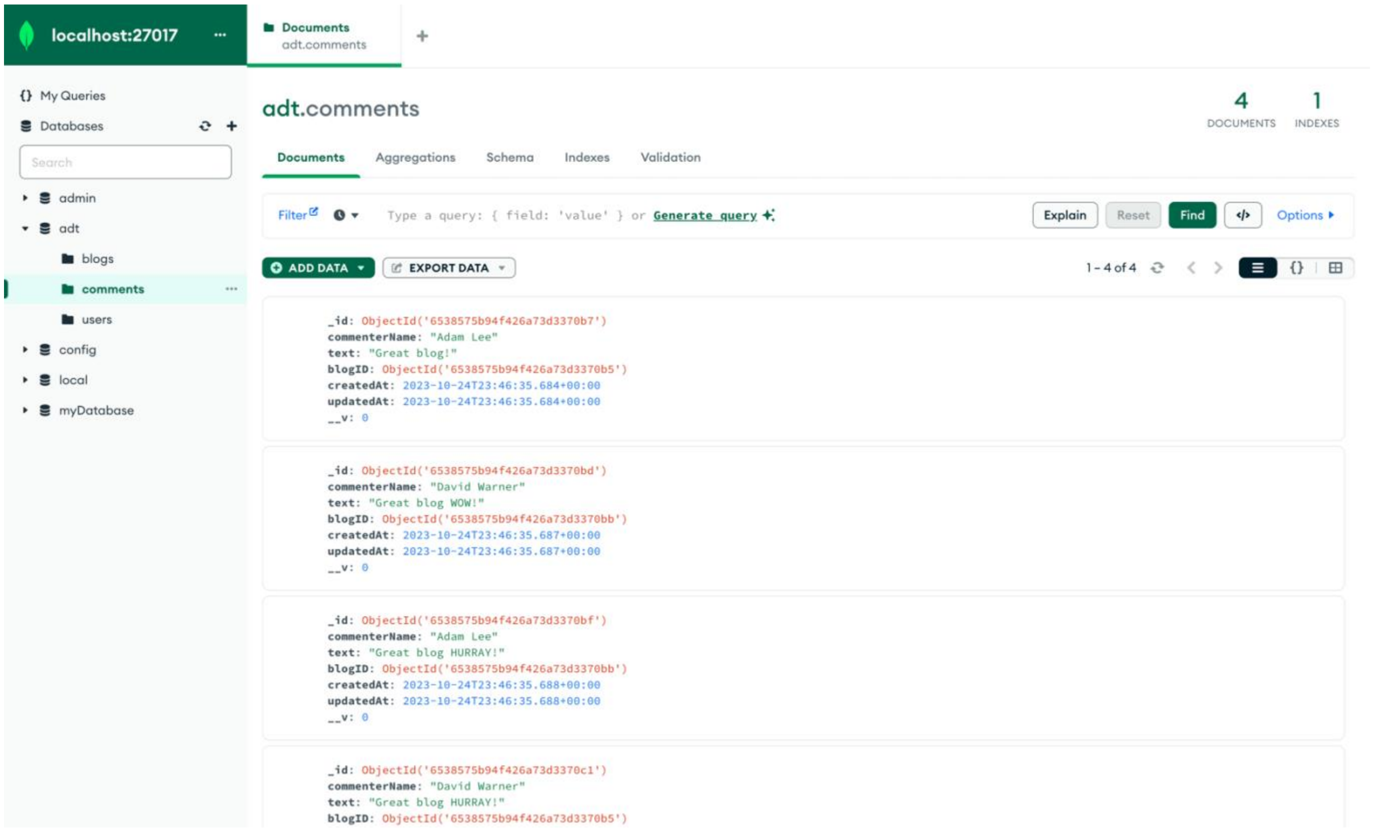


```
_id: ObjectId('6538575b94f426a73d3370b5')
title: "Sample Blog 1"
content: "Content for the blog"
author: ObjectId('6538575b94f426a73d3370b3')
authorName: "John Will"
categories: Array (2)
tags: Array (2)
createdAt: 2023-10-24T23:46:35.683+00:00
updatedAt: 2023-10-24T23:46:35.683+00:00
__v: 0
```

```
_id: ObjectId('6538575b94f426a73d3370bb')
title: "Sample Blog 2"
content: "Content here"
author: ObjectId('6538575b94f426a73d3370b9')
authorName: "John Will"
categories: Array (2)
tags: Array (2)
createdAt: 2023-10-24T23:46:35.686+00:00
updatedAt: 2023-10-24T23:46:35.686+00:00
__v: 0
```

```
_id: ObjectId('653859c5993743130827d9d8')
title: "Sample Blog 3"
content: "Content for Blog 3"
author: ObjectId('653859669937431308 27d9b6')
authorName: "Vishwa Patel"
categories: Array (2)
tags: Array (2)
```

Comments Collection



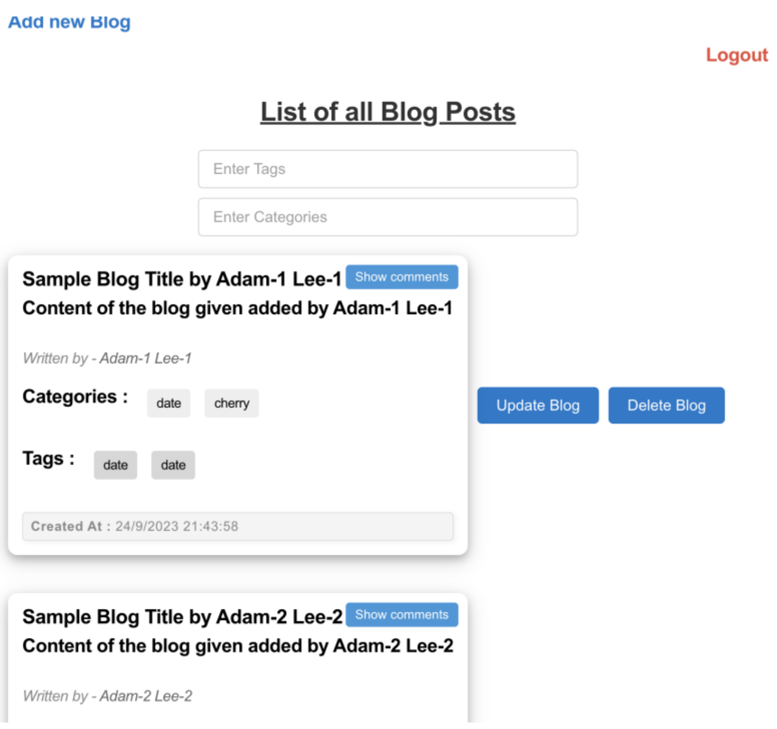
```
_id: ObjectId('6538575b94f426a73d3370b7')
commenterName: "Adam Lee"
text: "Great blog!"
blogID: ObjectId('6538575b94f426a73d3370b5')
createdAt: 2023-10-24T23:46:35.684+00:00
updatedAt: 2023-10-24T23:46:35.684+00:00
__v: 0

_id: ObjectId('6538575b94f426a73d3370bd')
commenterName: "David Warner"
text: "Great blog WOW!"
blogID: ObjectId('6538575b94f426a73d3370bb')
createdAt: 2023-10-24T23:46:35.687+00:00
updatedAt: 2023-10-24T23:46:35.687+00:00
__v: 0

_id: ObjectId('6538575b94f426a73d3370bf')
commenterName: "Adam Lee"
text: "Great blog HURRAY!"
blogID: ObjectId('6538575b94f426a73d3370bb')
createdAt: 2023-10-24T23:46:35.688+00:00
updatedAt: 2023-10-24T23:46:35.688+00:00
__v: 0

_id: ObjectId('6538575b94f426a73d3370c1')
commenterName: "David Warner"
text: "Great blog HURRAY!"
blogID: ObjectId('6538575b94f426a73d3370b5')
```

Script to generate random users, blogs and comments

```
const mongoose = require('mongoose');
const User = require('./model/userSchema');
const Blog = require('./model/blogSchema');
const Comment = require('./model/commentSchema');
const bcrypt = require("bcrypt");
const saltRounds = 10

const hashPassword = async (password) => {
  try {
    const hash = await bcrypt.hash(password, saltRounds);
    return hash;
  } catch (error) {
    console.error(err.message);
  }
}
```

```
function getRandomStringFromArray(stringArray) {

    // Generate a random index between 0 and the length of the array
    const randomIndex = Math.floor(Math.random() * stringArray.length);

    // Return the random string from the array
    return stringArray[randomIndex];
}

const myArray = ["apple", "banana", "cherry", "date"];

async function populateData() {
    try {
        await mongoose.connect('mongodb://localhost:27017/adt', {
            useUnifiedTopology: true,
            useNewUrlParser: true
        });
        console.log("Connected to mongodb");
        console.log("Populating the Data");

        const newPassword = await hashPassword('password');

        for (let i = 1; i < +100; i++) {
            // sample users
            const user1 = await User.create({
                firstName: `Adam-${i}`,
                lastName: `Lee-${i}`,
                email: `john${i}@example.com`,
                password: newPassword,
                mobile: 8375022778,
                dob: '1990-05-24'
            });

            getRandomStringFromArray(myArray);

            // sample blog posts
            const blog = await Blog.create({
                title: `Sample Blog Title by Adam-${i} Lee-${i}`,
                content: `Content of the blog given added by Adam-${i} Lee-${i}`,
            author: user1._id,
            authorName: `Adam-${i} Lee-${i}`,
            tags: [getRandomStringFromArray(myArray), getRandomStringFromArray(myArray)],
            categories: [getRandomStringFromArray(myArray), getRandomStringFromArray(myArray)]
          });

          // sample comments
          const comment1 = await Comment.create({
            text: `Great blog by Adam-${i} Lee-${i}`,
            blogID: blog._id,
            commenterName: `Adam-${i} Lee-${i}`
          });
        }

        console.log("Date Populated");
        mongoose.disconnect();
    } catch (error) {
        console.log(`error while connecting mongodb : ${error}`);
    }
}

populateData();
```

UI Screenshots


Add new Blog

Logout

List of all Blog Posts

Enter Tags

Enter Categories

Sample Blog Title by Adam-1 Lee-1 Show comments
Content of the blog given added by Adam-1 Lee-1

Written by - Adam-1 Lee-1

Categories : date cherry

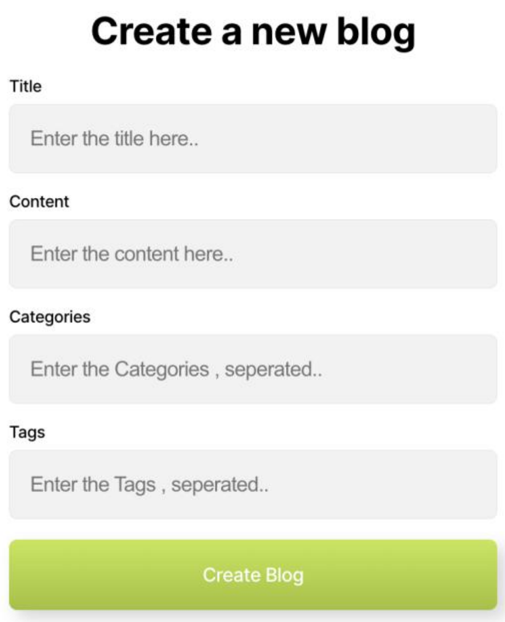
Tags : date date

Created At : 24/9/2023 21:43:58

Update Blog Delete Blog

Sample Blog Title by Adam-2 Lee-2 Show comments
Content of the blog given added by Adam-2 Lee-2

Written by - Adam-2 Lee-2


Create a new blog

Title

Enter the title here..

Content

Enter the content here..

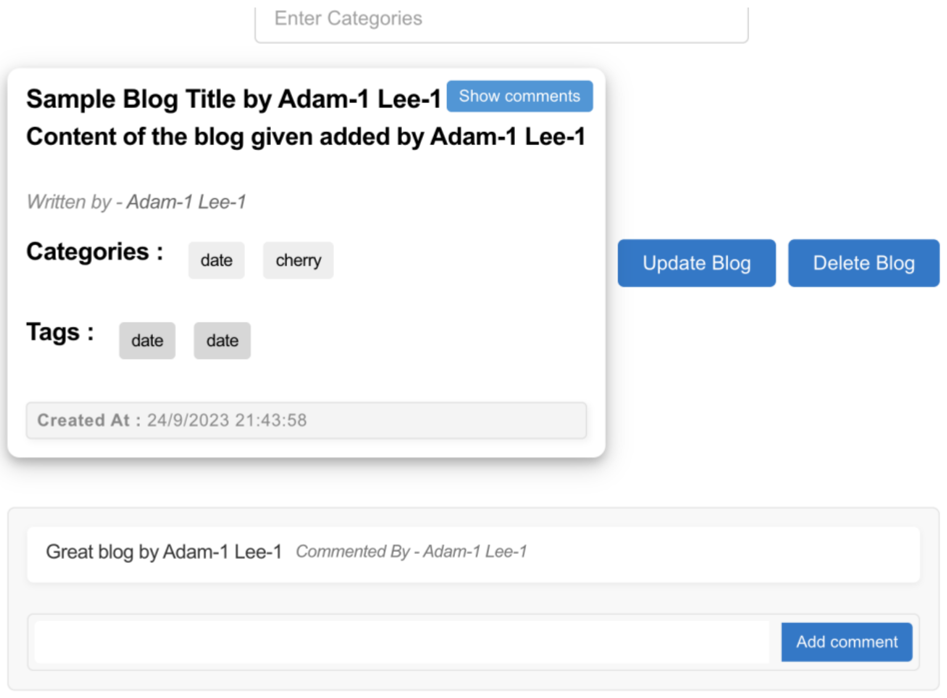
Categories

Enter the Categories , seperated..

Tags

Enter the Tags , seperated..

Create Blog


Enter Categories

Sample Blog Title by Adam-1 Lee-1 Show comments

Content of the blog given added by Adam-1 Lee-1

Written by - Adam-1 Lee-1

Categories : date cherry

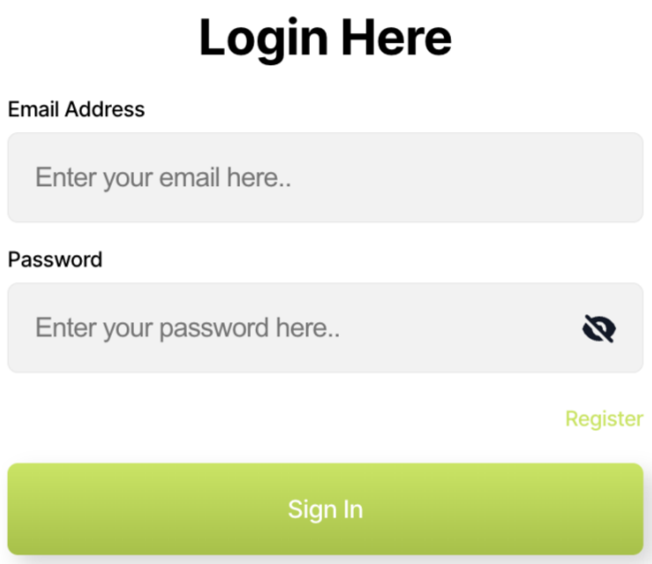
Tags : date date

Created At : 24/9/2023 21:43:58

Update Blog Delete Blog

Great blog by Adam-1 Lee-1 *Commented By - Adam-1 Lee-1*

Add comment


Login Here

Email Address

Enter your email here..

Password

Enter your password here..

Register

Sign In