**ASSIGNMENT-2**

**CODE:**

```
const express = require('express');
const bodyParser = require('body-parser');
const { MongoClient } = require("mongodb")
const app = express();
const PORT = 3000;
dbname = "Blog"

// Middleware to parse JSON bodies
app.use(bodyParser.json());

// MongoDB URI
const uri = "mongodb://127.0.0.1"
const client = new MongoClient(uri)

// Function to connect to the MongoDB database
async function connectToDatabase() {
    try {
        await client.connect()
        console.log("Connected to MongoDB ...")
    } catch (error) {
        console.error(`error connecting to {database}`)
    }
}

// GET - /posts - Fetch all blog posts
app.get('/posts', async (req, res) => {
    await client.connect();
```

```
    db = client.db(dbname)
    const collection = db.collection('posts');
    try {
        const posts = await collection.find({}).toArray();
        res.json(posts);
        console.log("Fetched all the blog posts")
    client.close();
    } catch (err) {
        console.log(err);
        res.status(500).send("Failed to fetch posts.");
    }
});

// POST - /posts - Add a new blog post
app.post('/posts', async (req, res) => {
    // Check if request body exists and is not empty
    if (!req.body || Object.keys(req.body).length === 0) {
        return res.status(400).send('Request body is empty or not valid
JSON.');
    }

    await client.connect();
    db = client.db(dbname)
    const collection = db.collection('posts');

    try {
        const result = await collection.insertOne(req.body);
        if (result.acknowledged) {
            res.status(200).json({ 'Message': 'Post added successfully' });
            console.log("Added the new blog post successfully")
        } else {
            res.status(400).send('Adding new post failed');
```

```
        }
        client.close();
    } catch (err) {
        console.log(err);
        res.status(500).send('Adding new post failed');
    }
});

// GET - /posts/user/:userName - Retrieve all blog posts by a given username
app.get('/posts/user/:userName', async (req, res) => {
    try {
        await client.connect();
        db = client.db(dbname)
        const userName = req.params.userName;
        const collection = db.collection('posts');
        const posts = await collection.find({ Username: userName }).toArray();
        res.json(posts);
        console.log("All the blog posts added by the "+{userName}+ "are
fetched")
        console.log(posts)
        client.close();
    } catch (error) {
        res.status(500).json({ message: error.message });
    }
});

// GET - /posts/title/:searchWord - Retrieve all blog posts containing a given
word in the title
app.get('/posts/title/:searchWord', async (req, res) => {
    try {
        const searchWord = req.params.searchWord;
        await client.connect();
```

```
        const db = client.db(dbname);
        const collection = db.collection('posts');


        const posts = await collection.find({ Title: { $regex: searchWord, $options: 'i' } }).toArray();
        res.json(posts);
        console.log("Blogs containing the given keyword of title are fetched");


        client.close();
    } catch (error) {
        res.status(500).json({ message: error.message });
    }
});


// GET - /posts/content/:searchWord - Retrieve all blog posts containing a given word in the content
app.get('/posts/content/:searchWord', async (req, res) => {
    try {
        await client.connect();
        const db = client.db(dbname);
        const searchWord = req.params.searchWord;
        const collection = db.collection('posts');


        const posts = await collection.find({ Content: { $regex: searchWord, $options: 'i' } }).toArray();
        res.json(posts);
        console.log("Blogs containing the given keyword in content are fetched")


        client.close();
    } catch (error) {
        res.status(500).json({ message: error.message });
```

```
    }
});

// DELETE - /posts/user/:userName - Delete all blogs by a given username
app.delete('/posts/user/:userName', async (req, res) => {
    try {
        const userName = req.params.userName;
        await client.connect();
        const db = client.db(dbname);
        const collection = db.collection('posts');

        const result = await collection.deleteMany({ Username: userName });
        res.json({ message: `${result.deletedCount} post(s) deleted` });

        client.close();
    } catch (error) {
        res.status(500).json({ message: error.message });
    }
});

// DELETE - /posts/content/:searchWord - Delete all blogs that contain a
certain word in the content
app.delete('/posts/content/:searchWord', async (req, res) => {
    try {
        const searchWord = req.params.searchWord;
        await client.connect();
        const db = client.db(dbname);
        const collection = db.collection('posts');

        const result = await collection.deleteMany({ Content: { $regex:
searchWord, $options: 'i' } });
        res.json({ message: `${result.deletedCount} post(s) deleted` });
```

```
        client.close();
    } catch (error) {
        res.status(500).json({ message: error.message });
    }
});

// GET - /posts/count/:userName - Returns the username and the number of blogs posted by that username
app.get('/posts/count/:userName', async (req, res) => {
    try {
        await client.connect();
        const db = client.db(dbname);
        const userName = req.params.userName;
        const collection = db.collection('posts');

        // Counting the number of posts by the given username
        const count = await collection.countDocuments({ Username: userName });

        res.json({ userName, postCount: count });

        client.close();
    } catch (error) {
        res.status(500).json({ message: error.message });
    }
});

// Start the Express server
app.listen(PORT, async () => {
    console.log(`Connected to the Express server on port ${PORT}`)
    await connectToDatabase()
});
```

OUTPUT:

PS C:\Users\HARSHA\OneDrive\Desktop\Cloud Computing\Assignment-2> node app.js

Connected to the Express server on port 3000

Connected to MongoDB ...

Fetched all the blog posts

Added the new blog post successfully

Fetched all the blog posts

All the blog posts added by the [object Object]are fetched


[

{

_id: new ObjectId('65e15681ebed54ecebfdbb39'),

Username: 'Mike',

Title: 'The Importance of Data Privacy',

Content: 'Data privacy is a critical aspect of modern technology. In this blog post, we discuss why data privacy matters, its implications for individuals and businesses, and best practices for ensuring data privacy.'

},

{

_id: new ObjectId('65e156d1ebed54ecebfdbb3e'),

Username: 'Mike',

Title: 'Second blog',

Content: 'This is a second sample blog post'

},

{

_id: new ObjectId('65e157f3bd8ebc23af331ba9'),

Username: 'Mike',

Title: 'Third blog',

Content: 'This is a second sample blog post'

}

]

All the blog posts added by the [object Object]are fetched

[

{

_id: new ObjectId('65e1566febed54ecebfdbb38'),

Username: 'Harsha',

Title: 'Exploring JavaScript Frameworks',

Content: "In this blog post, we'll dive into some popular JavaScript frameworks like React, Angular, and Vue.js, exploring their features, advantages, and use cases."

},

{

_id: new ObjectId('65e156bbebed54ecebfdbb3c'),

Username: 'Harsha',

Title: 'The Future of Artificial Intelligence',

Content: 'Artificial intelligence (AI) is reshaping industries and society as a whole. In this blog post, we explore the latest advancements in AI technology, ethical considerations, and potential future developments.'

},

{

_id: new ObjectId('65e156c8ebed54ecebfdbb3d'),

Username: 'Harsha',

Title: 'First blog',

Content: 'This is a sample first blog'

}

]

Blogs containing the given keyword of title are fetched

Blogs containing the given keyword in content are fetched

All the blog posts added by the [object Object]are fetched

[

{

_id: new ObjectId('65e156d1ebed54ecebfdbb3e'),

```
  Username: 'Mike',
  Title: 'Second blog',
  Content: 'This is a second sample blog post'
 },
 {
  _id: new ObjectId('65e157f3bd8ebc23af331ba9'),
  Username: 'Mike',
  Title: 'Third blog',
  Content: 'This is a second sample blog post'
 }
]
```

README.md file

```
T
# app.js Blog API

This is a simple Express.js RESTful API for managing a blog system. It
provides endpoints for creating, reading, updating, and deleting blog posts.

## Installation

1. Clone this repository.
2. Install dependencies using `npm install`.
3. Make sure you have MongoDB installed and running on your local machine.

## Configuration

1. Set up your MongoDB URI in the `uri` variable in `app.js`.
2. Ensure that MongoDB is running locally or provide the appropriate URI for
your MongoDB instance.

## Running the Server
```

To start the server, run:

```
node app.js
```

The server will start running on port 3000 by default.

## API Endpoints

### GET /posts

- Retrieves all blog posts.

### POST /posts

- Adds a new blog post with a title, content, and username.

### GET /posts/user/:userName

- Retrieves all blog posts by a given username.

### GET /posts/title/:searchWord

- Retrieves all blog posts containing a given word in the title.

### GET /posts/content/:searchWord

- Retrieves all blog posts containing a given word in the content.

### DELETE /posts/user/:userName

- Deletes all blogs by a given username.

### DELETE /posts/content/:searchWord

- Deletes all blogs that contain a certain word in the content.

### GET /posts/count/:userName

- Returns the username and the number of blogs posted by that username.

## Usage

- Use any API testing tool like Postman or cURL to interact with the API endpoints.
- Ensure proper authentication and authorization mechanisms are implemented in a production environment.