**PROJECT NAME:**

API for list of users and their posts and comments.

**INTRODUCTION:**

REST API for store and retrieve list of users, posts and comments from the database (MONGODB).

**REQUIREMENTS:**

* + - Visual studio code
    - Postman
    - MongoDB
    - Node js

**INSTALLATION:**

* Install visual studio code from this link -> <https://code.visualstudio.com/download>.
* Install postman from this link -> <https://www.postman.com/downloads>
* Install mongoDB from this link -> <https://www.mongodb.com/try/download/community>
* Install node js from this link -> <https://nodejs.org/en/download>

**CONFIGURATION:**

**NPM:**

* After installing node js , initiate npm by ***npm init*** in VS code terminal.

**Visual studio code:**

* After installing the VS code, install required packages.
* For our project the required packages are,
  + **Express** ( *npm i express* to install the package).
  + **Mongoose** ( *npm i mongoos*e to install the package).

**MongoDB:**

* After installing mongoDB , Create a database using cmd or mongoDB compass.
* Create a database by giving the required informations (in mongoDB compass).

**Program:**

Create required schemas:

* User Schema
* Post schema
* Comment Schema

Now create routes,

1. **Add new user**

const express = require('express');

const router = express.Router();

const User = require('../model/User');

const Post = require('../model/Post');

const Comment = require('../model/Comment');

// add a new user

router.post('/', async (req, res) => {

    const newUser = new User(req.body);

    try {

        await newUser.save();

        res.status(201).send(newUser);

    } catch (err) {

        res.status(500).send();

    }

});

1. **Get all users**

//get all users

router.get('/', async (req, res) => {

    try {

        const users = await User.find({});

        res.send(users);

    } catch (error) {

        res.status(404).send({ error: 'Path not found' });

    }

});

1. **Get user by ID**

router.get('/:id', async (req, res) => {

    const userId = req.params.id;

    try {

        const user = await User.findById(userId);

        if (!user) {

            return res.status(404).send({ error: 'User not found' });

        }

        res.send(user);

    } catch (error) {

        res.status(500).send({ error: 'Internal server error' });

    }

});

1. **Update user by ID**

//update a user

router.patch('/:id', async (req, res) => {

    const updates = Object.keys(req.body);

    const allowedUpdates = ['name','phone','username'];

    const isValidOperation = updates.every((update) => {

        return allowedUpdates.includes(update);

    });

    if (!isValidOperation) {

        return res.status(400).send({ error: 'Invalid Operation' });

    }

    try {

        const user = await User.findById(req.params.id);

        if (!user) {

            return res.status(404).send({ error: 'User not found' });

        }

        updates.forEach((update) => {

            user[update] = req.body[update];

        });

        await user.save();

        res.send(user);

    } catch (error) {

        res.status(500).send({ error: 'Internal server error' });

    }

});

1. **Delete user by ID**

// delete a user

router.delete('/:id', async (req, res) => {

    try {

        const user = await User.findByIdAndDelete(req.params.id);

        if (!user) {

            return res.status(404).send({ error: 'User not found' });

        }

        res.send(user);

    } catch (error) {

        res.status(500).send({ error: 'Internal server error' });

    }

});

1. **Add a new post**

router.post('/:id/posts', async (req, res) => {

    const newPost = new Post(req.body);

    const userId = req.params.id;

    try {

        const user = await User.findById(userId);

        if (!user) {

            return res.status(404).send({ error: 'User not found' });

        }

        newPost.userId = userId;

        await newPost.save();

        res.status(201).send(newPost);

    } catch (err) {

        res.status(500).send();

    }

});

1. **Get all posts**

router.get('/0/posts', async (req, res) => {

    try {

        const posts = await Post.find({});

        res.send(posts);

    } catch (error) {

        res.status(404).send({ error: 'Path not found' });

    }

});

1. **Delete a post from user**

router.delete('/:id/posts/:pid', async (req, res) => {

    const userId = req.params.id;

    const postId = req.params.pid;

    try {

        const user = await User.findById(userId);

        if (!user) {

            return res.status(404).send({ error: 'User not found' });

        }

        const post = await Post.findByIdAndDelete(postId);

        if (!post) {

            return res.status(404).send({ error: 'Post not found' });

        }

        res.send(post);

    } catch (error) {

        res.status(500).send({ error: 'Internal server error' });

    }

});

1. **Add new comment to a post**

router.post('/:id/posts/:pid/comments', async (req, res) => {

    const newCmnt = new Comment(req.body);

    const postId = req.params.pid;

    const userId = req.params.id;

    try {

        const user = await User.findById(userId);

        if (!user) {

            return res.status(404).send({ error: 'User not found' });

        }

        const post = await Post.findById(postId);

        if (!post) {

            return res.status(404).send({ error: 'Post not found' });

        }

        newCmnt.postId = postId;

        await newCmnt.save();

        res.status(201).send(newCmnt);

    } catch (err) {

        res.status(500).send();    }

});

1. **Delete a comment**

router.delete('/:id/posts/:pid/comments/:cid', async (req, res) => {

    const userId = req.params.id;

    const postId = req.params.pid;

    const cmntId = req.params.cid;

    try {

        const user = await User.findById(userId);

        if (!user) {

            return res.status(404).send({ error: 'User not found' });

        }

        const post = await Post.findById(postId);

        if (!post) {

            return res.status(404).send({ error: 'Post not found' });

        }

        const comment = await Comment.findById(cmntId);

        if (!comment) {

            return res.status(404).send({ error: 'Comment not found' });

        }

        res.send(comment);

    } catch (error) {

        res.status(500).send({ error: 'Internal server error' });    }

});

module.exports = router;

**Start Server:**

const express = require('express');

const mongoose = require('mongoose');

const users = require('./routes/user.js');

const db ='mongodb://127.0.0.1:27017';

const port = 4000;

const app = express();

app.use(express.json());

app.use('/', users);

mongoose

    .connect(db, {

        useNewUrlParser: true,

        useUnifiedTopology: true,

        useCreateIndex: true

    })

    .then(() => {

        console.log('MongoDB Connnected');

    })

    .catch((err) => {

        console.log({ err: err });

    });

app.listen(port, (err) => {

    if (err) {

        console.log({ err: err });

    }

    console.log('Server running on port ' + port);

});

Now the server running in port 4000 and database is connected.