**DOCUMENTATION**

**INSTALL:**

1. Nodejs installation
2. Npm initialization
3. Npm install express
4. Npm install mongoose
5. Npm install –D nodemon

**CRUD OPERATIONS:**

**CRUD:** Create,Read,Update,Delete.

**TO CREATE:**

**Server.js:**

const express = require("express");

const mongoose = require("mongoose");

const insta = require('./routers/insta.js');

const db ="mongodb+srv://vandhana\_1999:vandhana0904@cluster0.6mjxg.mongodb.net/insta?retryWrites=true&w=majority";

const port = 3000;

const app = express();

app.use(express.json())

mongoose.connect(db,{

useNewUrlParser: true,

useUnifiedTopology: true,

useCreateIndex: true

}).then(()=>{

console.log("mongodb connected");

}).catch((err)=>{

console.log({err: err });

})

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

res.send("hi")

})

app.use('/insta', insta);

app.listen(port,(err)=>{

if(err){

console.log( {err : err});

}

console.log("server running on port" +port);

});

**MODEL> Insta.js**

**(SCHEMA)**

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const commentschema=new Schema({

createdat:{

type:Date,

default:Date.now()

},

comment:{

type:String,

required:true

},

username:{

type:String,

required:true

},

})

const InstaShema = new Schema({

createdat:{

type:Date,

default:Date.now()

},

likes:{

type:String,

required:true

},

url:{

type:String,

required:true

},

caption:{

type:String,

required:true

},

username:{

type:String,

required:true

},

comments:[commentschema]

});

module.exports = Insta = mongoose.model('users', InstaShema);

**Routers>insta.js**

const express = require("express");

// const {create,update} = require('../model/user');

const router = express.Router();

const Insta = require(`../model/Insta`);

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

console.log('users route');

const newInsta = new Insta(req.body);

try {

await newInsta.save();

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

} catch (err) {

res.status(500).send();

}

});

**TO UPDATE:**

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

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

console.log(updates);

const allowedUpdates = ['likes', 'username'];

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

return allowedUpdates.includes(update);

});

if (!isValidOperation) {

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

}

try {

const Insta = await Insta.findById(req.params.id);

if (!Insta) {

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

}

updates.forEach((update) => {

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

});

await Insta.save();

res.send(Insta);

} catch (error) {

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

}

});

**TO DELETE:**

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

try {

const Insta = await Insta.findByIdAndDelete(req.params.id);

if (!Insta) {

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

}

res.send(Insta);

} catch (error) {

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

}

});

module.exports = router;

**DATABASE:**

