Crud Operation API

Aim: API created to perform crud operation of Event model

API Link: - https://github.com/rbhagwane123/CrudOperation-Api-with-image-upload

This is a simple Event API where crud operation Insert, Delete, Create, Paginate, Update of events as model.

API Layout

- node\_modules // projects all libraries file

- routes/routes // File contains the routes for operation to be performed

- controllers/controller // Folder containing REST Controllers (User Controller)

-models/events // File containing

- public //it store the images or files which are provided of insertion event.

- server.js/ // main file or connection file with database.

User Service (Url’s for API functionality calling)

| **HTTP Method** | **URL** | **Description** |
| --- | --- | --- |
| POST | <http://localhost:8000/api/v3/app/events> | Create new Event |
| PUT | <http://localhost:8000/api/v3/app/events/uid> | Update Event by \_ID |
| GET | <http://localhost:8000/api/v3/app/events?id=event_id> | Get Events by \_ID |
| GET | <http://localhost:8000/api/v3/app/events?type=latest&limit=3&page=1> | Get 3 limit of Events or Pagination |
| DELETE | <http://localhost:8000/api/v3/app/events/uid> | Delete Event by \_ID |

API Code:-

**Index.js**

const express = require('express');

const app = express()

const cors = require('cors');

const mongoose = require('mongoose')

app.use(cors({

    origin: '\*'

}));

mongoose.connect("mongodb://127.0.0.1:27017/event\_crud");

const post\_route = require('./routes/routes');

app.use('/api/v3/app', post\_route);

app.listen(8000, function(){

    console.log('Server started');

})

**routes/routes.js**

const express = require('express');

const post\_route = express();

post\_route.use(express.urlencoded({ extended: true }));

post\_route.use(express.json());

const multer = require('multer');

const path = require('path');

const controller = require('../controllers/controller');

post\_route.use(express.static('public'));

const storage = multer.diskStorage({

    destination: function (req, file, cb) {

        cb(null, path.join(\_\_dirname, '../public'), function (error, success) {

            if (error) {

                console.log(error);

            }

        });

    },

    filename: function (req, file, cb) {

        const name = Date.now() + '-' + file.originalname;

        cb(null, name, function (error, success) {

            if (error) {

                console.log(error);

            }

        });

    }

});

const upload = multer({ storage: storage });

post\_route.post('/events', upload.single('file'), controller.createEvent);

// post\_route.get('/events', controller.getUIDEvent);

post\_route.get('/events', controller.getEvent);

post\_route.put('/events/:uid', controller.updateUIDEvent);

post\_route.delete('/events/:uid', controller.deleteUIDEvent);

module.exports = post\_route;

**controllers/controller.js**

const Event = require('../models/events');

const createEvent = async (req, res) => {

    try {

        const event = new Event({

            name: req.body.name,

            tagline: req.body.tagline,

            file: 'http://localhost:8000/api/v3/app/' + req.file.filename,

            schedule: req.body.schedule,

            descrption: req.body.descrption,

            moderator: req.body.moderator,

            category: req.body.category,

            sub\_category: req.body.sub\_category,

            rigor\_rank: req.body.rigor\_rank

        });

        const eventData = await event.save();

        res.status(400).send({ success: true, msg: 'Event data', data: eventData });

    } catch (error) {

        res.status(400).send({ success: false, msg: error.message });

    }

}

const updateUIDEvent = async (req, res) => {

    try {

        const event = await Event.findByIdAndUpdate(req.params.uid, req.body);

        res.status(200).send({ success: true, msg: 'Post updated' });

    }

    catch (error) {

        res.status(400).send({ success: false, msg: error.message });

    }

}

const deleteUIDEvent = async (req, res) => {

    try {

        const event = await Event.deleteOne({ \_id: req.params.uid });

        res.status(200).send({ success: true, msg: 'Post deleted' });

    }

    catch (error) {

        res.status(400).send({ success: false, msg: error.message });

    }

}

const getEvent = async (req, res) => {

    try {

        if (req.query.page) {

            const page = parseInt(req.query.page); // Current page number, defaulting to 1

            const limit = parseInt(req.query.limit);

            const totalCount = await Event.countDocuments();

            const totalPages = Math.ceil(totalCount / limit);

            const event = await Event.find().sort({ createdAt: -1 }).skip((page - 1) \* limit).limit(limit);

            res.status(200).send({ success: true, msg: 'Data paginated retrive', data: event });

        }

        else if (req.query.id) {

            const uid = req.query.id;

            const event = await Event.find({ \_id: uid });

            res.status(200).send({ success: true, msg: 'Event data', data: event });

        }

    } catch (error) {

        res.status(400).send({ success: false, msg: error.message });

    }

}

module.exports = {

    createEvent,

    updateUIDEvent,

    deleteUIDEvent,

    getEvent

}

**models/events.js**

const mongoose = require('mongoose');

const eventSchema = new mongoose.Schema({

    name: {

        type: String,

        required: true,

    },

    file: {

        type: String,

        required: true,

        default:'IMG\_20230312\_203410.jpg'

    },

    tagline: {

        type: String,

        required: true

    },

    schedule: {

        type: Date,

        required: true,

        default: Date.now,

    },

    descrption: {

        type: String,

        required: true,

    },

    moderator: {

        type: String,

        rrequired: true,

    },

    category: {

        type: String,

        required: true,

    },

    sub\_category: {

        type: String,

        required: true,

    },

    rigor\_rank: {

        type: Number,

        required: true,

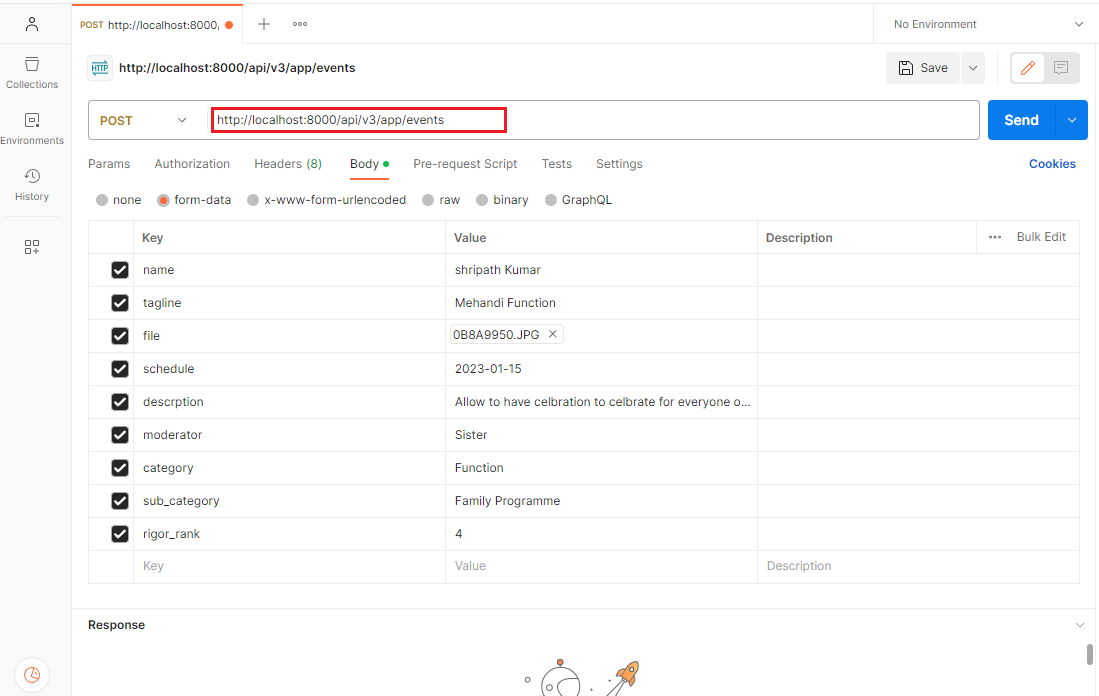
    }

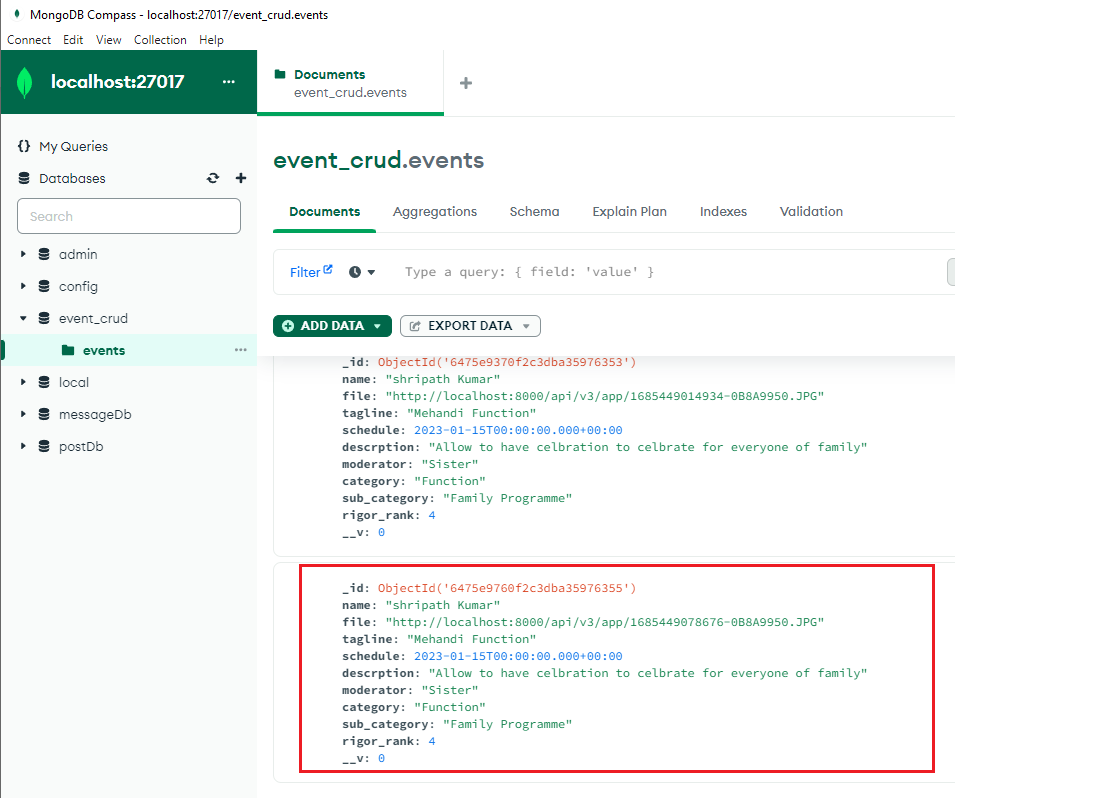
});

module.exports = mongoose.model('Events', eventSchema);

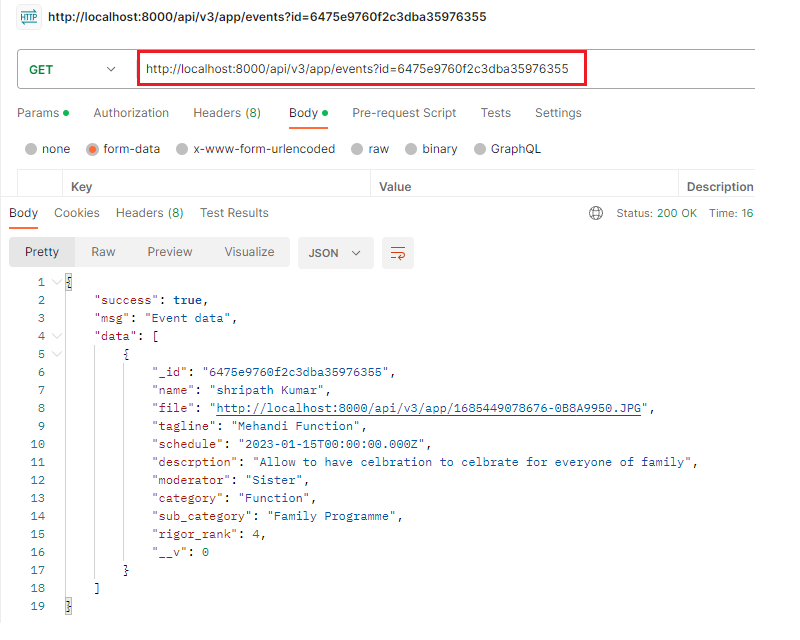
Output:-

Insert

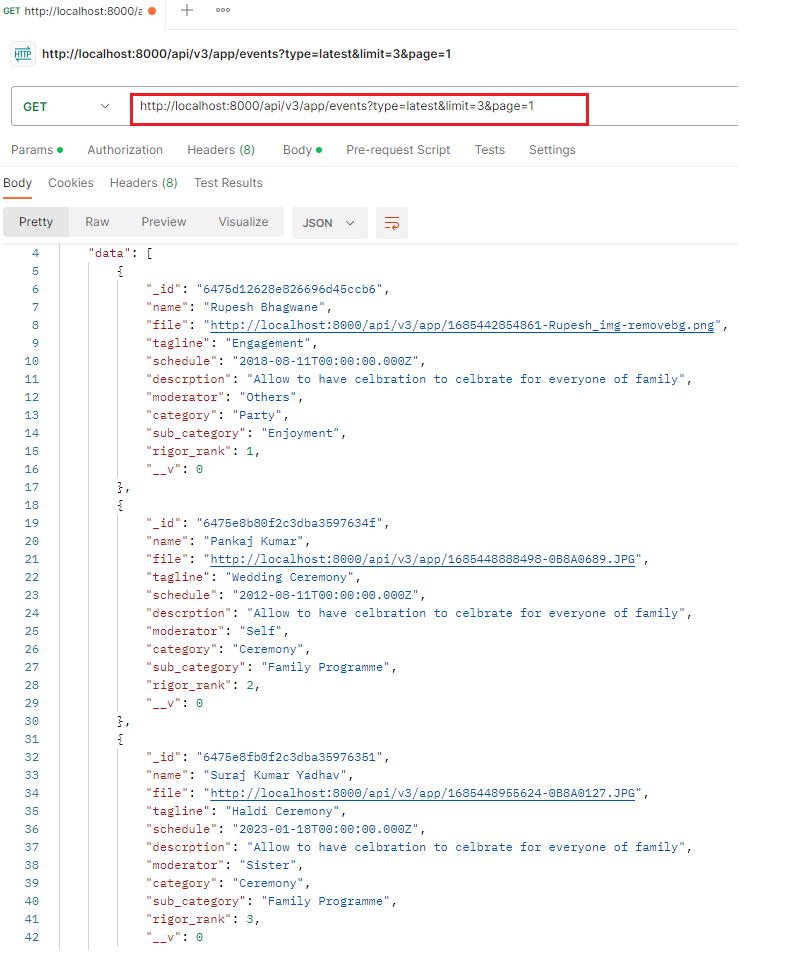




Get (According to event\_id)

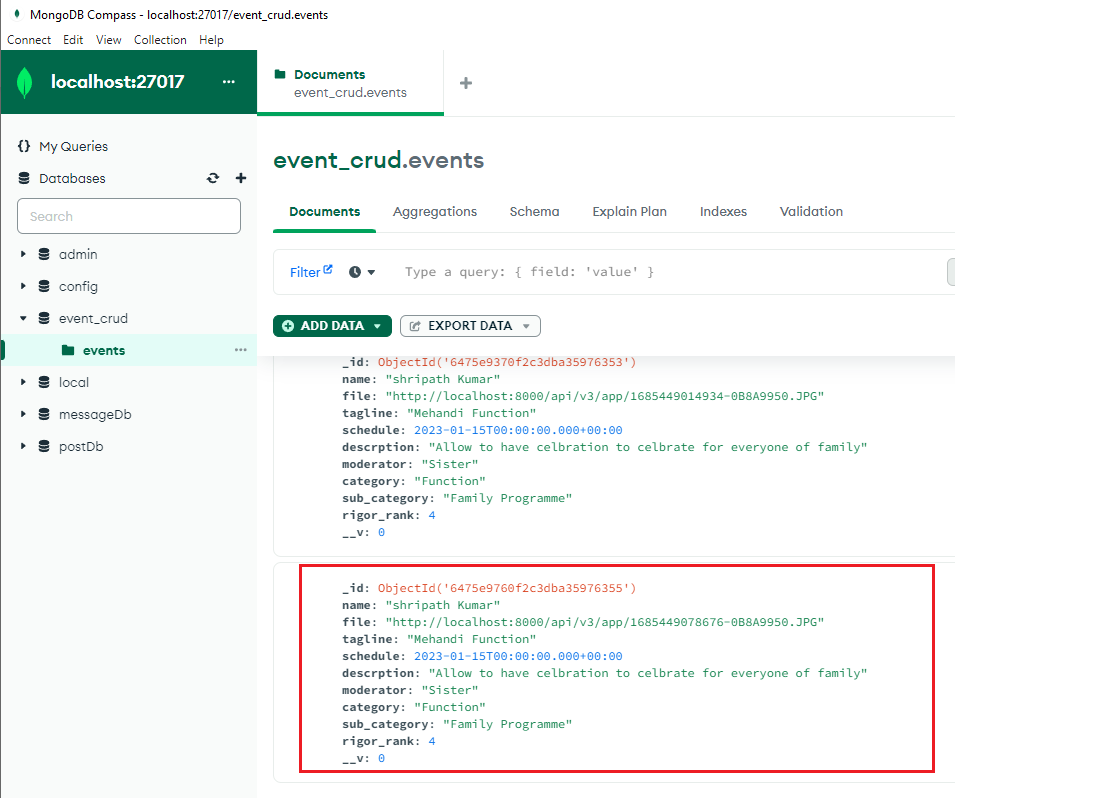


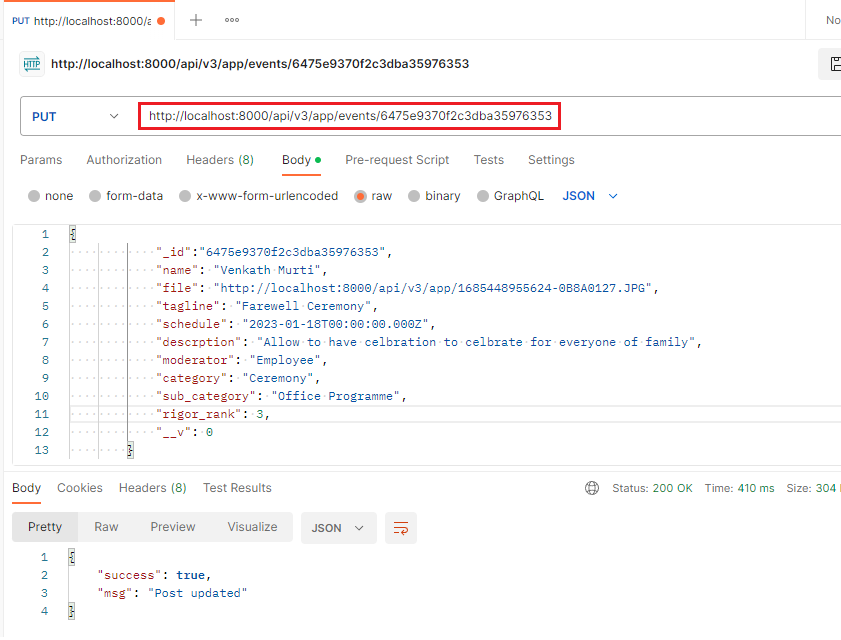
Get (paginate)



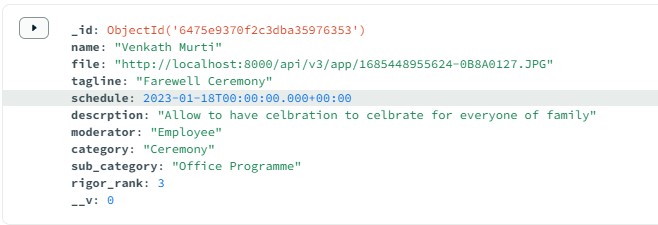
Update

Before Update





After Update



Delete

