**PROJECT NAME:**

Movies Api

**INTRODUCTION:**

REST API to perform movie details and actor details

**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
* Actors Schema

Movie schema

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const UserSchema = new Schema({

    id: {

        type: Number,

        required: true,

        index: {

          unique: true

        }

      },

    moviename: {

        type: String,

        required: true

    },

    movieGenre: {

        type: String,

        required: true

    },

    movieRating: {

        type: String,

        required: true

    },

    movieLanguage: {

        type: String,

        required: true

    },

    movieReleaseDate: {

        type: String,

        required: true

    },

    movieDirector: {

        type: String,

        required: true

    },

    movieDuration: {

        type: String,

        required: true,

    },

    actors: [{

        type : mongoose.Schema.ObjectId,

        ref : 'Actors'

      }]

});

module.exports = User = mongoose.model('project', UserSchema);

Actors schema

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const UserSchema = new Schema({

    id: {

        type: Number,

        required: true,

        index: {

          unique: true

        }

      },

    moviename: {

        type: String,

        required: true

    },

    movieGenre: {

        type: String,

        required: true

    },

    movieRating: {

        type: String,

        required: true

    },

    movieLanguage: {

        type: String,

        required: true

    },

    movieReleaseDate: {

        type: String,

        required: true

    },

    movieDirector: {

        type: String,

        required: true

    },

    movieDuration: {

        type: String,

        required: true,

    },

    actors: [{

        type : mongoose.Schema.ObjectId,

        ref : 'Actors'

      }]

});

module.exports = User = mongoose.model('project', UserSchema);

Now create routes,

1. **Add new movie**

const express = require('express');

const router = express.Router();

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

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

//to add movie

router.post('/', async(req, res) => {

    console.log("hello");

    const newUser = new User(req.body);

    try {

        await newUser.save();

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

    } catch (err) {

        res.status(500).send();

    }

});

1. **Get all movie details**
2. //to get all movies
3. router.get('/', async(req, res) => {
4. console.log("hi");
5. try {
6. const projects = await User.find({});
7. res.send(projects);
8. } catch (error) {
9. res.status(404).send({ error: 'Path not found' });
10. }
11. });

**3. Get movie details by id**

1. // to get movie details using ID
2. router.get('/:id', async(req, res) => {
3. const \_id = req.params.id;
4. try {
5. const user = await User.findById(\_id);
6. if (!user) {
7. return res.status(404).send({ error: 'movie not found' });
8. }
9. res.send(user);
10. } catch (error) {
11. res.status(500).send({ error: 'Internal server error' });
12. }
13. });

**4. To add a actors deails**

1. // to add actors
2. router.post('/Actors', async(req, res) => {
3. console.log("hell");
4. const newActors = new Actors(req.body);
5. try {
6. await newActors.save();
7. res.status(201).send(newActors);
8. } catch (err) {
9. res.status(500).send();
10. }
11. });

**5. update actor details using actor Id**

1. // to update actor details using id
2. router.patch('/:id', async(req, res) => {
3. console.log("hi0");
4. const updates = Object.keys(req.body);
5. const allowedUpdates = ['actorname', 'age'];
6. const isValidOperation = updates.every((update) => {
7. return allowedUpdates.includes(update);
8. });
9. if (!isValidOperation) {
10. return res.status(400).send({ error: 'Invalid Operation' });
11. }
12. try {
13. const user = await Actors.findById(req.params.id);
14. if (!user) {
15. return res.status(404).send({ error: 'User not found' });
16. }
17. updates.forEach((update) => {
18. user[update] = req.body[update];
19. });
20. await user.save();
21. res.send(user);
22. } catch (error) {
23. res.status(500).send({ error: 'Internal server error' });
24. }
25. });

**6. delete actor details using actor id**

//to delete actor details using actor id

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

    console.log("log");

    try {

        const user = await Actors.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' });

    }

});

Run the server

module.exports = User = mongoose.model('project', UserSchema);

**Run The Server**

const express = require('express');

const mongoose = require('mongoose');

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

const db = "mongodb+srv://saikumar2912:saikumar@cluster0.6llhp.mongodb.net/sai?retryWrites=true&w=majority"

const port = 6000;

const app = express();

app.use(express.json());

mongoose

    .connect(db, {

        useNewUrlParser: true,

        useUnifiedTopology: true,

        useCreateIndex: true

    })

    .then(() => {

        console.log('MongoDB Connnected');

    })

    .catch((err) => {

        console.log({ err: err });

    });

app.use('/users', users);

app.listen(port, () =>

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