**Assignment 7**

**Development and Testing of REST API with NodeJS, ExpresJS, and MongoDB**

Q. Create a NodeJS application to provide endpoints for a project called **Campus Trade.** The objective of the project is to trade used items (like book, calculator, Engineering Equipment, Sports equipment etc) with in the campus. There are two collections **Users** and **Listings**. Complete the below tasks to provide endpoints for the project.

Install required packages like – express, mongoose, nodemon and perform the following operations. Handle asynchronous behaviour and exceptions properly.

1. Define schema and model for **User,** with field name, mobile, email, sic number and password. All fields must be required and sic, mobile, and email must be unique.
2. Create a file to establish the connection with MongoDB and select the database.
3. Create a controller file for **User** and add functionalities for –
   1. Add a user.
   2. Retrieve all users.
   3. Retrieve a single user based on ID
   4. Retrieve a single user based on SIC/email/mobile
   5. Update a student based on SIC
   6. Delete a student based on SIC
4. Create a router file to connect with the User controller methods. Link the router to entry point of the project and test all the endpoints using Postman.

**Solution:**

**## server.js**

require("dotenv").config();

const express = require("express");

const dbConnection = require("./utils/db.js");

const userRouter = require("./routes/userRoutes.js");

const app = express();

app.use(express.json());

app.use('/user', userRouter);

dbConnection();

app.listen(process.env.PORT, () => {

console.log(`Server started at Port:${process.env.PORT}`);

})

**##/models/userSchema.js**

const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({

name: {

type: String,

required: true

},

email: {

type: String,

required: true,

unique: true

},

password: {

type: String,

required: true

},

mobile: {

type: Number,

required: true,

unique: true

},

sic: {

type: String,

required: true,

unique: true

}

});

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

**##/utils/db.js**

const mongoose = require("mongoose")

const dbConnection = async() => {

try {

await mongoose.connect(process.env.DB\_URL + "/" + process.env.DB\_NAME);

console.log("Database connected");

} catch (e) {

console.log(e);

}

}

module.exports = dbConnection;

**##/routes/userRoutes.js**

const express = require("express");

const { addUserController, getAllUserController, getUserByIdController, getUserByDataController, updateUserController, deleteUserController } = require("../controllers/userController");

const router = express.Router();

router.post("/add", addUserController);

router.get('/getAll', getAllUserController);

router.get('/get/:id', getUserByIdController);

router.get('/search/:info', getUserByDataController);

router.put('/update/:id', updateUserController);

router.delete('/delete/:id', deleteUserController);

module.exports = router;

**## controllers/userController.js**

const User = require("../models/userSchema")

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

try {

const user = await req.body;

console.log(user);

const newUser = await User.create(user);

if (!newUser) {

res.status(400).json({

success: false,

message: "User not created"

})

}

res.status(201).json({

success: true,

message: "user created",

data: newUser,

})

} catch (e) {

console.log(e);

res.status(500).json({

success: false,

message: "Internal sever error"

})

}

}

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

try {

const users = await User.find();

console.log(users);

if (users) {

res.status(200).json({

success: true,

message: 'Users fetched successfully',

data: users

});

} else {

res.status(404).json({

success: false,

message: "No users found"

})

}

} catch (e) {

console.log(e);

res.status(504).json({

success: false,

message: "Internal Server Error",

})

}

}

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

try {

const id = req.params.id;

console.log(id);

const user = await User.findById(id);

if (user) {

console.log(user);

res.status(200).json({

success: true,

message: "User found",

data: user

})

} else {

res.status(404).json({

success: false,

message: "User not found"

})

}

} catch (e) {

res.status(504).json({

success: false,

message: "Internal Server Error"

})

}

}

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

try {

const info = req.params.info;

const user = await User.findOne({

$or: [{

email: info

},

{

mobile: parseInt(info)

},

{

sic: info

}

]

});

if (user) {

console.log(user);

res.status(200).json({

success: true,

message: "User found",

data: user

})

} else {

res.status(404).json({

success: false,

message: "User not found"

})

}

} catch (e) {

console.log(e);

res.status(504).json({

success: true,

message: "Internal Server Error"

})

}

}

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

try {

const id = req.params.id;

const newData = req.body;

const updatedUser = await User.findByIdAndUpdate(id, newData);

if (updatedUser) {

console.log(updatedUser);

res.status(200).json({

success: true,

message: "User updated successfully",

data: updatedUser

})

} else {

res.status(400).json({

success: false,

message: "Update failed"

})

}

} catch (e) {

console.log(e);

res.status(504).json({

success: false,

message: "Internal Server Error"

})

}

}

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

try {

const id = req.params.id;

const deletedUser = await User.findByIdAndDelete(id);

if (deletedUser) {

res.status(200).json({

success: true,

message: "User Deleted successfully",

data: deletedUser

})

} else {

res.status(404).json({

success: false,

message: "User not found"

})

}

} catch (e) {

console.log(e);

res.status(500).json({

success: false,

message: "Internal Server Error"

})

}

}

module.exports = {addUserController, getAllUserController, getUserByIdController, getUserByDataController, updateUserController, deleteUserController };

**Presented By,**

Name- Rahul Kumar Singh

Class- CSE A2

Roll No- 16