**Module 4 Activity 1: MongoDB**

Saurabh Kale

IFT 458/554: Middleware Programming & Database Security

Dinesh Sthapit

Sep 29th, 2023

1. **Intro Full Stack**

**Anatomy of Modern Full Stack Development Architecture-**

**A diagram of a software development

Description automatically generated**

1. **Intro Full Stack Middleware-**

**Server.js-**

const express = require('express');

const app = express();

const port = 3000;

app.get('/', (req, res) => res.send('Hello World! this is my first node app'));

app.get('/student', (req, res) => res.send('Hello World! from student'))

app.get('/student/grade', (req, res) => res.send('Hello World! from student grades'))

app.listen(port, () => console.log(`App listening on port ${port}!`));

**Screenshot of code running –**

**A screen shot of a computer

Description automatically generated**

**Get request for root-**

**A screenshot of a computer

Description automatically generated**

**Get request for student route-**

**A screenshot of a computer

Description automatically generated**

**Get request for grades route-**

**A screenshot of a computer

Description automatically generated**

1. **Test Mongo DB Database-**

Inserted a record in the collection Users in database IFT-458-2023

A screenshot of a computer

Description automatically generated

1. **Connecting to MongoDB using Code-**

**Server.js-**

const express = require('express');

const app = express();

const port = 3000;

app.get('/', (req, res) => res.send('Hello World! this is my first node app'));

app.get('/student', (req, res) => res.send('Hello World! from student'))

app.get('/student/grade', (req, res) => res.send('Hello World! from student grades'))

app.listen(port, () => console.log(`App listening on port ${port}!`));

//Connecting to the database

const mongoose = require('mongoose');

//Synchronous connection

// mongoose.connect('mongodb://localhost:27017/StudentDB', {useNewUrlParse: true} , (err) => {

//     if(!err){console.log('MongoDB Connection Succeeded.')}

//     else{console.log('Error in DB connection : ' + err)}

// })

//Asynchronous connection

mongoose.connect('mongodb+srv://skale12:Bing\_2905@cluster0.hi0evjq.mongodb.net/demodb')

    .then(() => console.log('MongoDB connection Succesful'))

    .catch((err) => console.error(err))

**Screenshot of connection to MongoDB-**

A screenshot of a computer screen

Description automatically generated

1. **MVC Concept-**

A diagram of a software development

Description automatically generated

**Server.js-**

const express = require('express');

const app = express();

const port = 3000;

//routes

const studentRoute = require('./routes/studentRoute')

app.use('/student', studentRoute)

app.listen(port, () => console.log(`App listening on port ${port}!`));

//Connecting to the database

const mongoose = require('mongoose');

//Synchronous connection

// mongoose.connect('mongodb://localhost:27017/StudentDB', {useNewUrlParse: true} , (err) => {

//     if(!err){console.log('MongoDB Connection Succeeded.')}

//     else{console.log('Error in DB connection : ' + err)}

// })

//Asynchronous connection

mongoose.connect('mongodb+srv://skale12:Bing\_2905@cluster0.hi0evjq.mongodb.net/demodb')

    .then(() => console.log('MongoDB connection Succesful'))

    .catch((err) => console.error(err))

**studentRoute.js-**

const express = require('express');

const router = express.Router()

const studentController = require('../controllers/studentController')

router.get('/', studentController.getData);

router.post('/', studentController.postData);

router.put('/', studentController.updateData);

router.delete('/', studentController.deleteData);

module.exports = router;

**studentController.js-**

exports.getData = (req, res) => res.send('Hello World! from student GET');

exports.postData =  (req, res) => res.send('Hello World! from student POST');

exports.updateData = (req, res) => res.send('Hello World! from student PUT');

exports.deleteData =  (req, res) => res.send('Hello World! from student DELETE');

**Screenshot for student GET-**

A screenshot of a computer

Description automatically generated

**Screenshot for student POST-**

A screenshot of a computer

Description automatically generated

**Screenshot for student PUT-**

**A screenshot of a computer

Description automatically generated**

**Screenshot for student DELETE-**

**A screenshot of a computer

Description automatically generated**

1. **Refactoring and MVC Best Practices-**

**Server.js-**

const express = require('express');

const app = express();

const port = 3000;

//routes

const studentRoute = require('./routes/studentRoute')

const gradeRoute = require('./routes/gradeRoute')

app.use('/student', studentRoute)

app.use('/grade', gradeRoute)

app.listen(port, () => console.log(`App listening on port ${port}!`));

//Connecting to the database

const mongoose = require('mongoose');

//Synchronous connection

// mongoose.connect('mongodb://localhost:27017/StudentDB', {useNewUrlParse: true} , (err) => {

//     if(!err){console.log('MongoDB Connection Succeeded.')}

//     else{console.log('Error in DB connection : ' + err)}

// })

//Asynchronous connection

mongoose.connect('mongodb+srv://skale12:Bing\_2905@cluster0.hi0evjq.mongodb.net/demodb')

    .then(() => console.log('MongoDB connection Succesful'))

    .catch((err) => console.error(err))

**StudentRoute.js-**

const express = require('express');

const router = express.Router()

const studentController = require('../controllers/studentController')

router.get('/', studentController.getData);

router.post('/', studentController.postData);

router.put('/', studentController.updateData);

router.delete('/', studentController.deleteData);

module.exports = router;

**StudentController.js-**

exports.getData = (req, res) => {

    //get data from database

    const data = {

        name: 'john',

        age: 30,

        birth: '01/01/1993'

    }

    res.send(`Hello World! from student GET ${JSON.stringify(data)}`)

};

exports.postData =  (req, res) => res.send('Hello World! from student POST');

exports.updateData = (req, res) => res.send('Hello World! from student PUT');

exports.deleteData =  (req, res) => res.send('Hello World! from student DELETE');

**gradeRoute.js-**

const express = require('express');

const router = express.Router()

const gradeController = require('../controllers/gradeController')

router.get('/', gradeController.getData);

router.post('/', gradeController.postData);

router.put('/', gradeController.updateData);

router.delete('/', gradeController.deleteData);

module.exports = router;

**gradeController.js-**

exports.getData = (req, res) => {

    //get data from database

    const data = {

        subject: 'IFT 458',

        grade : "A",

        date: '01/01/1993'

    }

    res.send(`Hello World! from grade GET ${JSON.stringify(data)}`)

};

exports.postData =  (req, res) => res.send('Hello World! from grade POST');

exports.updateData = (req, res) => res.send('Hello World! from grade PUT');

exports.deleteData =  (req, res) => res.send('Hello World! from grade DELETE');

**Screenshot for GET grade-**

A screenshot of a computer

Description automatically generated

**Screenshot for POST grade-**

**A screenshot of a computer

Description automatically generated**

**Screenshot for PUT Grade-**

**A screenshot of a computer

Description automatically generated**

**Screenshot for DELETE Grade-**

**A screenshot of a computer

Description automatically generated**

1. **CRUD Operations Mongo DB-**

**Server.js-**

const express = require('express');

const bodyParser = require('body-parser');

const app = express();

const port = 3000;

app.use(bodyParser.json());

//routes

const studentRoute = require('./routes/studentRoute')

const gradeRoute = require('./routes/gradeRoute')

app.use('/student', studentRoute)

app.use('/grade', gradeRoute)

app.listen(port, () => console.log(`App listening on port ${port}!`));

//Connecting to the database

const mongoose = require('mongoose');

//Synchronous connection

// mongoose.connect('mongodb://localhost:27017/StudentDB', {useNewUrlParse: true} , (err) => {

//     if(!err){console.log('MongoDB Connection Succeeded.')}

//     else{console.log('Error in DB connection : ' + err)}

// })

//Asynchronous connection

mongoose.connect('mongodb+srv://skale12:Bing\_2905@cluster0.hi0evjq.mongodb.net/demodb')

    .then(() => console.log('MongoDB connection Succesful'))

    .catch((err) => console.error(err))

**StudentModel.js-**

const mongoose = require('mongoose');

const studentSchema = new mongoose.Schema({

    name: String,

    age: Number,

    birth: Date

})

const Student = mongoose.model('Student', studentSchema);

module.exports = Student;

**StudentRoute.js-**

const express = require('express');

const router = express.Router()

const studentController = require('../controllers/studentController')

router.get('/', studentController.getData);

router.post('/', studentController.postData);

router.put('/', studentController.updateData);

router.delete('/', studentController.deleteData);

module.exports = router;

**studentController.js-**

const Student = require('../models/studentModel')

exports.getData = async (req, res) => {

    //get data from database

    const students = await Student.find()

    res.status(200).json({

        status: 'success',

        results: students.length,

        data: {

            students

        }

    });

};

exports.postData = async (req, res) =>{

    const newStudent = req.body;

    const student = await Student.create(newStudent);

    console.log(`new Student - ${newStudent}`);

    res.status(201).json({

        status: 'success',

        data: student

    })

}

exports.updateData = (req, res) => res.send('Hello World! from student PUT');

exports.deleteData =  (req, res) => res.send('Hello World! from student DELETE');

**gradeRoute.js-**

const express = require('express');

const router = express.Router()

const gradeController = require('../controllers/gradeController')

router.get('/', gradeController.getData);

router.post('/', gradeController.postData);

router.put('/', gradeController.updateData);

router.delete('/', gradeController.deleteData);

module.exports = router;

**gradeController.js-**

exports.getData = (req, res) => {

    //get data from database

    const data = {

        subject: 'IFT 458',

        grade : "A",

        date: '01/01/1993'

    }

    res.send(`Hello World! from grade GET ${JSON.stringify(data)}`)

};

exports.postData =  (req, res) => res.send('Hello World! from grade POST');

exports.updateData = (req, res) => res.send('Hello World! from grade PUT');

exports.deleteData =  (req, res) => res.send('Hello World! from grade DELETE');

**POST Student request-**

A screenshot of a computer

Description automatically generated

**Screenshot of new student added in Database-**

A screenshot of a computer

Description automatically generated

**Get student in POSTMAN-**

A screenshot of a computer

Description automatically generated

**View Added-**

**Server.js-**

const express = require('express');

const bodyParser = require('body-parser');

const app = express();

const port = 3000;

app.use(bodyParser.json());

//routes

const studentRoute = require('./routes/studentRoute')

const gradeRoute = require('./routes/gradeRoute')

app.use(express.static('views'));

app.use('/student', studentRoute)

app.use('/grade', gradeRoute)

app.listen(port, () => console.log(`App listening on port ${port}!`));

//Connecting to the database

const mongoose = require('mongoose');

//Synchronous connection

// mongoose.connect('mongodb://localhost:27017/StudentDB', {useNewUrlParse: true} , (err) => {

//     if(!err){console.log('MongoDB Connection Succeeded.')}

//     else{console.log('Error in DB connection : ' + err)}

// })

//Asynchronous connection

mongoose.connect('mongodb+srv://skale12:Bing\_2905@cluster0.hi0evjq.mongodb.net/demodb')

    .then(() => console.log('MongoDB connection Succesful'))

    .catch((err) => console.error(err))

**Index.html-**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Grade and Student Management</title>

</head>

<body>

    <h1>Welcome to Grade and Student Management</h1>

    <h2>Grades</h2>

    <button onclick="getGrades()">Get All Grades</button>

    <button onclick="createGrade()">Create Grade</button>

    <h3>Create a New Grade</h3>

    <label for="subject">Subject:</label>

    <input type="text" id="subject" placeholder="Math">

    <label for="grade">Grade:</label>

    <input type="text" id="grade" placeholder="A">

    <label for="date">Date:</label>

    <input type="text" id="date" placeholder="01/01/1990">

    <button onclick="createGrade()">Submit</button>

    <h2>Students</h2>

    <button onclick="getStudents()">Get All Students</button>

    <button onclick="createStudent()">Create Student</button>

    <h3>Create a New Student</h3>

    <label for="name">Name:</label>

    <input type="text" id="name" placeholder="John Doe">

    <label for="age">Age:</label>

    <input type="text" id="age" placeholder="25">

    <label for="birth">Birth Date:</label>

    <input type="text" id="birth" placeholder="01/01/1990">

    <button onclick="createStudent()">Submit</button>

    <h3>Response:</h3>

    <div id="response"></div>

    <script>

        function getGrades() {

            fetch('/grade')

                .then(response => response.json())

                .then(data => {

                    displayResponse(data,'allGrades');

                });

        }

        function createGrade() {

            const subject = document.getElementById('subject').value;

            const grade = document.getElementById('grade').value;

            const date = document.getElementById('date').value;

            const newGrade = {

                subject,

                grade,

                date

            };

            fetch('/grade', {

                method: 'POST',

                headers: {

                    'Content-Type': 'application/json',

                },

                body: JSON.stringify(newGrade),

            })

            .then(response => response.json())

            .then(data => {

                displayResponse(data,'singleGrade');

                document.getElementById('subject').value = '';

                document.getElementById('grade').value = '';

                document.getElementById('date').value = '';

            });

        }

        function getStudents() {

            fetch('/student')

                .then(response => response.json())

                .then(data => {

                    displayResponse(data,'allStudents');

                });

        }

        function createStudent() {

            const name = document.getElementById('name').value;

            const age = document.getElementById('age').value;

            const birth = document.getElementById('birth').value;

            const newStudent = {

                name,

                age,

                birth

            };

            fetch('/student', {

                method: 'POST',

                headers: {

                    'Content-Type': 'application/json',

                },

                body: JSON.stringify(newStudent),

            })

            .then(response => response.json())

            .then(data => {

                displayResponse(data, 'singleStudent');

                document.getElementById('name').value='';

                document.getElementById('age').value='';

                document.getElementById('birth').value='';

            });

        }

        function displayResponse(data, type) {

            const responseDiv = document.getElementById('response');

            responseDiv.innerHTML = '';

            if (data.status === 'success') {

                var results= 1

                if(type=='allStudents'||type=='allGrades')

                    results = data.results;

                if (results > 0 ) {

                    const table = document.createElement('table');

                    var headers = 0

                    if(type=='allStudents')

                        headers = Object.keys(data.data['students'][0]);

                    else if(type=='allGrades')

                        headers = Object.keys(data.data['grades'][0]);

                    else

                        headers = Object.keys(data.data);

                    // Create table headers

                    const headerRow = table.insertRow(0);

                    headers.forEach(headerText => {

                        const header = document.createElement('th');

                        header.textContent = headerText;

                        headerRow.appendChild(header);

                    });

                    var mainData;

                    if(type=='allStudents')

                        mainData = data.data['students'];

                    else if(type=='allGrades')

                        mainData = data.data['grades'];

                    else

                        mainData = [data.data];

                    // Create table rows

                    mainData.forEach(item => {

                        const row = table.insertRow();

                        headers.forEach(header => {

                            const cell = row.insertCell();

                            cell.textContent = item[header];

                        });

                    });

                    responseDiv.appendChild(table);

                } else {

                    responseDiv.textContent = 'No data found.';

                }

            } else {

                responseDiv.textContent = 'Error occurred.';

            }

        }

    </script>

</body>

</html>

**Screenshot of create Student-**

**A screenshot of a computer

Description automatically generated**

**Screenshot of create Grade-**

**A screenshot of a computer

Description automatically generated**

**Screenshot of all students-**

**A screenshot of a computer

Description automatically generated**

**Screenshot of get all grades-**

**A screenshot of a computer

Description automatically generated**

**Deliverables-**

**Screenshot of email confirmation from mongodb atlas-**

**A screenshot of a computer

Description automatically generated**

**Screenshot of project setup in separate folder-**

**A computer screen shot of a program

Description automatically generated**

**Screenshot of cluster in MongoDB-**

**A screenshot of a computer

Description automatically generated**

**Screenshot of database and collections-**

**A screenshot of a computer

Description automatically generated**

**Screenshot IP-**

**A screenshot of a computer

Description automatically generated**

**Screenshot of User –**

**A screenshot of a computer

Description automatically generated**

**Screenshot of connection string-**

**A screenshot of a computer

Description automatically generated**