# **Module 4 Activity 1: MongoDB**

## IFT 544: Middleware Prog & Database Sec (2023 Fall)

Santrupth Sunkari

Prof. Dinesh Sthapit

October 1, 2023

1. **Introduction to Full Stack**

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

A computer screen shot of a diagram

Description automatically generated

1. **Introduction to 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 Running Code-**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. **Test Mongo DB Database-**

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! Its my first node application'));

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

app.get('/student/grade', (req, res) => res.send('Hello World! From Student Grades View'))

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://Testing:test1234@cluster0.vhexbih.mongodb.net/demodb')

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

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

A screenshot of a computer

Description automatically generated

1. **MVC Concept –**

**Anatomy of Modern Full Stack Development Architecture**

A computer screen shot of a computer screen

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://Testing:test1234@cluster0.vhexbih.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');

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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.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://Testing:test1234@cluster0.vhexbih.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: 'Ujjwal',

age: 23,

birth: '09/30/2023'

}

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 554',

grade : "A+",

date: '09/30/2023'

}

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');

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. **CRUD operations from 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://Testing:test1234@cluster0.vhexbih.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('../Model/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 554',

grade : "A+",

date: '09/30/2023'

}

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');

A screenshot of a computer

Description automatically generated

**Screenshot of New Student in Database-**

A computer screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**After Adding Views-**

**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://Testing:test1234@cluster0.vhexbih.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>Student and Grade Management System</title>

<style>

/\* Add your background color here \*/

body {

background-color: #F0F8FF; /\* Replace with your desired color code \*/

}

</style>

</head>

<body>

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

<h2>Grades</h2>

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

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

<h3>Creation of a New Grade</h3>

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

<input type="text" id="subject" placeholder="Data Visualization">

<label for="grade">Grade:</label>

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

<label for="date">Date:</label>

<input type="text" id="date" placeholder="09/30/2023">

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

<h2>Students</h2>

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

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

<h3>Creation of New Students</h3>

<label for="name">Name:</label>

<input type="text" id="name" placeholder="Name">

<label for="age">Age:</label>

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

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

<input type="text" id="birth" placeholder="DOB">

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

<h3>Responses:</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>

**Default Page:**

A screenshot of a computer

Description automatically generated

**Getting Students-**

A screenshot of a computer

Description automatically generated

**Getting all Students-**

A screenshot of a computer

Description automatically generated

**Deliverables-**

**Screenshot of the email confirmation from MongoDB Atlas-**

A computer screen shot of a chat window

Description automatically generated

**Screenshot of Project Setup**-

A computer screen shot of a program code

Description automatically generated

**Screenshot of the new cluster in MongoDB Atlas.**

A screenshot of a computer

Description automatically generated

**Screenshot of the database and collections-**

A computer screen shot of a computer

Description automatically generated

**Screenshot of the connection IP and user setup-**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Screenshot of the connection string**

A screenshot of a computer screen

Description automatically generated