## **Module 3 lab 1: Modules**

Saurabh Kale

IFT 458/554: Middleware Programming & Database Security

Dinesh Sthapit

Sep 24th, 2023

**App.js-**

// Name: Saurabh Kale

// ASUID: 1223450319

// Date: 9/24/2023

const express = require('express');

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

const { makeUpperCase, checkAdmin } = require('./middlewares');

const { getSpecificStudent, getStudents, addStudent, updateStudent, deleteStudent } = require('./controllers/students');

const app = express();

app.use(bodyParser.json());

// app.use(makeUpperCase);

app.get('/students/:id', checkAdmin, getSpecificStudent);

app.get('/students', getStudents);

app.post('/students', addStudent);

app.put('/students/:id', updateStudent);

app.delete('/students/:id', checkAdmin, deleteStudent);

app.use('/', function(req, res, next){

    console.log('Request Url:' + req.url);

    res.send('Hello');

});

app.listen(3000, () => {

    console.log('Server is running on port 3000')

});

**Students.js-**

// Name: Saurabh Kale

// ASUID: 1223450319

// Date: 9/24/2023

const students = [];

const getSpecificStudent = (req, res) => {

    const id = req.params.id;

    const currentStudent = students[id]

    if(currentStudent === undefined){

        res.send(`Student with the ${id} is not found`)

    }

    res.send(`Student with the ${id} is ${currentStudent.fname} ${currentStudent.mname} ${currentStudent.lname}`);

}

const getStudents = (req, res) => {

    res.send(students);

}

const addStudent = (req, res) => {

    const student = req.body;

    students.push(student);

    res.send(`Student with the name ${student.fname} ${student.mname} ${student.lname} added to the database`);

}

const updateStudent = (req, res) => {

    const id = req.params.id;

    const student = req.body;

    students[id] = student;

    res.send(`Student with the id ${id} has been updated`);

}

const deleteStudent = (req, res) => {

    const id = req.params.id;

    const student = students[id];

    students.splice(id, 1);

    res.send(`Student with the id ${id} has been deleted`);

}

module.exports = {

    getSpecificStudent,

    getStudents,

    addStudent,

    updateStudent,

    deleteStudent

}

**Index.js-**

// Name: Saurabh Kale

// ASUID: 1223450319

// Date: 9/24/2023

const makeUpperCase = (req, res, next) => {

    const student = req.body;

    if(req.body.fname == undefined || req.body.mname == undefined || req.body.lname == undefined){

        return res.send('Please enter a valid name');

    }

    req.body.fname = student.fname.toUpperCase();

    req.body.mname = student.mname.toUpperCase();

    req.body.lname = student.lname.toUpperCase();

    next();

}

const checkAdmin = (req, res, next) => {

    if(req.query.admin === 'true'){

        next();

    } else{

        res.status(400).send('Should be admin');

    }

}

module.exports = {

    makeUpperCase,

    checkAdmin

}

**getStudents-**

A screenshot of a computer

Description automatically generated

**Add student-**

A screenshot of a computer

Description automatically generated

**Get specific id should be admin–**

A screenshot of a computer

Description automatically generated

**Get specific id with admin true-**

A screenshot of a computer

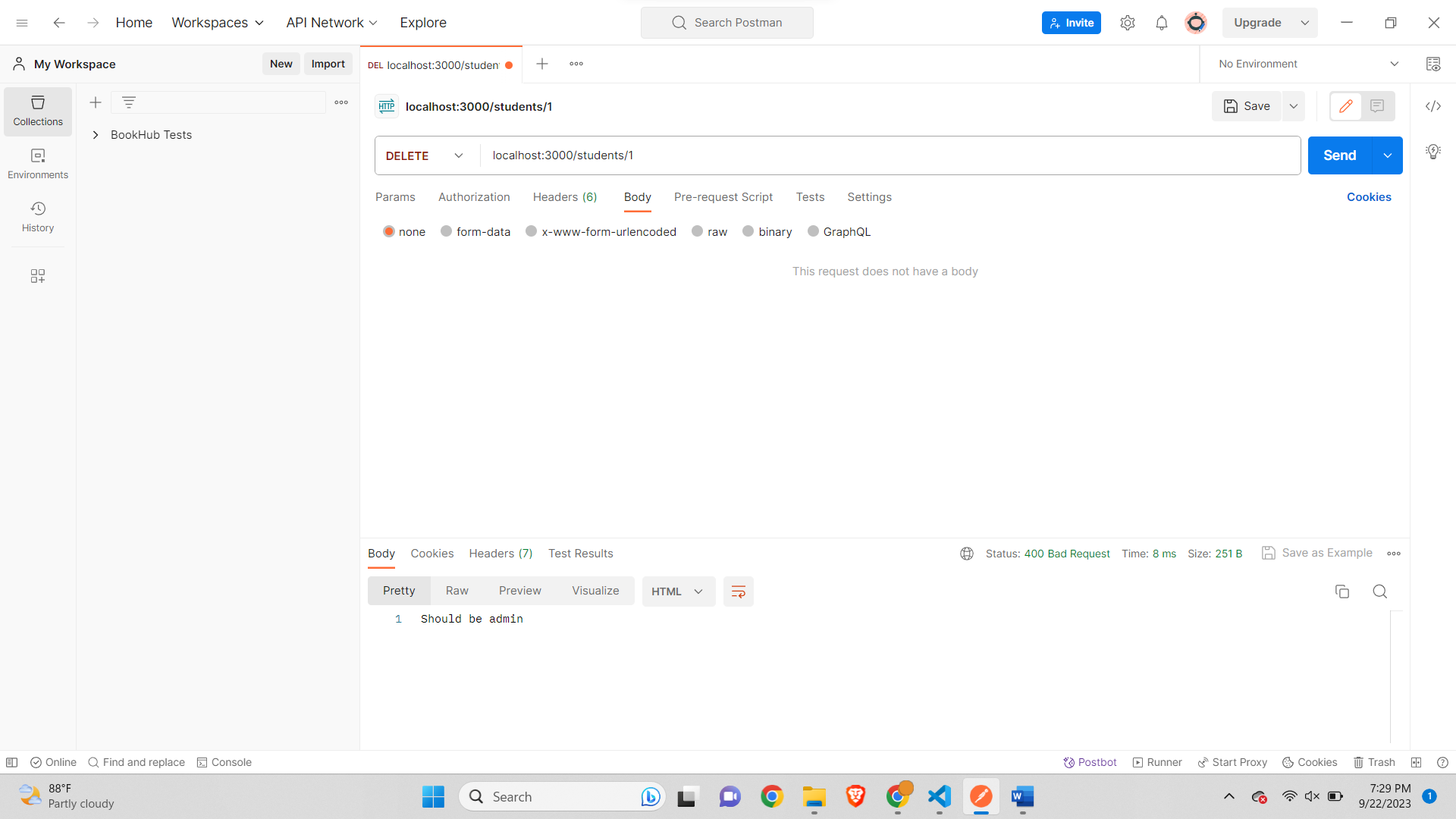
Description automatically generated

**Put-**

A screenshot of a computer

Description automatically generated

**Delete specifc with no admin privileges-**



**Delete Specific with Admin privileges-**

A screenshot of a computer

Description automatically generated

**Get all –**

A screenshot of a computer

Description automatically generated

**Assignment: User Management System-**

**userController.js-**

// Name: Saurabh Kale

// ASUID: 1223450319

// Date: 9/24/2023

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

const registerUser =  (req, res) => {

    const { username, email, password, verifyPassword } = req.body;

    // Check for duplicate usernames and emails

    const existingUser = global.users.find(user => user.username === username || user.email === email);

    if (existingUser) {

        // User or email already exists, show an error message

        res.render('register', { errorMessage: 'Username or email already exists' });

    }

    else if(verifyPassword != password) {

        // password and verify password not matching show an error

        res.render('register', { errorMessage: 'Password and verify password not matching type again' });

    }

    else {

        // Create a new user and add it to the users array (for simplicity)

        const newUser = new User(username, email, password);

        global.users.push(newUser);

        console.log(`users- ${JSON.stringify(global.users)}`)

        // Redirect to login page after successful registration

        res.redirect('/login?success=User%20Created%20successfully!');

    }

};

const login = (req, res) => {

    const successMessage = req.query.success;

    res.render('login', { successMessage });

}

const home =  (req, res) => {

    const successMessage = req.query.success;

    res.render('home', { successMessage });

}

const postLogin = (req, res) => {

    const { username, password } = req.body;

    // Find the user by username

    const user = global.users.find(user => user.username === username);

    global.currentLoggedInUser = user;

    if (!user || user.password !== password) {

        // User not found or password is incorrect, show an error message

        res.render('login', { errorMessage: 'Invalid username or password' });

    } else {

        // Redirect to a user profile or dashboard page after successful login

        res.render('home'); //Succesfull Login

    }

}

const updateInfo = (req, res) => {

    const successMessage = req.query.success;

    res.render('updateInfo', { successMessage });

  }

const getUpdatePassword = (req, res) => {

    const successMessage = req.query.success;

    res.render('updatePassword', { successMessage });

  }

const postUpdateInfo = (req, res) => {

    const { username, email  } = req.body;

        const user = global.users.find(user => user.username === global.currentLoggedInUser.username);

        user.username = username;

        user.email = email

        console.log(`user updated and updated list- ${JSON.stringify(global.users)}`)

        res.redirect('/home?success=User%20Details%20Updated%20successfully!');

    }

const postUpdatePassword =  (req, res) => {

    const { password, verifyPassword } = req.body;

    if(verifyPassword != password) {

        // password and verify password not matching show an error

        res.render('updatePassword', { errorMessage: 'Password and verify password not matching type again' });

    }

    else{

        // Find the user by username

        const user = global.users.find(user => user.username === global.currentLoggedInUser.username);

        user.password = password;

        console.log(`user updated and updated list- ${JSON.stringify(global.users)}`)

        res.redirect('/home?success=User%20Details%20Updated%20successfully!');

    }

}

const getLogout = (req, res) => {

    global.currentLoggedInUser = {}

  res.redirect('/login?success=User%20Logged%20Out%20successfully!');

}

module.exports = {

    registerUser,

    login,

    home,

    postLogin,

    updateInfo,

    getUpdatePassword,

    postUpdateInfo,

    postUpdatePassword,

    getLogout

}

**userModel.js-**

// Name: Saurabh Kale

// ASUID: 1223450319

// Date: 9/24/2023

class User {

    constructor(username, email, password) {

        this.username = username;

        this.email = email;

        this.password = password;

    }

}

module.exports = User;

**home.ejs-**

<!-- Name: Saurabh

ASU ID: 1223450319

Date: 9/24/2023 -->

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Home Page</title>

</head>

<body>

    <header>

        <h1>Welcome to User Management System Website</h1>

        <nav>

            <ul>

                <li><a href="/">Home</a></li>

                <li><a href="/updateInfo">Change Info</a></li>

                <li><a href="/updatePassword">Change Password</a></li>

                <li><a href="/logout">Logout</a></li>

            </ul>

        </nav>

    </header>

    <% if (typeof errorMessage !== 'undefined') { %>

        <p style="color:red"><%= errorMessage %></p>

    <% } %>

    <% if (typeof successMessage !== 'undefined') { %>

        <p style="color:green"><%= successMessage %></p>

    <% } %>

    <section>

        <h2>About Us</h2>

        <p>Welcome to our User Management System, where simplicity meets efficiency.

            At the heart of our platform is a commitment to streamlining user administration, ensuring seamless access control, and enhancing security.

            Join us on this journey of empowering businesses with user-centric solutions, and experience the ease of managing your digital identity.</p>

    </section>

    <section>

        <h2>Our Services</h2>

        <ul>

            <li>Service 1</li>

            <li>Service 2</li>

            <li>Service 3</li>

        </ul>

    </section>

    <section>

        <h2>Contact Us</h2>

        <p>Email: skale12@asu.edu</p>

        <p>Phone: +1 (123) 456-7890</p>

    </section>

    <footer>

        <p>&copy; 2023 User Management System</p>

    </footer>

</body>

</html>

**Login.ejs-**

<!-- Name: Saurabh

ASU ID: 1223450319

Date: 9/24/2023 -->

<!DOCTYPE html>

<html>

<head>

    <title>Login Page</title>

</head>

<body>

    <h1>Login</h1>

    <form action="/login" method="POST">

        <label for="username">Username:</label>

        <input type="text" id="username" name="username" required>

        <label for="password">Password:</label>

        <input type="password" id="password" name="password" required>

        <input type="submit" value="Login">

    </form>

    <p>Don't have an account? <a href="/register">Register here</a></p>

    <% if (typeof errorMessage !== 'undefined') { %>

        <p style="color:red"><%= errorMessage %></p>

    <% } %>

    <% if (typeof successMessage !== 'undefined') { %>

        <p style="color:green"><%= successMessage %></p>

    <% } %>

</body>

</html>

**Register.ejs-**

<!-- Name: Saurabh

ASU ID: 1223450319

Date: 9/24/2023 -->

<!DOCTYPE html>

<html>

<head>

    <title>Registration Page</title>

</head>

<body>

    <h1>Registration</h1>

    <form action="/register" method="POST">

        <label for="username">Username:</label>

        <input type="text" id="username" name="username" required><br><br>

        <label for="email">Email:</label>

        <input type="email" id="email" name="email" required><br><br>

        <label for="password">Password:</label>

        <input type="password" id="password" name="password" required><br><br>

        <label for="verifyPassword">Verify Password:</label>

        <input type="password" id="verifyPassword" name="verifyPassword" required><br><br>

        <input type="submit" value="Register">

        <p>Already have an account? <a href="/login">Login here</a></p>

        <% if (typeof errorMessage !== 'undefined') { %>

            <p style="color:red"><%= errorMessage %></p>

        <% } %>

    </form>

</body>

</html>

**updateInfo.ejs-**

<!-- Name: Saurabh

ASU ID: 1223450319

Date: 9/24/2023 -->

<!DOCTYPE html>

<html>

<head>

    <title>Update Info Page</title>

</head>

<body>

    <h1>Update Information</h1>

    <form action="/updateInfo" method="POST">

        <label for="username">Username:</label>

        <input type="text" id="username" name="username" required><br><br>

        <label for="email">Email:</label>

        <input type="email" id="email" name="email" required><br><br>

        <input type="submit" value="Submit"><br>

        <a href="/home">cancel</a>

        <% if (typeof errorMessage !== 'undefined') { %>

            <p style="color:red"><%= errorMessage %></p>

        <% } %>

    </form>

</body>

</html>

**updatePassword.ejs-**

<!-- Name: Saurabh

ASU ID: 1223450319

Date: 9/24/2023 -->

<!DOCTYPE html>

<html>

<head>

    <title>Update Password Page</title>

</head>

<body>

    <h1>Password change</h1>

    <form action="/updatePassword" method="POST">

        <label for="password">Password:</label>

        <input type="password" id="password" name="password" required>

        <label for="verifyPassword">Verify Password:</label>

        <input type="password" id="verifyPassword" name="verifyPassword" required><br><br>

        <input type="submit" value="Submit">

    </form><br>

    <a href="/home">cancel</a>

    <% if (typeof errorMessage !== 'undefined') { %>

        <p style="color:red"><%= errorMessage %></p>

    <% } %>

    <% if (typeof successMessage !== 'undefined') { %>

        <p style="color:green"><%= successMessage %></p>

    <% } %>

</body>

</html>

**app.js-**

const express = require('express');

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

const User = require('./model/userModel'); // Import the User model

const app = express();

const port = 3000; // You can change the port as needed

// Set up EJS as the view engine

app.set('view engine', 'ejs');

app.set('views', \_\_dirname + '/views');

const { registerUser, login, home, postLogin, updateInfo, getUpdatePassword, postUpdateInfo, postUpdatePassword, getLogout } = require('./controllers/userController');

// Middleware to parse JSON bodies

app.use(bodyParser.urlencoded({ extended: false }));

// In-memory user storage (for simplicity)

global.users = [];

global.currentLoggedInUser = {}

// Routes

// Home page (login page)

app.get('/', (req, res) => {

    res.render('login');

});

// Registration page

app.get('/register', (req, res) => {

    res.render('register');

});

// Handle registration form submission

app.post('/register',registerUser);

// Login page

app.get('/login', login);

app.get('/home', home);

// Handle login form submission

app.post('/login', postLogin );

app.get('/updateInfo', updateInfo);

app.get('/updatePassword', getUpdatePassword);

// Handle login form submission

app.post('/updateInfo', postUpdateInfo);

// Handle login form submission

app.post('/updatePassword',postUpdatePassword);

app.get('/logout', getLogout);

// Start the server

app.listen(port, () => {

    console.log(`Server is running on port ${port}`);

});

**Login page-**

A screenshot of a computer

Description automatically generated

**Registration page-A screenshot of a computer

Description automatically generated**

**After registration –**

A screenshot of a computer

Description automatically generated

**After login-**

A screenshot of a computer

Description automatically generated

**Update Info page-**

A screenshot of a computer

Description automatically generated

**After updating information-**

A screenshot of a computer

Description automatically generated

**Update password page-**

A screenshot of a computer

Description automatically generated

**After updating password-**

A screenshot of a computer

Description automatically generated

**User logged out-**

A screenshot of a computer

Description automatically generated

**Part 3-**

**App.js-**

// Name: Saurabh Kale

// ASUID: 1223450319

// Date: 9/24/2023

const express = require('express');

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

const { makeUpperCase } = require('./middlewares');

const studentsRouter = require('./controllers/students')

const app = express();

app.use(bodyParser.json());// using the middleware body-parser to parse

app.use(makeUpperCase);// global middleware

app.use('/students', studentsRouter)// use studentsRouter

app.use('/', function(req, res, next){

    console.log('Request Url:' + req.url);

    res.send('Hello');

});

app.listen(3000, () => {

    console.log('Server is running on port 3000')

});

**Index.js-**

// Name: Saurabh Kale

// ASUID: 1223450319

// Date: 9/24/2023

const makeUpperCase = (req, res, next) => {

    const student = req.body;

    if(req.body.fname == undefined || req.body.mname == undefined || req.body.lname == undefined){

        return res.send('Please enter a valid name');

    }

    req.body.fname = student.fname.toUpperCase();

    req.body.mname = student.mname.toUpperCase();

    req.body.lname = student.lname.toUpperCase();

    next();

}

const checkAdmin = (req, res, next) => {

    if(req.query.admin === 'true'){

        next();

    } else{

        res.status(400).send('Should be admin');

    }

}

module.exports = {

    makeUpperCase,

    checkAdmin

}

**Students.js-**

// Name: Saurabh Kale

// ASUID: 1223450319

// Date: 9/24/2023

const express = require('express');

const router = express.Router();

const { makeUpperCase, checkAdmin } = require('../middlewares/index');

const students = [];

const getSpecificStudent = (req, res) => {

    const id = req.params.id;

    const currentStudent = students[id]

    if(currentStudent === undefined){

        res.send(`Student with the ${id} is not found`)

    }

    res.send(`Student with the ${id} is ${currentStudent.fname} ${currentStudent.mname} ${currentStudent.lname}`);

}

const getStudents = (req, res) => {

    res.send(students);

}

const addStudent = (req, res) => {

    const student = req.body;

    students.push(student);

    res.send(`Student with the name ${student.fname} ${student.mname} ${student.lname} added to the database`);

}

const updateStudent = (req, res) => {

    const id = req.params.id;

    const student = req.body;

    students[id] = student;

    res.send(`Student with the id ${id} has been updated`);

}

const deleteStudent = (req, res) => {

    const id = req.params.id;

    const student = students[id];

    students.splice(id, 1);

    res.send(`Student with the id ${id} has been deleted`);

}

router.get('/:id', checkAdmin, getSpecificStudent);

router.get('/', getStudents);

router.post('/', addStudent);

router.put('/:id', updateStudent);

router.delete('/:id', checkAdmin, deleteStudent);

module.exports = router ;

**Screenshots-**

**Get students-**

**A screenshot of a computer

Description automatically generated**

**Get specific student without admin privileges-**

**A screenshot of a computer

Description automatically generated**

**Post student –**

**A screenshot of a computer

Description automatically generated**

**Get specific student with admin privileges-**

**A screenshot of a computer

Description automatically generated**

**Update specifc student-**

**A screenshot of a computer

Description automatically generated**

**Delete specific student without admin privileges-**

**A screenshot of a computer

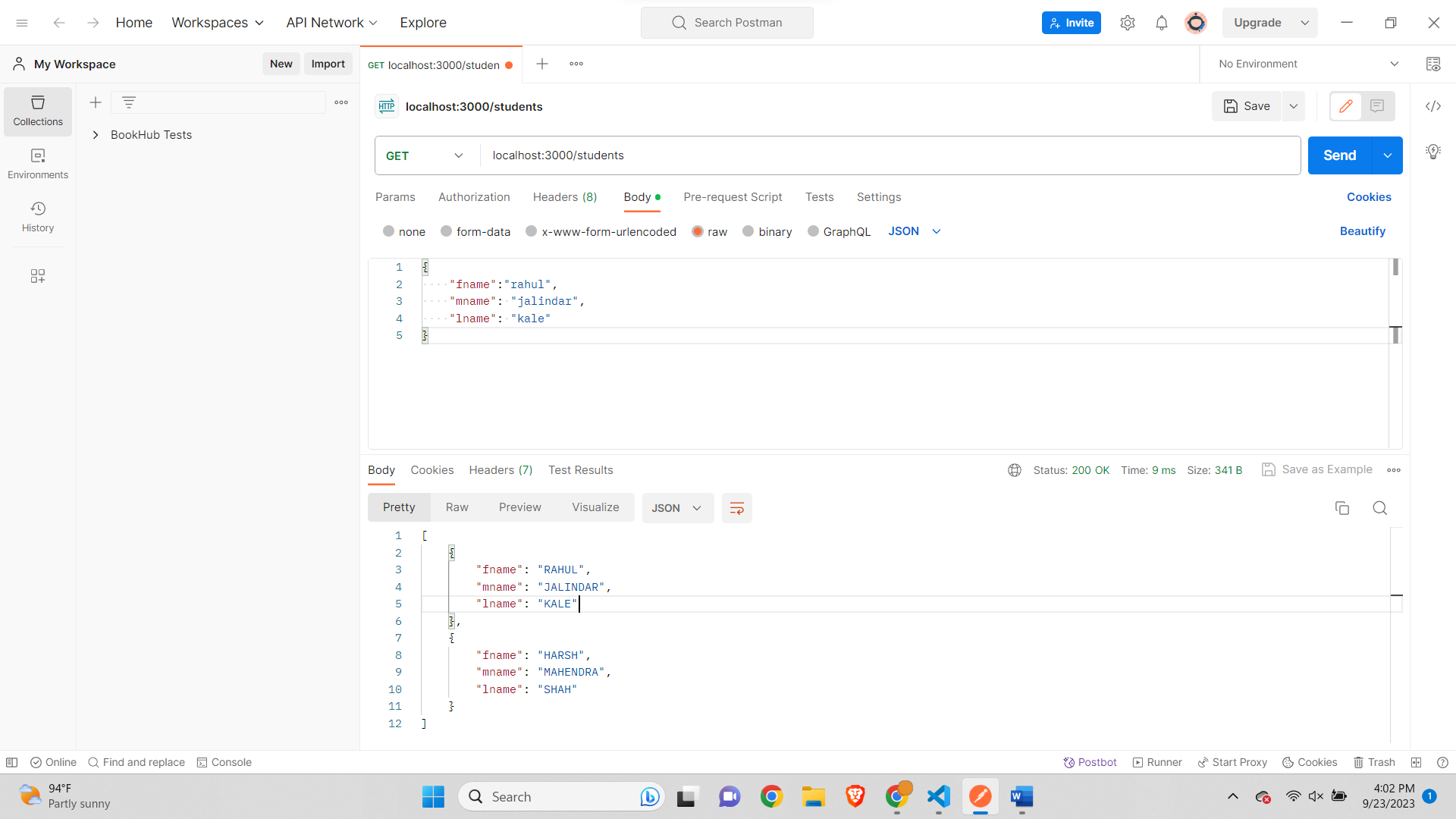
Description automatically generated**

**Delete student with admin privileges-**

**A screenshot of a computer

Description automatically generated**

**Get all students-**

****