PINEDA, Eldrin Josh P. WD - 302 09/16/25

FINALS ACTIVITY 1: Node, Express, Mongo, & Vue

Codes

Home.vue

```vue
<template>
  <div>
    <header>
      <a href="/" class="logo-link">
        <img src="/HauLogo.png" alt="HAU-Logo" />
        <span>Holy Angel University</span>
      </a>
    </header>
    <nav>
      <ul>
        <li><RouterLink to="/student">Student Sign-In</RouterLink></li>
        <li><RouterLink to="/admin">Admin Sign-In</RouterLink></li>
      </ul>
    </nav>
  </div>
</template>
```

Student.vue

```vue
<script setup>
import { ref } from 'vue'
import axios from 'axios'

const studentID = ref('')
const firstName = ref('')
const lastName = ref('')
const section = ref('')
const response = ref(null)

async function submitForm() {
  try {
    const res = await axios.post('http://localhost:5000/addStudent', {
      studentID: studentID.value,
      firstName: firstName.value,
      lastName: lastName.value,
      section: section.value
    })
    response.value = res.data
  } catch (err) {
    console.error("Error saving student:", err)
  }
}
</script>
```

```vue
<template>
  <div>
    <header>
      <RouterLink to="/" class="logo-link">
        <img src="/HauLogo.png" alt="HAU-Logo" />
        <span> Holy Angel University </span>
      </RouterLink>
    </header>

    <form @submit.prevent="submitForm">
      <label for="studentID">StudentID</label>
      <input v-model="studentID" type="text" id="studentID" />
      <br />

      <label for="firstName">First Name</label>
      <input v-model="firstName" type="text" id="firstName" />
      <br />

      <label for="lastName">Last Name</label>
      <input v-model="lastName" type="text" id="lastName" />
      <br />

      <label for="section">Section</label>
      <input v-model="section" type="text" id="section" />
      <br />

      <button type="submit">Submit</button>
    </form>

  </div>
</template>
```

Admin.vue

```vue
<script setup>
import { ref } from 'vue'
import axios from 'axios'

const adminID = ref('')
const firstName = ref('')
const lastName = ref('')
const department = ref('')

async function submitForm() {
  try {
    const res = await axios.post('http://localhost:5000/addAdmin', {
      adminID: adminID.value,
      firstName: firstName.value,
      lastName: lastName.value,
      department: department.value
    })
    console.log("Saved to DB:", res.data) // This shows up in your VS Code
    terminal
  } catch (err) {
    console.error("Error saving admin:", err)
  }
}
</script>
```

```vue
<template>
  <div>
    <header>
      <RouterLink to="/" class="logo-link">
        <img src="/HauLogo.png" alt="HAU-Logo" />
        <span> Holy Angel University </span>
      </RouterLink>
    </header>

    <form @submit.prevent="submitForm">
      <label for="adminID">AdminID</label>
      <input v-model="adminID" type="text" id="adminID" />
      <br />

      <label for="firstName">First Name</label>
      <input v-model="firstName" type="text" id="firstName" />
      <br />

      <label for="lastName">Last Name</label>
      <input v-model="lastName" type="text" id="lastName" />
      <br />

      <label for="department">Department</label>
      <input v-model="department" type="text" id="department" />
      <br />

      <button type="submit">Submit</button>
    </form>
  </div>
</template>
```

express.js (Server)

```
// PINEDA, Eldrin Josh P.
// WD-302

import mongoose from 'mongoose';
import express from 'express';
import cors from 'cors';

mongoose.connect("mongodb://127.0.0.1:27017/schoolDB", {
    useNewUrlParser: true,
    useUnifiedTopology: true,
})
.then(() => console.log("MongoDB connected"))
.catch(err => console.error("MongoDB connection error:", err));

const studentSchema = new mongoose.Schema({
    studentID: { type: String, required: true },
    firstName: String,
    lastName: String,
    section: String,
});

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

const adminSchema = new mongoose.Schema({
    adminID: { type: String, required: true },
    firstName: String,
    lastName: String,
    department: String,
});

const Admin = mongoose.model("Admin", adminSchema);

const app = express();
app.use(cors());
app.use(express.json());

// Save student
app.post('/addStudent', async (req, res) => {
    try {
        const newStudent = new Student(req.body);
        await newStudent.save();
        console.log("Student saved:", newStudent);
        res.json({ message: "Student saved", student: newStudent });
    } catch (err) {
        console.error("Error saving student:", err);
        res.status(500).json({ error: err.message });
    }
});
```

```
// Fetch all students
app.get('/getStudents', async (req, res) => {
    try {
        const students = await Student.find();
        console.log("All students in DB:", students);
        res.json(students);
    } catch (err) {
        console.error("Error fetching students:", err);
        res.status(500).json({ error: err.message });
    }
});

// --- Delete student by ID ---
app.delete('/deleteStudent/:id', async (req, res) => {
  try {
    const deletedStudent = await Student.findByIdAndDelete(req.params.id);
    if (!deletedStudent) return res.status(404).json({ message: "Student not
    found" });
    res.json({ message: "🗑️ Student deleted", student: deletedStudent });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// Save admin
app.post('/addAdmin', async (req, res) => {
    try {
        const newAdmin = new Admin(req.body);
        await newAdmin.save();
        console.log("Admin saved:", newAdmin);
        res.json({ message: "Admin saved", admin: newAdmin });
    } catch (err) {
        console.error("Error saving admin:", err);
        res.status(500).json({ error: err.message });
    }
});

// Fetch all admins
app.get('/getAdmins', async (req, res) => {
    try {
        const admins = await Admin.find();
        console.log("All admins in DB:", admins);
        res.json(admins);
    } catch (err) {
        console.error("Error fetching admins:", err);
        res.status(500).json({ error: err.message });
    }
```

```
// --- Delete student by ID ---
app.delete('/deleteAdmin/:id', async (req, res) => {
  try {
    const deletedAdmin = await Admin.findByIdAndDelete(req.params.id);
    if (!deletedAdmin) return res.status(404).json({ message: "Student not
    found" });
    res.json({ message: "Admin deleted", admin: deletedAdmin });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});



const PORT = 5000;
app.listen(PORT, () => {
    console.log(`Server running at http://localhost:${PORT}`);
});
```

index.js (Router)

```
import { createRouter, createWebHistory } from 'vue-router'
import Home from '../pages/home.vue'
import StudentSignIn from '../pages/student.vue'
import AdminSignIn from '../pages/admin.vue'

const routes = [
  { path: '/', component: Home },
  { path: '/student', component: StudentSignIn },
  { path: '/admin', component: AdminSignIn },
]

const router = createRouter({
  history: createWebHistory(),
  routes,
})

export default router
```

main.js

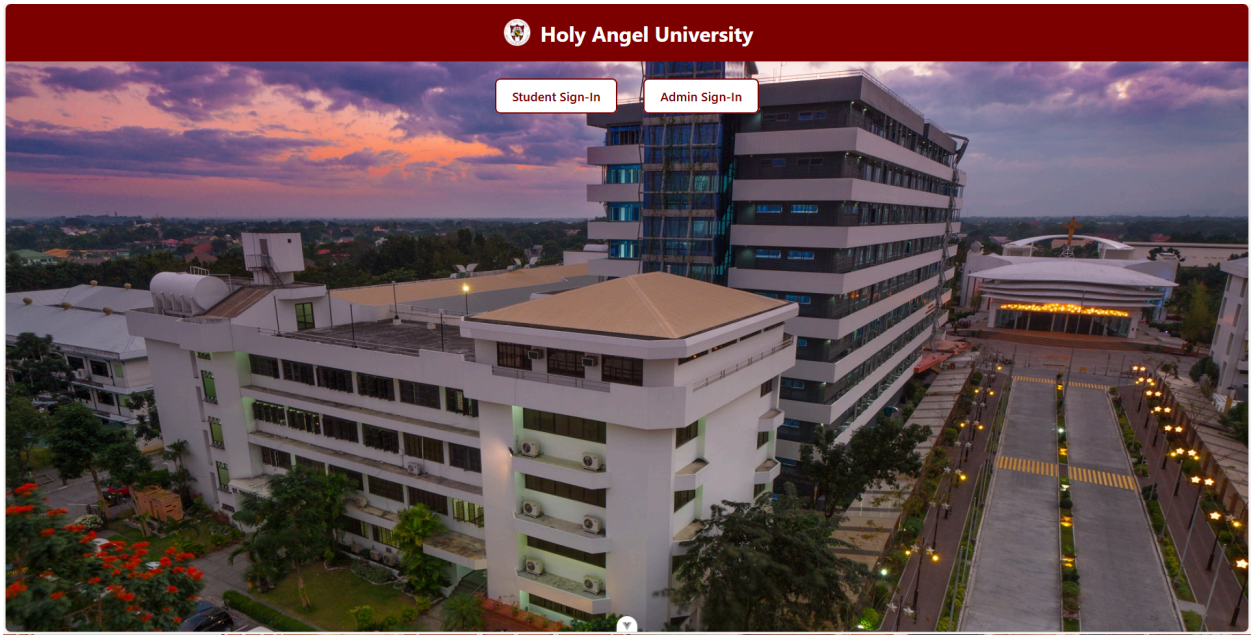
```
import { createApp } from 'vue'
import './style.css'
import App from './App.vue'
import router from './router'
import './assets/styles.css'

createApp(App).use(router).mount('#app')
```

styles.css

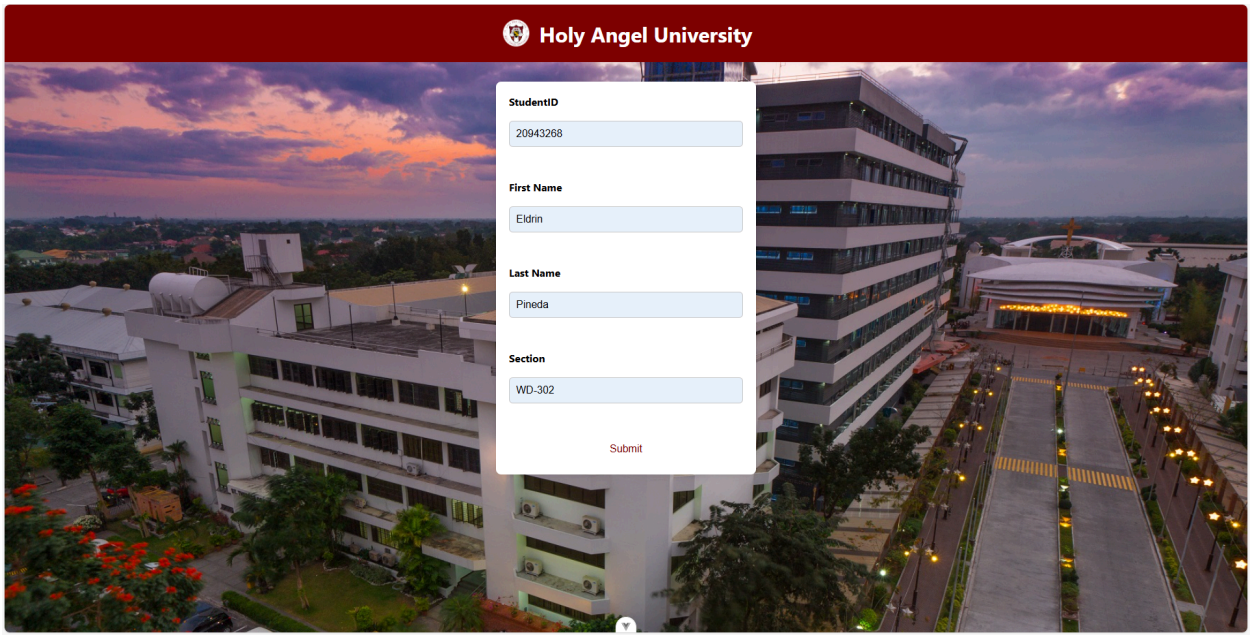
```css
/*
    PINEDA, Eldrin Josh P.
    WD-302
*/

/* HTML */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-image: url('/haubg.jpg');
    background-size: cover;
    background-repeat: no-repeat;
    height: 100vh;

}

/* Header */
header {
    background-color: maroon;
    color: white;
    padding: 20px;
    font-size: 2rem;
    font-weight: bold;
    display: flex;
    align-items: center;
    justify-content: center;
    width: 100%;
    position: relative;
    top: 0;
    left: 0;
}

header img {
    height: 3rem;
    width: auto;
    vertical-align: middle;
}

header a {
    display: flex;
    align-items: center;
    gap: 10px;
    text-decoration: none;
    color: white;
}

/* NAV */
nav {
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 40px 0;
}

nav ul {
    list-style: none;
    display: flex;
    gap: 40px;
}

nav li a {
    text-decoration: none;
    color: maroon;
    font-size: 1.2rem;
    font-weight: 600;
    padding: 12px 24px;
    border: 2px solid maroon;
    border-radius: 8px;
    transition: 0.3s ease;
    background-color: white;
}

nav li a:hover {
    background-color: maroon;
    color: white;
}

/* Form */
form {
    max-width: 400px;
    margin: 30px auto;
    padding: 20px;
    background: #fff;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
    display: flex;
    flex-direction: column;
    gap: 15px;
}

form label {
    font-weight: bold;
    margin-bottom: 4px;
}

form input[type="text"] {
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 6px;
    font-size: 1rem;
}

form input[type="submit"] {
    background-color: maroon;
    color: white;
    font-size: 1rem;
    padding: 10px;
    border: none;
    border-radius: 6px;
    cursor: pointer;
}

form button:hover{
    background-color: maroon;
    color: white;
    transition: 0.3s ease;
}
```

Functionality

Home page



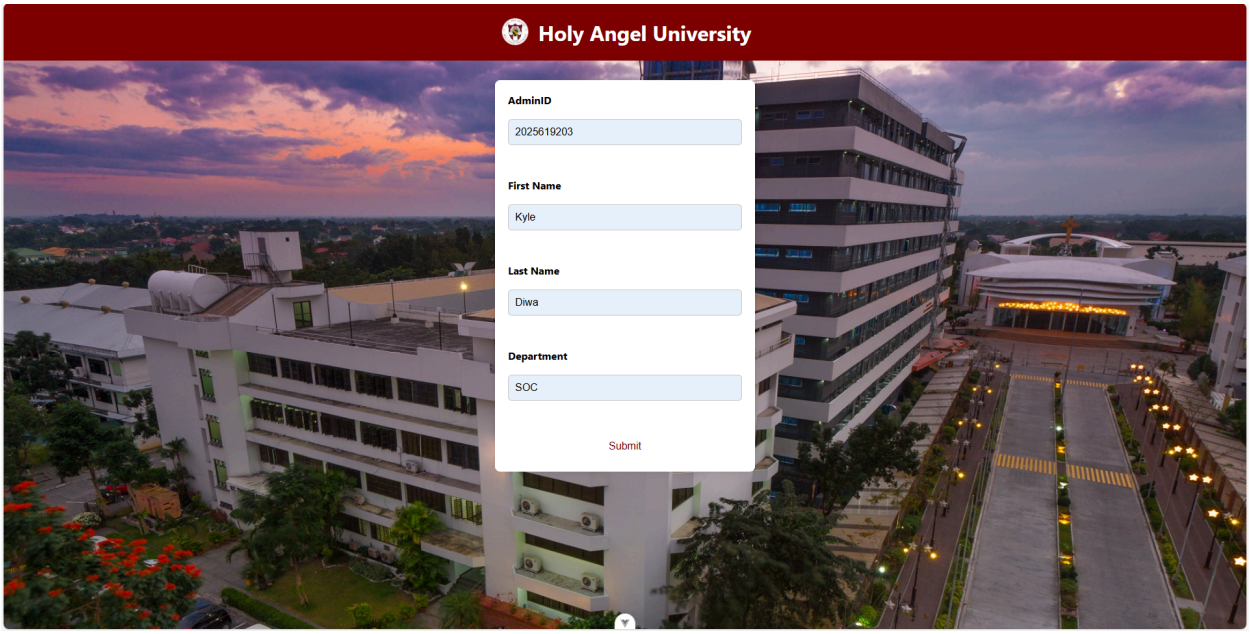
Student Sign-In



Terminal

```
Student saved: {
  studentID: '20943268',
  firstName: 'Eldrin ',
  lastName: 'Pineda',
  section: 'WD-302',
  _id: new ObjectId('68c8c9c600f82c045ed3222c'),
  __v: 0
}
```

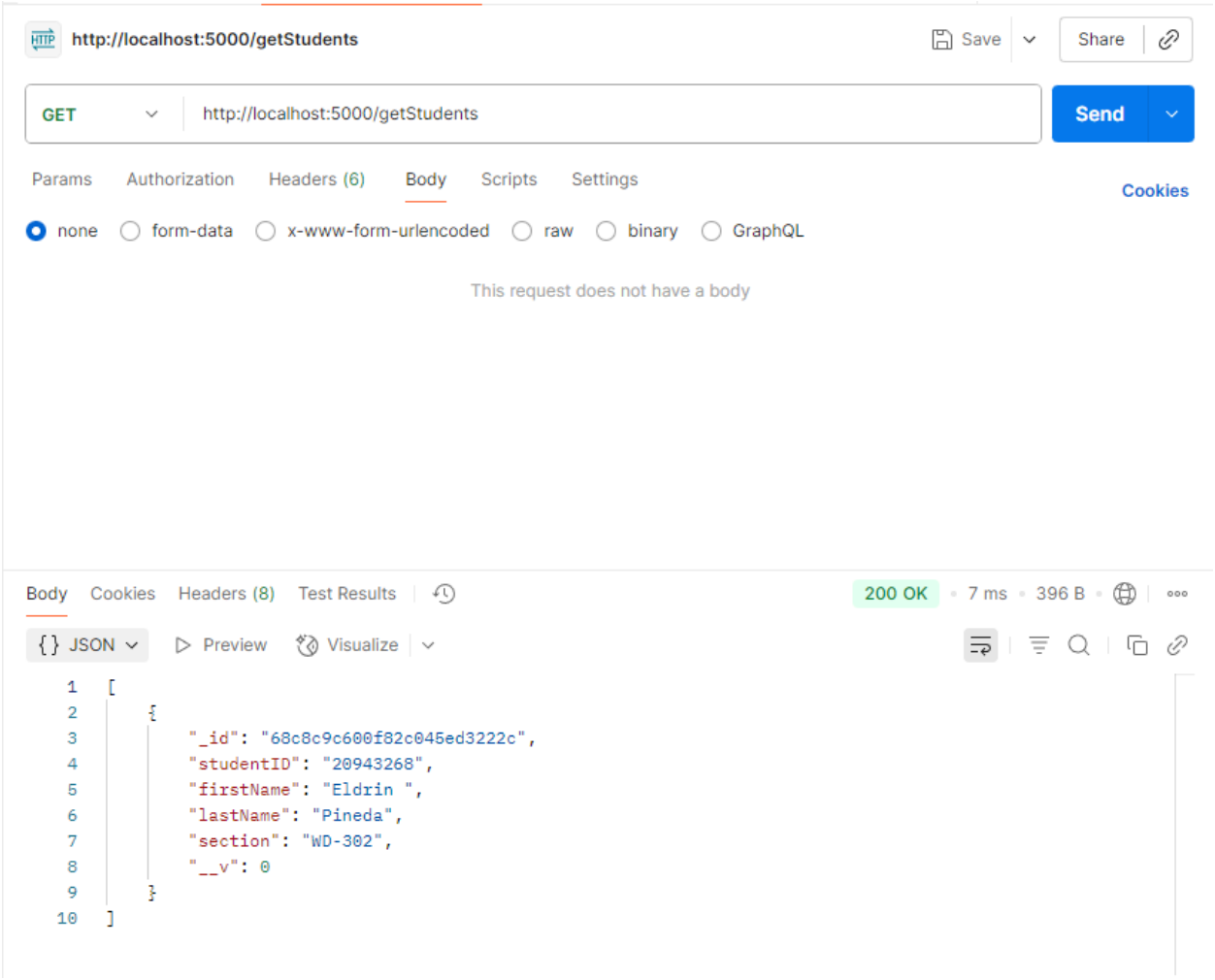
Admin Sign-In



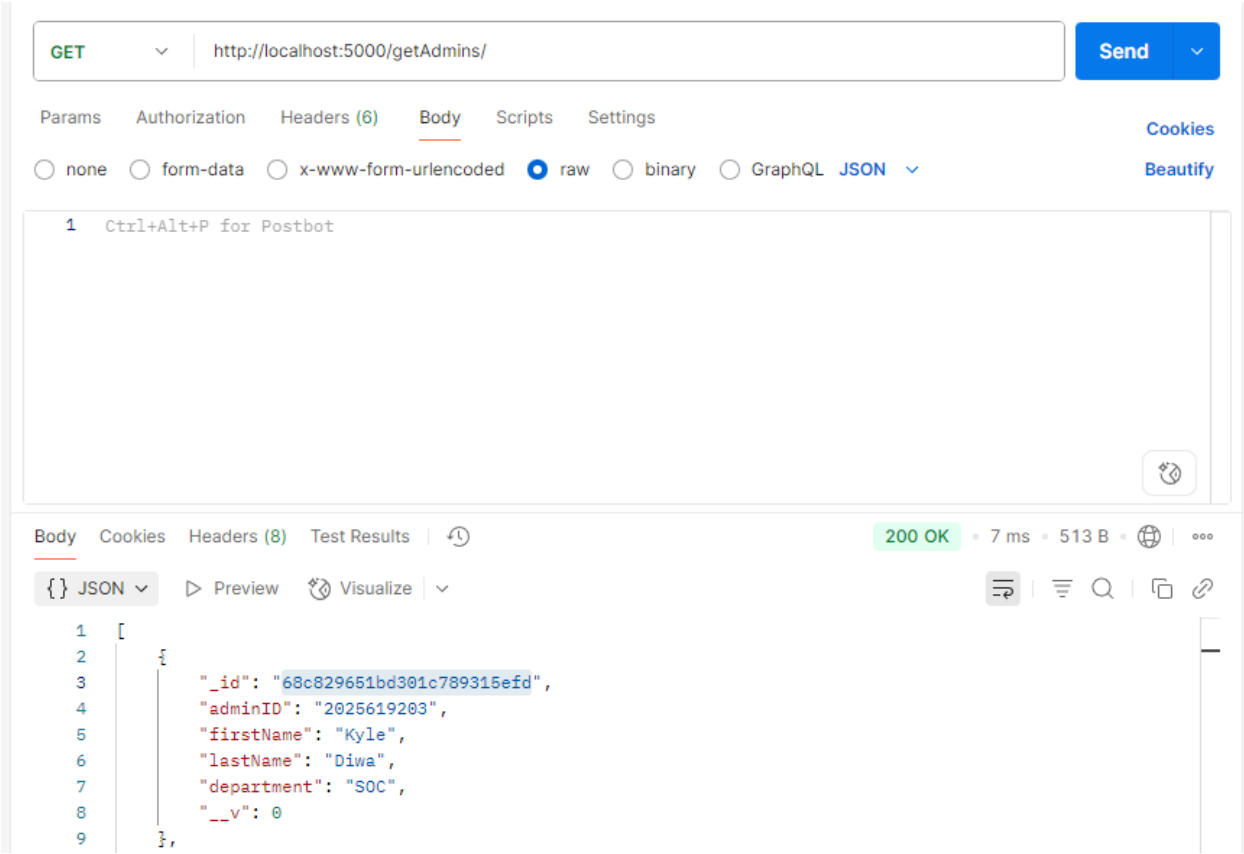
Terminal

```
Admin saved: {
  adminID: '2025619203',
  firstName: 'Kyle',
  lastName: 'Diwa',
  department: 'SOC',
  _id: new ObjectId('68c8c9dc00f82c045ed3222e'),
  __v: 0
}
```

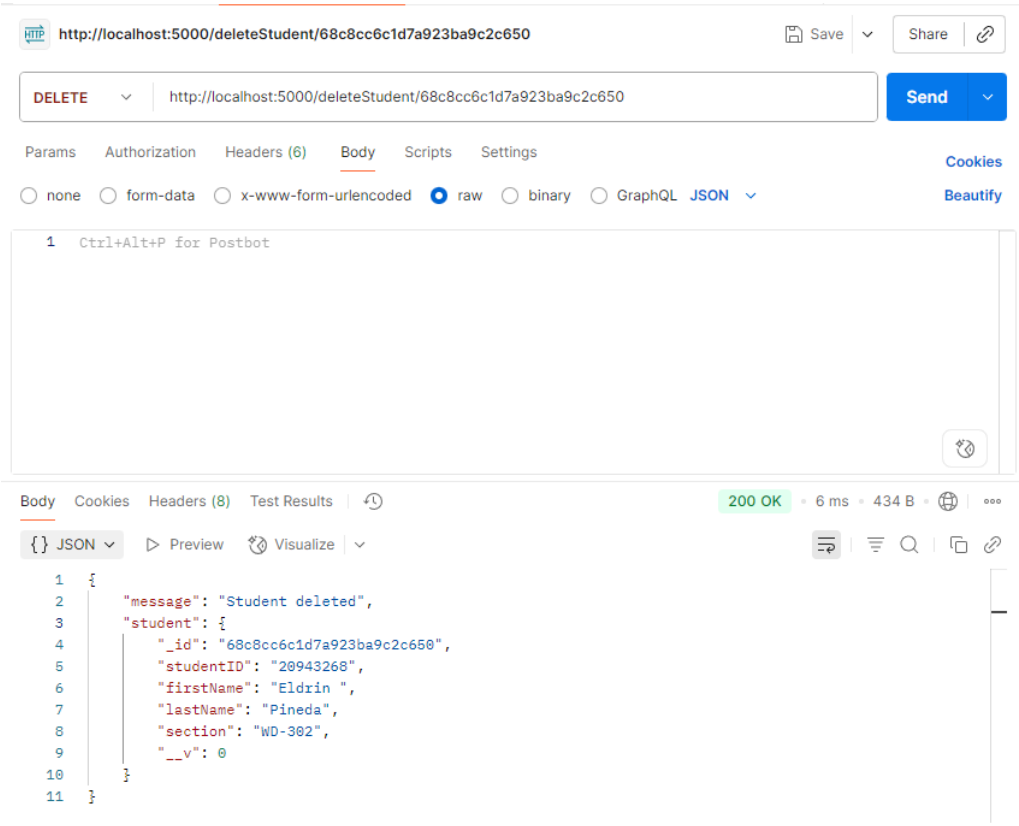
Postman
getStudents



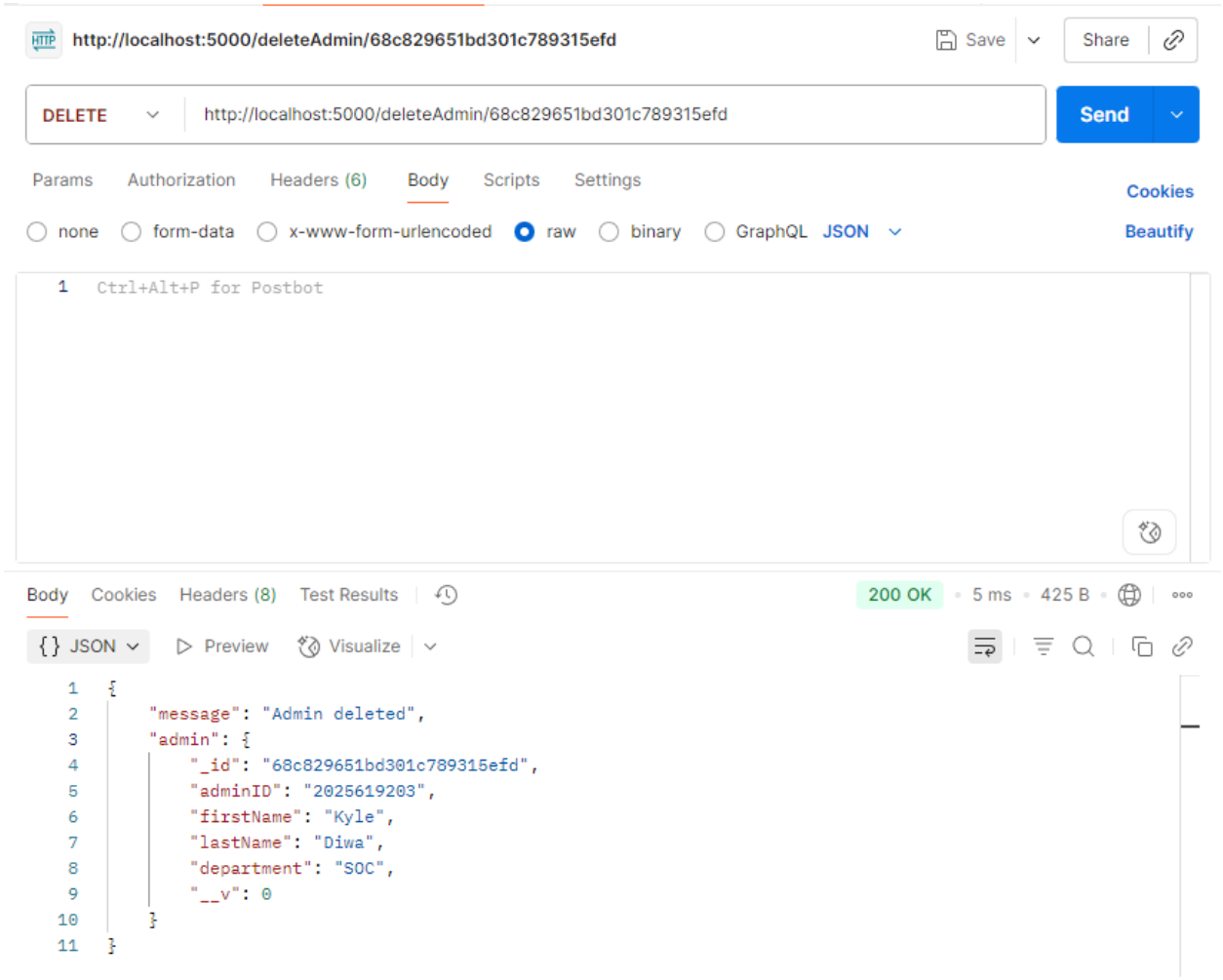
getAdmins



DELETE
deleteStudent



deleteAdmin



Reflection:

This activity was one of the most challenging 20-point tasks I have ever encountered. Honestly, without AI, I don't think I would have been able to finish it. It was a completely new environment for me since I hadn't used Vue before, so I wasn't familiar with its structure and functions. Jokes aside, it was excellent practice for our upcoming final projects. Working on this activity gave me ideas on how we can utilize Vue for our project. I was introduced to new functions and imports that are usually straightforward in other programming languages, but in Vue, even simple routing required additional imports. Similarly, on the backend, I learned to use Mongoose for MongoDB.

Overall, this activity was a good challenge. It helped me understand my limits in programming and showed me that there is still much room for improvement.