

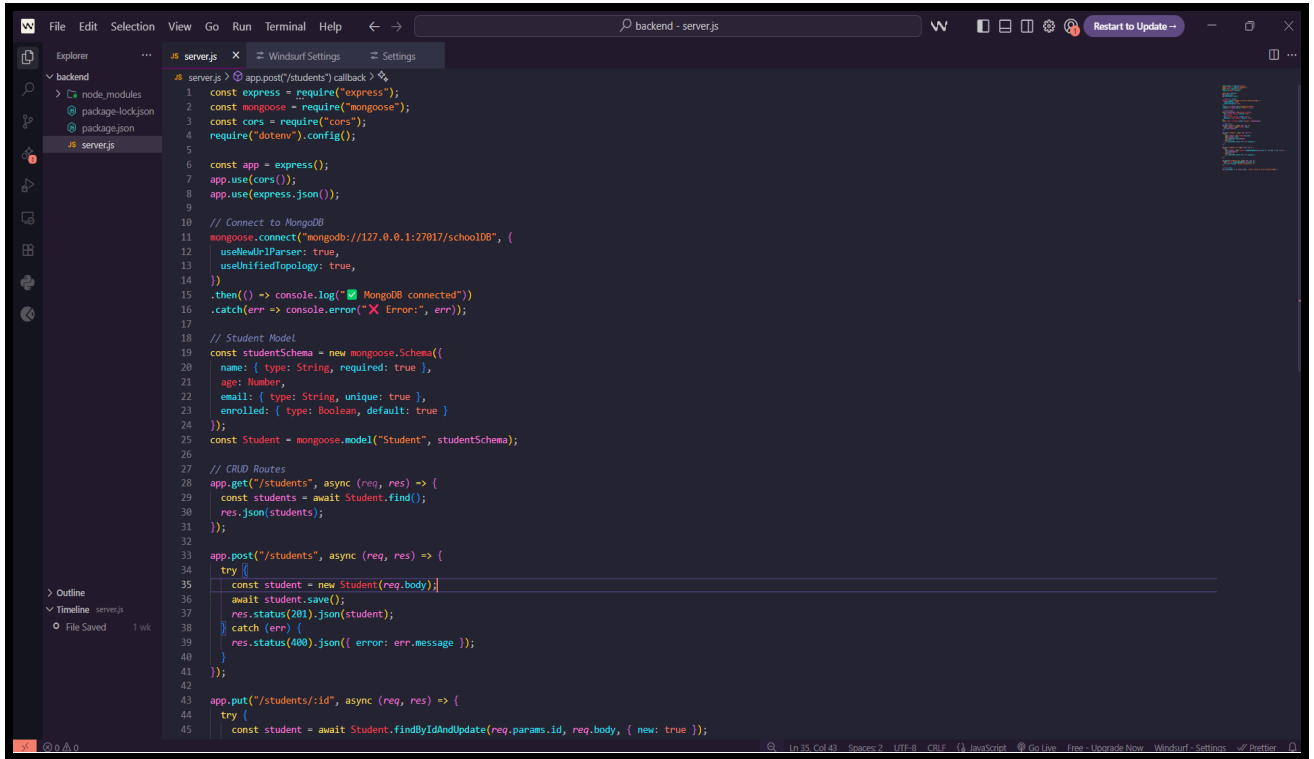
EXPERIMENT 4

RESTful API design with MongoDB + Mongoose integration

1. CODE

Figure 1.1: app.js File

```
src > App.js M X
src > App.js > App > [0] fetchStudents > useCallback callback >
1 import React, { useState, useEffect, useCallba } from 'react';
2 import axios from 'axios';
3 import { ToastContainer, toast } from 'react-toastify';
4 import 'react-toastify/dist/ReactToastify.css';
5
6 function App() {
7   const [students, setStudents] = useState([]);
8   const [form, setForm] = useState({ name: '', age: '', email: '' });
9   const [search, setSearch] = useState('');
10  const [isLoading, setIsLoading] = useState(false);
11  const [editingId, setEditingId] = useState(null);
12  const [errors, setErrors] = useState({});
13
14  const API_URL = "http://localhost:5000/students";
15
16  const fetchStudents = useCallba(async () => {
17    try {
18      setIsLoading(true);
19      const res = await axios.get(API_URL);
20      setStudents(res.data);
21    } catch (error) {
22      toast.error('Failed to fetch students');
23      console.error('Error fetching students:', error);
24    } finally {
25      setIsLoading(false);
26    }
27  }, []);
28
29  useEffect(() => {
30    fetchStudents();
31  }, [fetchStudents]);
32
33  const validateForm = () => {
34    const newErrors = {};
35    if (!form.name.trim()) newErrors.name = 'Name is required';
36    if (!form.email.trim()) {
37      newErrors.email = 'Email is required';
38    } else if (!/^[a-zA-Z0-9\.\%+\-]+@test\.com$/i.test(form.email)) {
39      newErrors.email = 'Email is invalid';
40    }
41    if (!form.age) newErrors.age = 'Age is required';
42    else if (!isNaN(form.age) || form.age < 1 || form.age > 120) {
43      newErrors.age = 'Please enter a valid age (1-120)';
44    }
45    setErrors(newErrors);
46    return Object.keys(newErrors).length === 0;
47  };
48
49  const handleSubmit = async (e) => {
50    e.preventDefault();
51    if (!validateForm()) return;
52
53    try {
54      setIsLoading(true);
55      if (editingId) {
```



```
1 const express = require("express");
2 const mongoose = require("mongoose");
3 const cors = require("cors");
4 require("dotenv").config();
5
6 const app = express();
7 app.use(cors());
8 app.use(express.json());
9
10 // Connect to MongoDB
11 mongoose.connect("mongodb://127.0.0.1:27017/schoolDB", {
12   useNewUrlParser: true,
13   useUnifiedTopology: true,
14 });
15 .then(() => console.log("MongoDB connected"))
16 .catch(err => console.error("Error:", err));
17
18 // Student Model
19 const studentSchema = new mongoose.Schema({
20   name: { type: String, required: true },
21   age: Number,
22   email: { type: String, unique: true },
23   enrolled: { type: Boolean, default: true }
24 });
25 const Student = mongoose.model("Student", studentSchema);
26
27 // CRUD Routes
28 app.get("/students", async (req, res) => {
29   const students = await Student.find();
30   res.json(students);
31 });
32
33 app.post("/students", async (req, res) => {
34   try {
35     const student = new Student(req.body);
36     await student.save();
37     res.status(201).json(student);
38   } catch (err) {
39     res.status(400).json({ error: err.message });
40   }
41 });
42
43 app.put("/students/:id", async (req, res) => {
44   try {
45     const student = await Student.findByIdAndUpdate(req.params.id, req.body, { new: true });
```

Figure 1.2: server.js File

2. OUTPUT

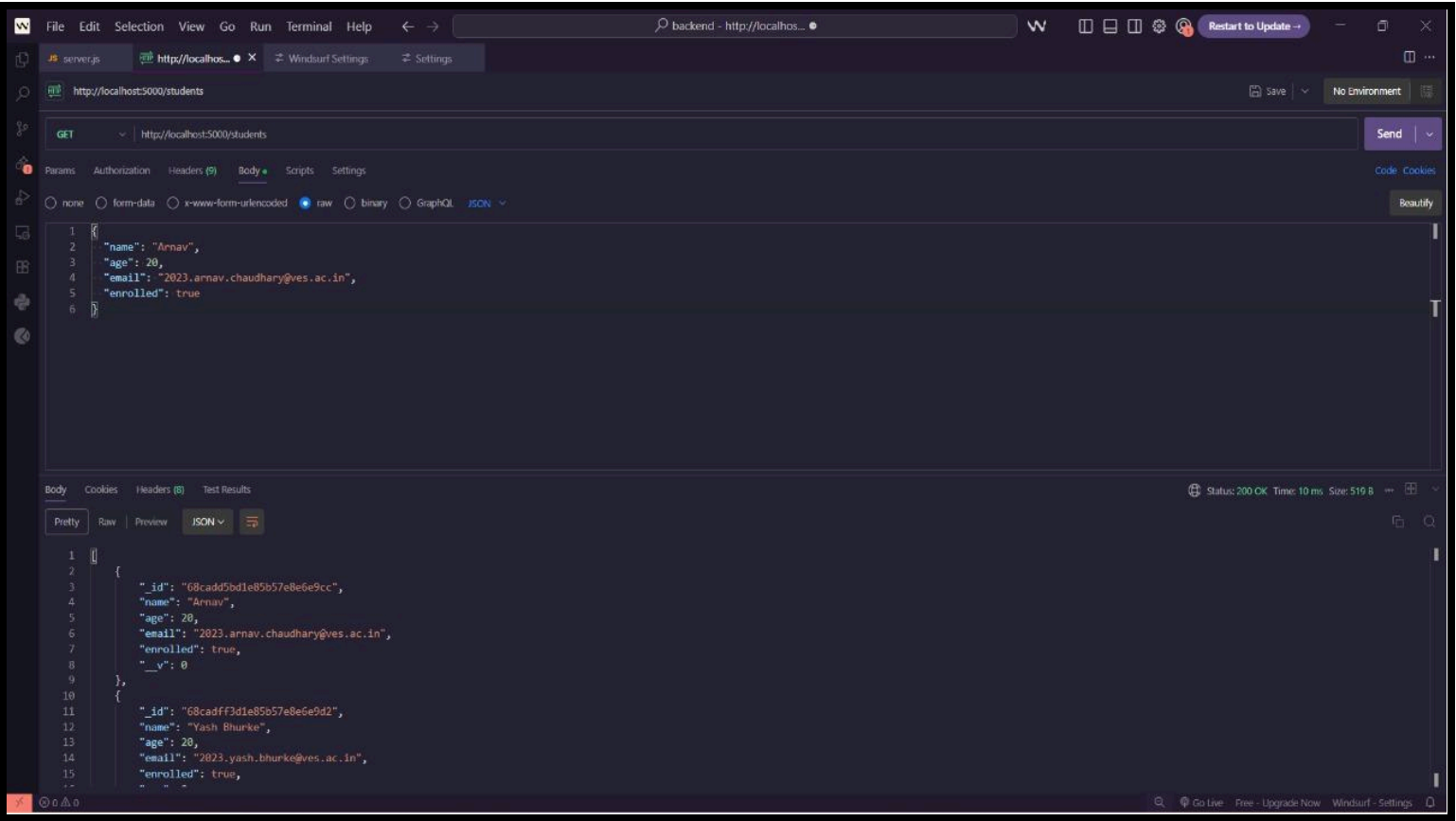


Figure 2.1:POSTMAN Get Query

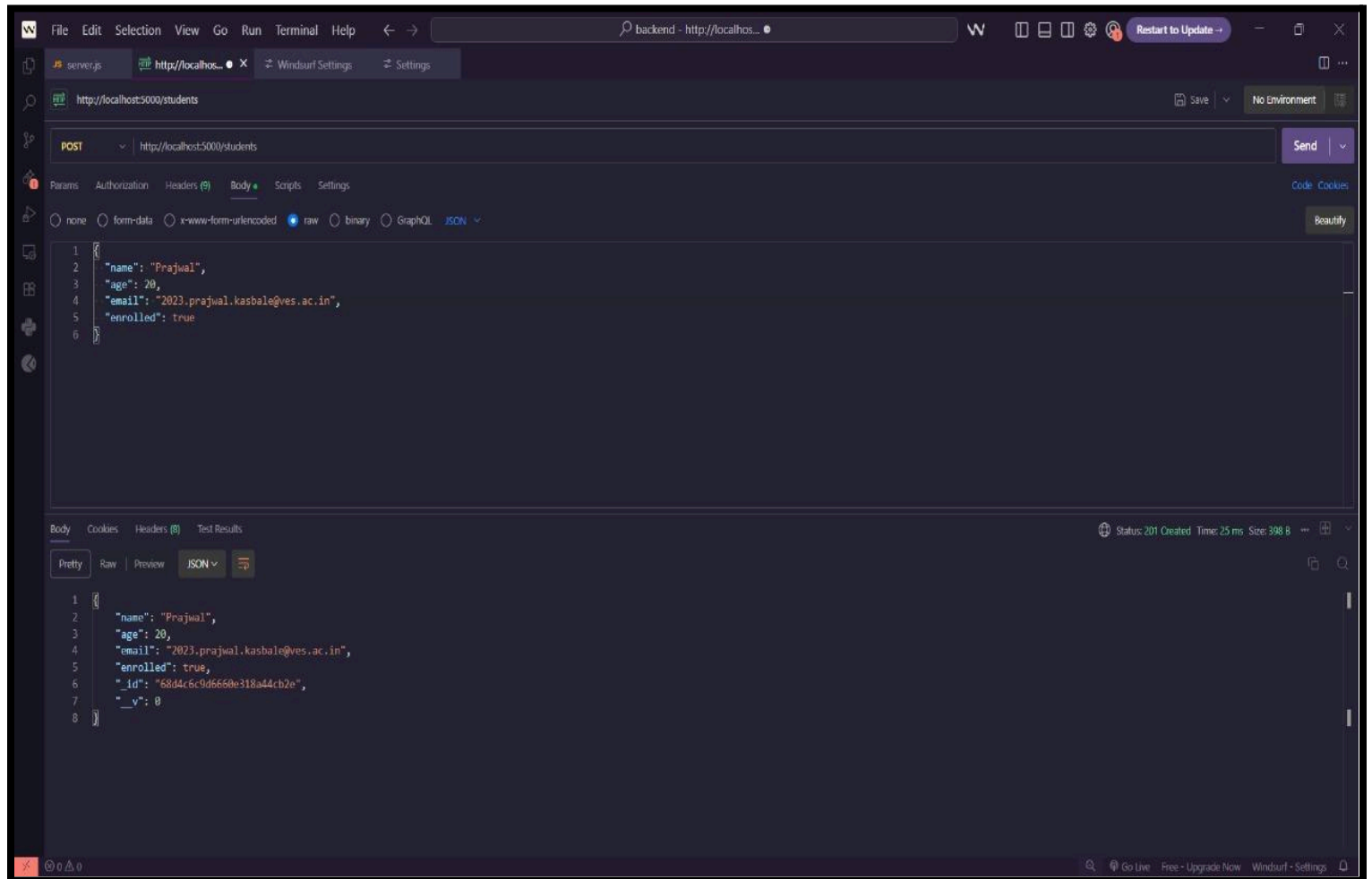


Figure 2.2:Postman Post Query

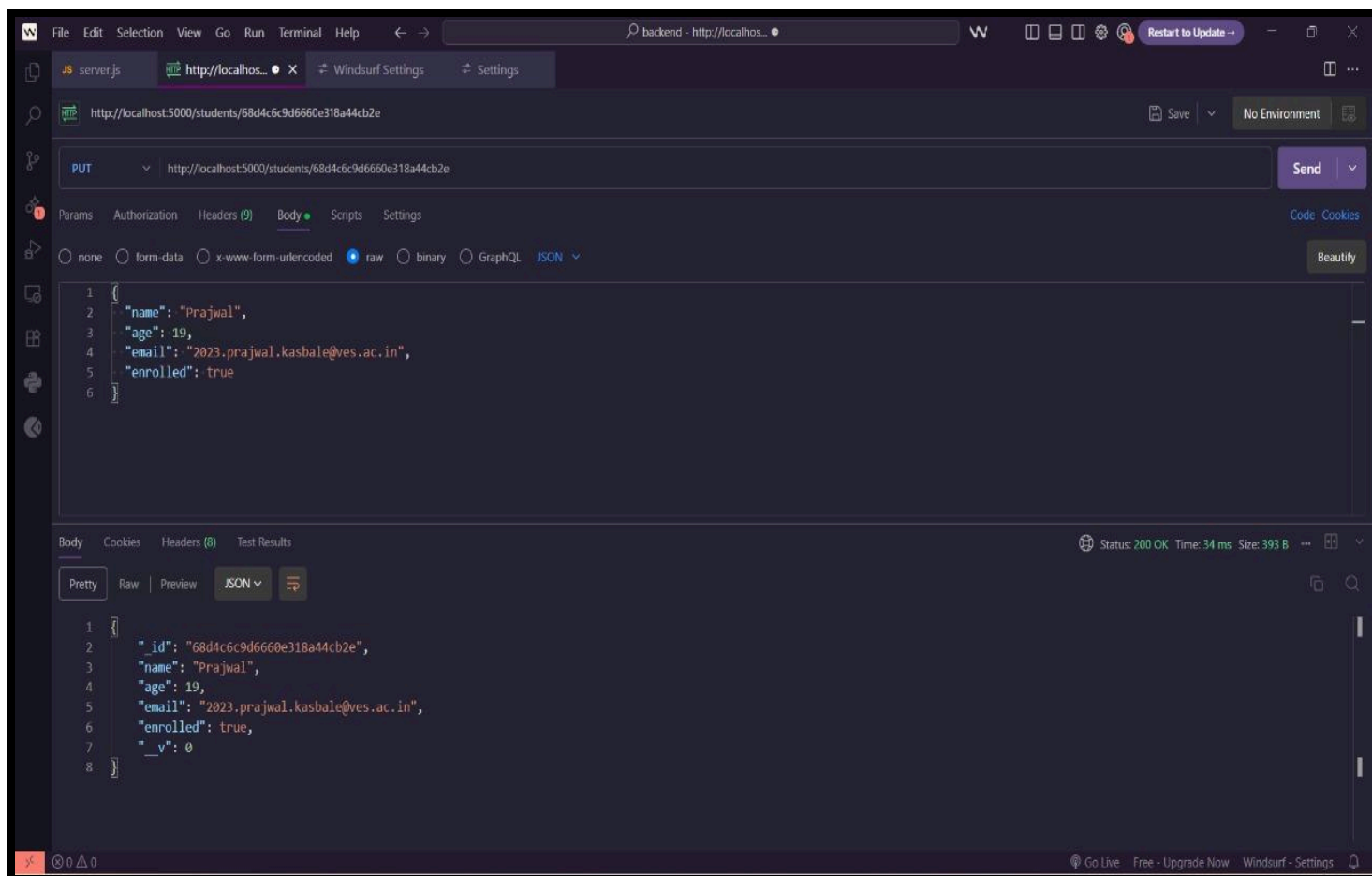


Figure 2.3:Postman PUT Query

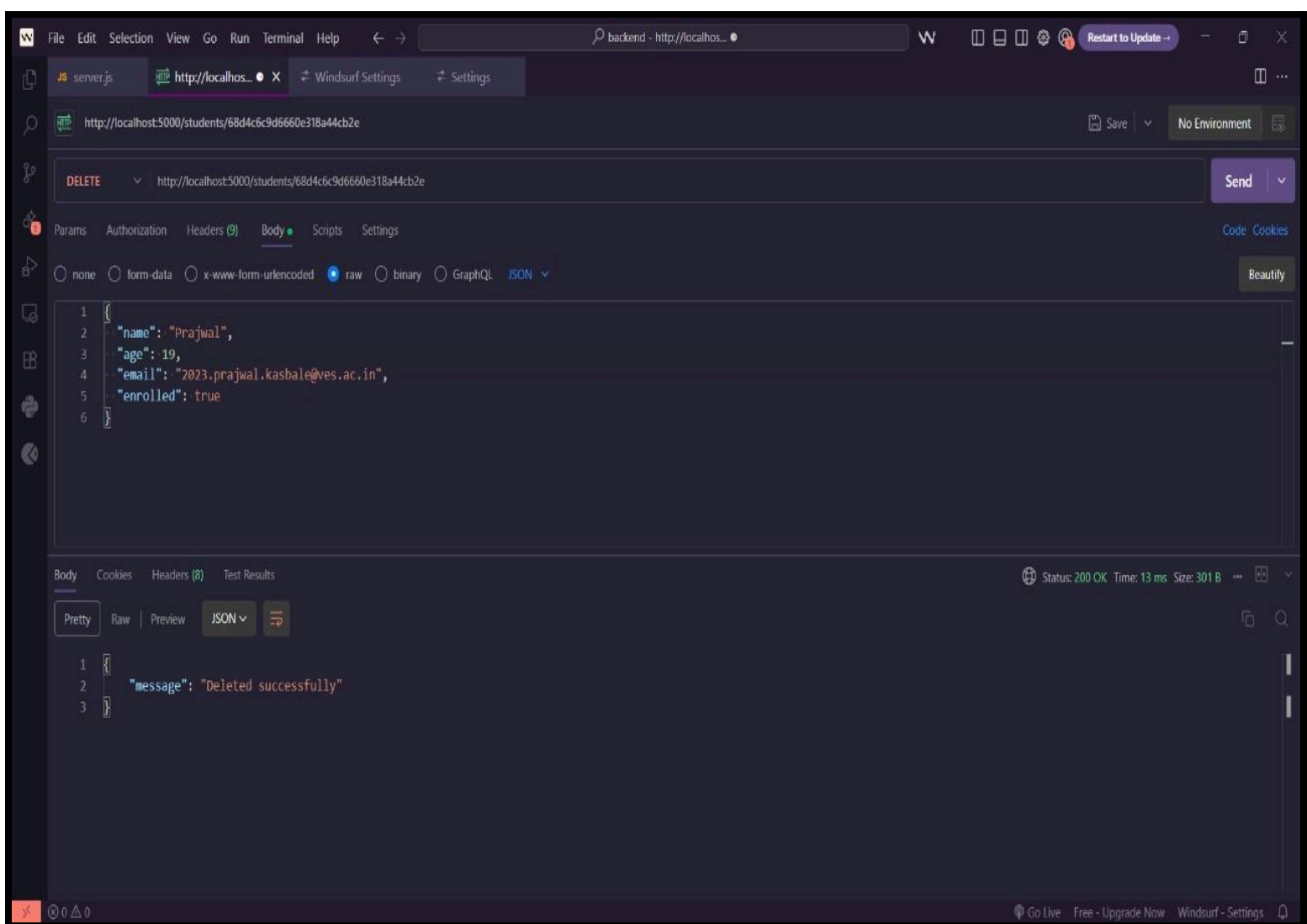


Figure 2.4: Postman Delete Query

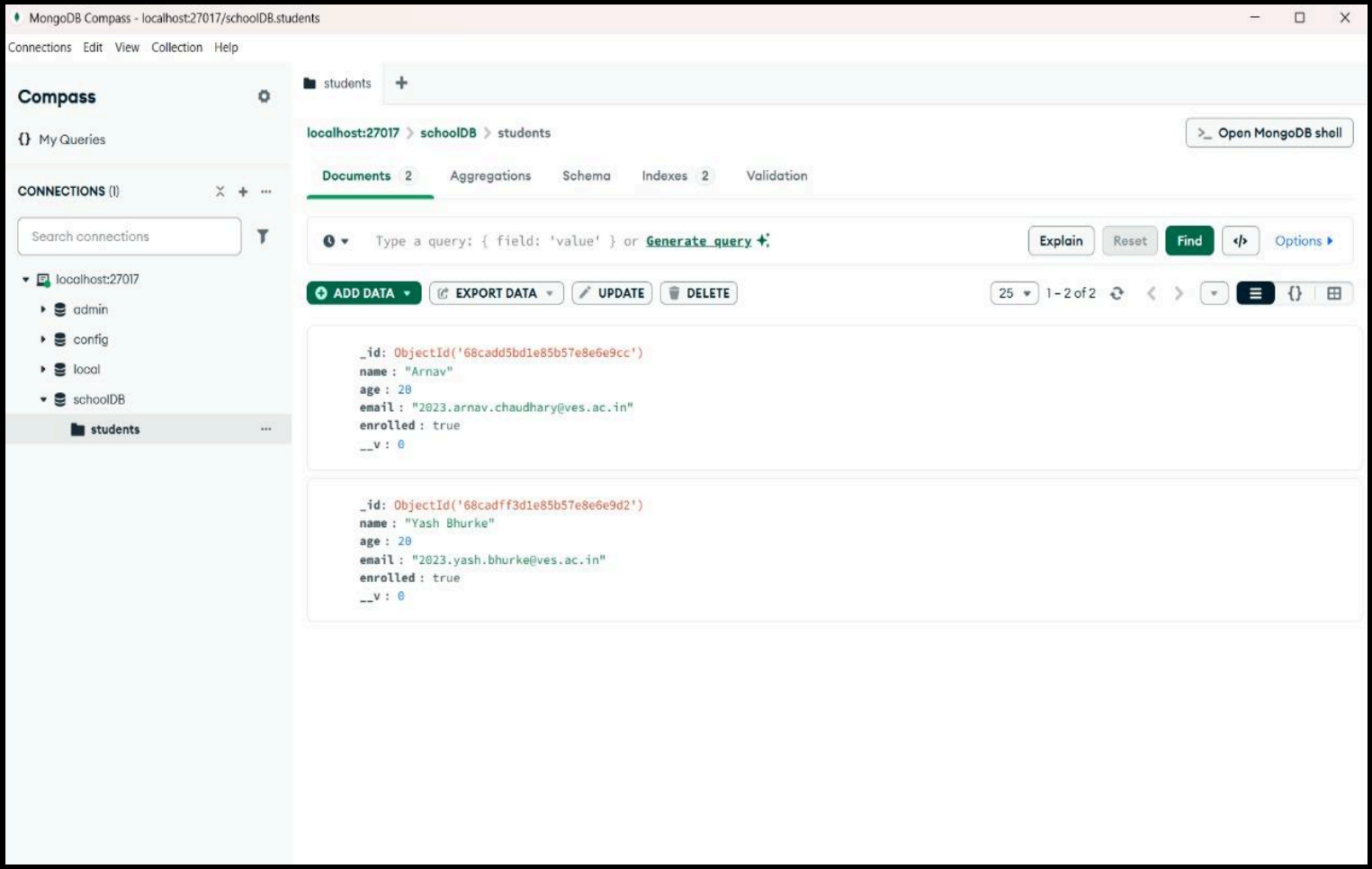


Figure 2.5: MongoDB Compass Database View