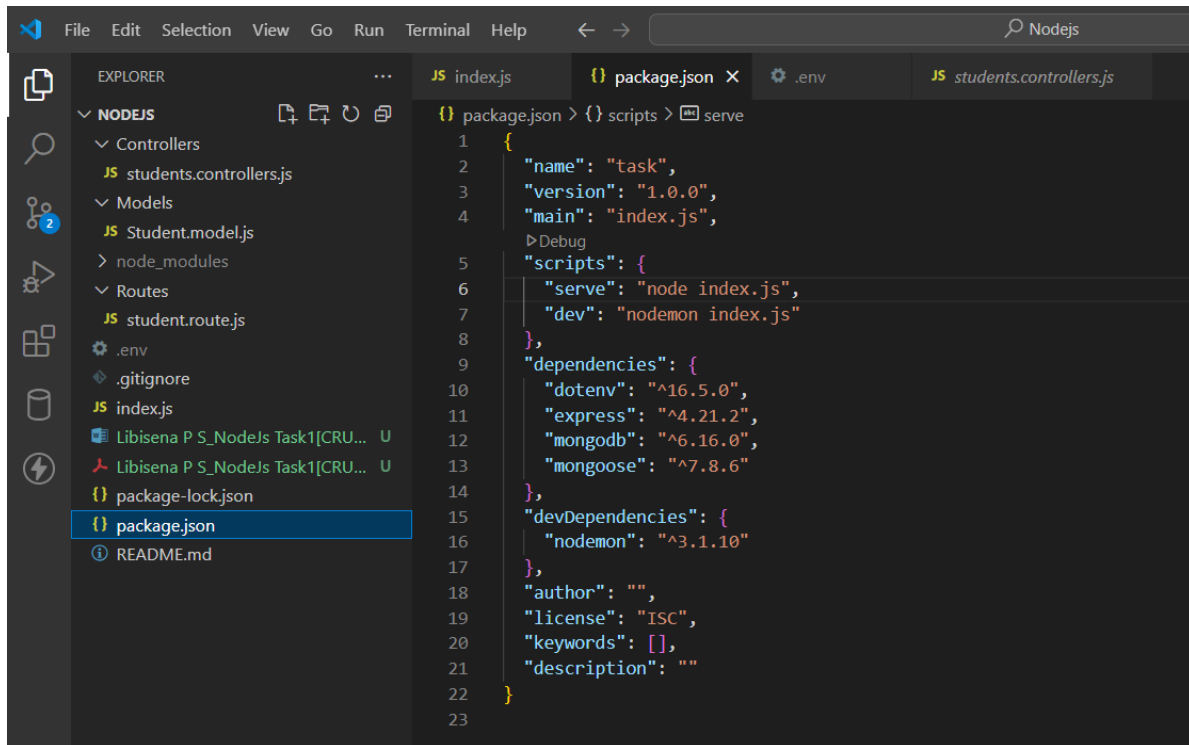


Node Js Task 1

GitHub Repository : <https://github.com/Libisenaps/NodeTask1-CRUD>

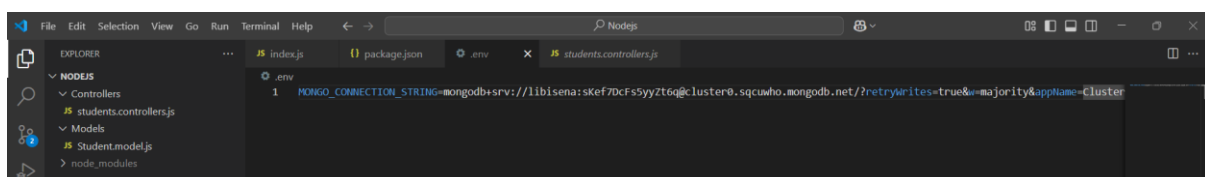
To build a College Student Details Application using Node.js with CRUD Operations.



The screenshot shows the Visual Studio Code interface. The Explorer view on the left displays the project structure, including the 'package.json' file which is selected. The Source Control view on the right shows the contents of the 'package.json' file. The file is a JSON object with the following structure:

```
1 {
2   "name": "task",
3   "version": "1.0.0",
4   "main": "index.js",
5   "scripts": {
6     "serve": "node index.js",
7     "dev": "nodemon index.js"
8   },
9   "dependencies": {
10    "dotenv": "^16.5.0",
11    "express": "^4.21.2",
12    "mongodb": "^6.16.0",
13    "mongoose": "^7.8.6"
14  },
15  "devDependencies": {
16    "nodemon": "^3.1.10"
17  },
18  "author": "",
19  "license": "ISC",
20  "keywords": [],
21  "description": ""
22 }
```

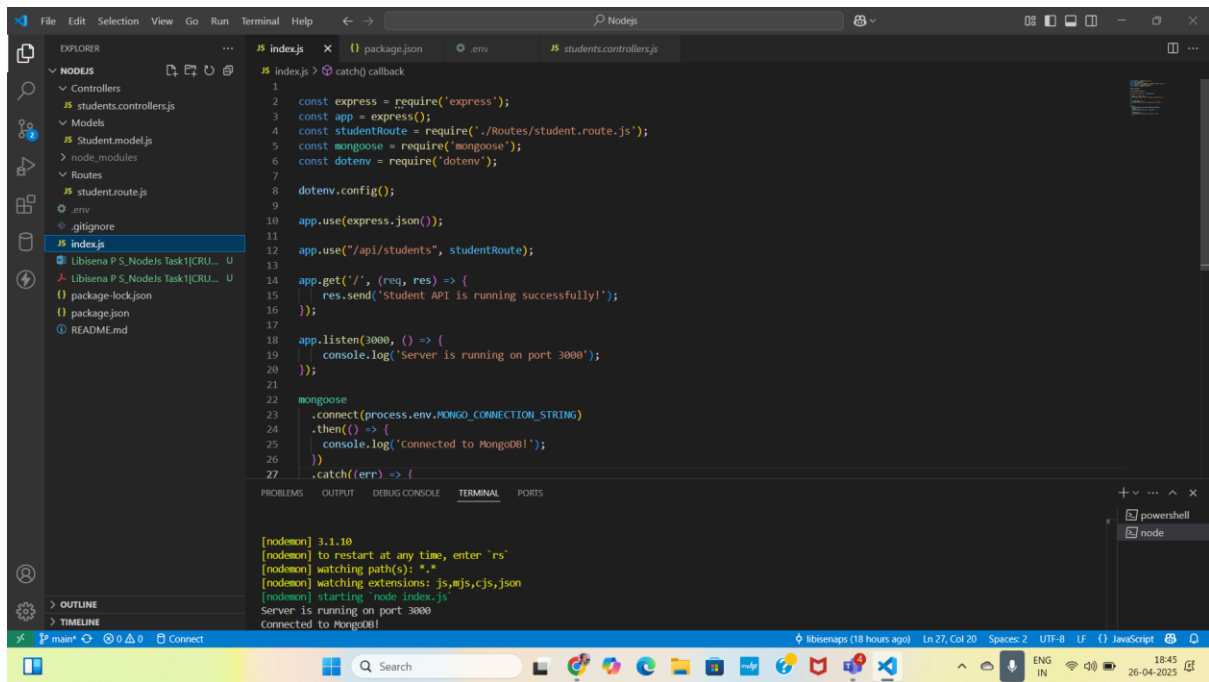
- ✓ Downloading the necessary packages initially and verifying that the dependencies are correctly updated in the package.json file.



The screenshot shows the Visual Studio Code interface. The Explorer view on the left displays the project structure, including the '.env' file which is selected. The Source Control view on the right shows the contents of the '.env' file. The file contains a single line of text:

```
1 MONGO_CONNECTION_STRING=mongodb+srv://libisena:sKef7DcF5yyZt6q@cluster0.sqcuwh.mongodb.net/?retryWrites=true&majority=appliance=Cluster
```

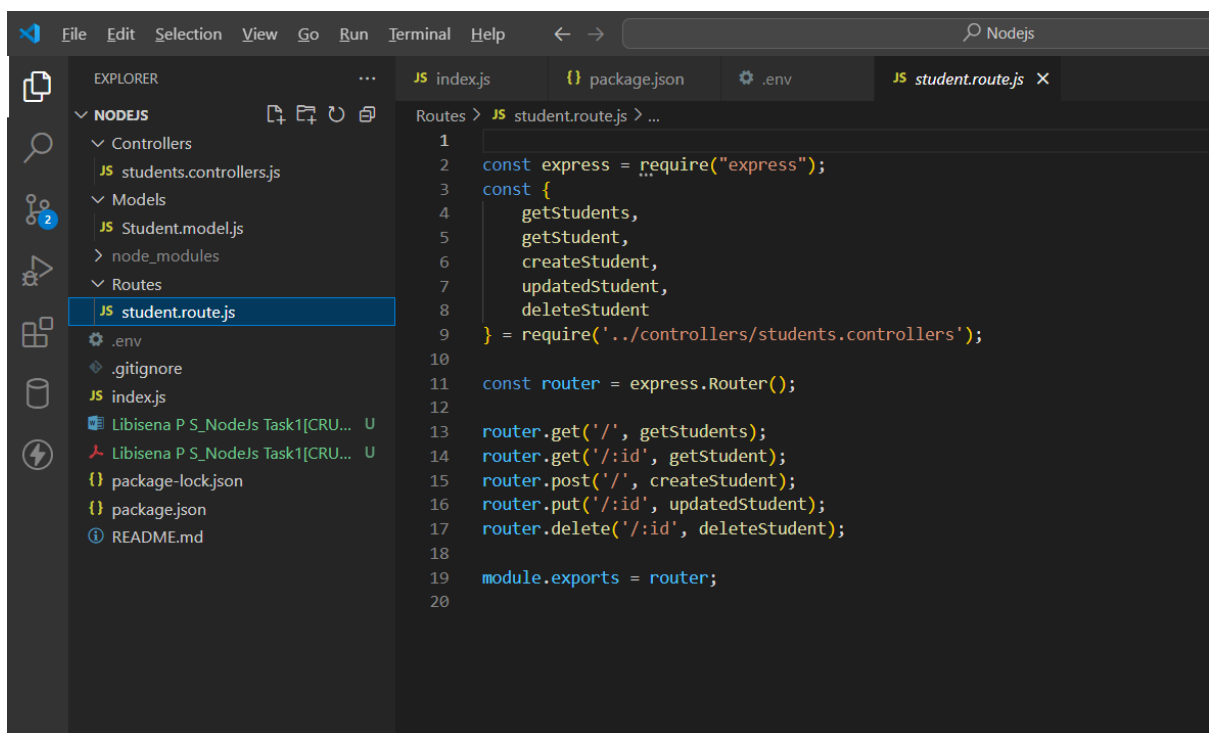
- ✓ MongoDB Connectivity: The .env file is used to store the connection string taken from MongoDB.



```
1  // catch() callback
2  const express = require('express');
3  const app = express();
4  const studentRoute = require('./Routes/student.route.js');
5  const mongoose = require('mongoose');
6  const dotenv = require('dotenv');
7
8  dotenv.config();
9
10 app.use(express.json());
11
12 app.use('/api/students', studentRoute);
13
14 app.get('/', (req, res) => {
15   res.send('Student API is running successfully!');
16 });
17
18 app.listen(3000, () => {
19   console.log('Server is running on port 3000');
20 });
21
22 mongoose
23   .connect(process.env.MONGO_CONNECTION_STRING)
24   .then(() => {
25     console.log('Connected to MongoDB!');
26   })
27   .catch(err => {
```

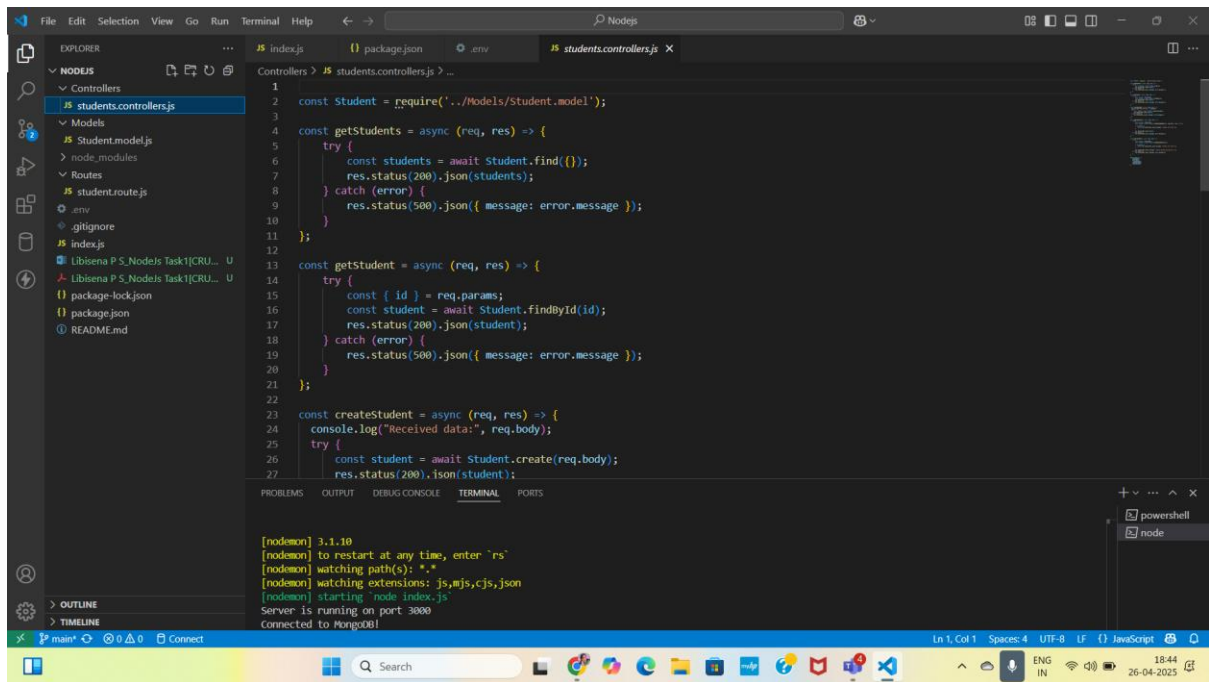
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Server is running on port 3000
Connected to MongoDB!

- ✓ The index.js file is used to set up the server, connect to the database, and define the main routes for the application.



```
1
2  const express = require("express");
3  const {
4    getStudents,
5    getStudent,
6    createStudent,
7    updatedStudent,
8    deleteStudent
9  } = require('../controllers/students.controllers');
10
11 const router = express.Router();
12
13 router.get('/', getStudents);
14 router.get('/:id', getStudent);
15 router.post('/', createStudent);
16 router.put('/:id', updatedStudent);
17 router.delete('/:id', deleteStudent);
18
19 module.exports = router;
20
```

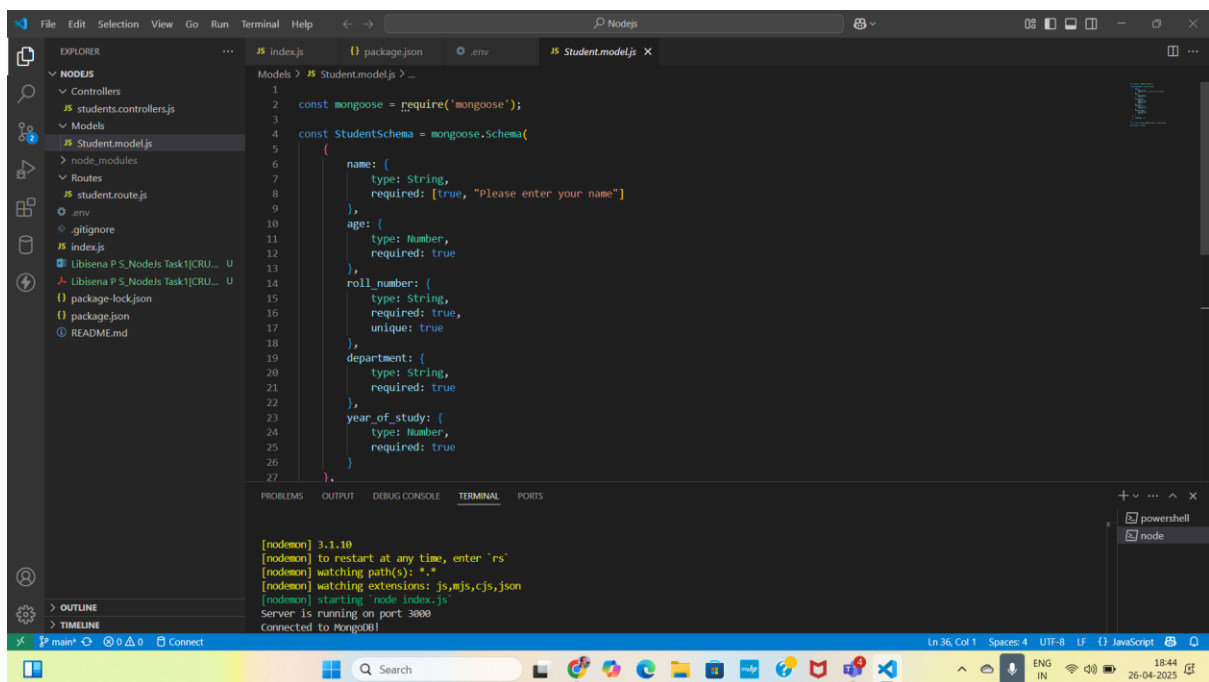
- ✓ Routes are used to handle operations like adding, fetching, and managing student details.



```
1 const Student = require('../Models/Student.model');
2
3
4 const getStudents = async (req, res) => {
5   try {
6     const students = await Student.find({});
7     res.status(200).json(students);
8   } catch (error) {
9     res.status(500).json({ message: error.message });
10  }
11 };
12
13
14 const getStudent = async (req, res) => {
15   try {
16     const { id } = req.params;
17     const student = await Student.findById(id);
18     res.status(200).json(student);
19   } catch (error) {
20     res.status(500).json({ message: error.message });
21   }
22 };
23
24 const createStudent = async (req, res) => {
25   console.log("Received data:", req.body);
26   try {
27     const student = await Student.create(req.body);
28     res.status(200).json(student);
29   } catch (error) {
30     res.status(500).json({ message: error.message });
31   }
32 };
33
34 module.exports = { getStudents, getStudent, createStudent };
```

[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Server is running on port 3000
Connected to MongoDB!

- ✓ Controllers handle the logic for different requests, such as saving and retrieving student details from the database.
- ✓ Models define the structure of the student data and how it will be stored in the MongoDB database.

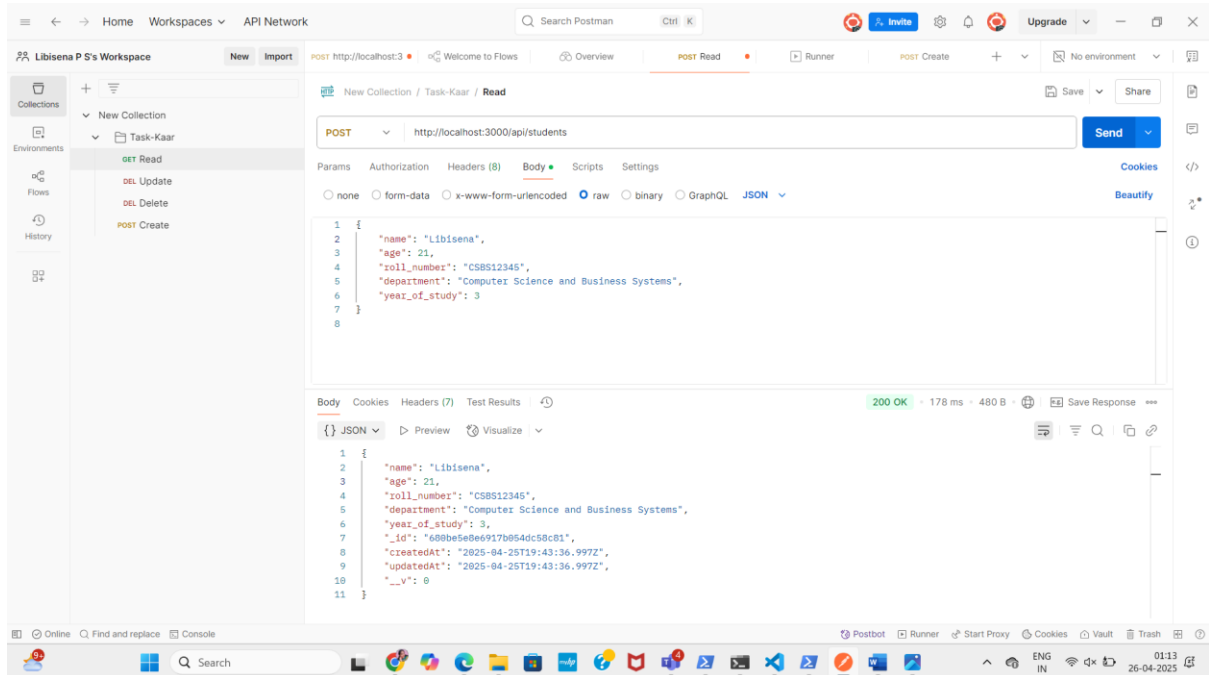


```
1 const mongoose = require('mongoose');
2
3
4 const StudentSchema = mongoose.Schema(
5   {
6     name: {
7       type: String,
8       required: [true, "Please enter your name"]
9     },
10    age: {
11      type: Number,
12      required: true
13    },
14    roll_number: {
15      type: String,
16      required: true,
17      unique: true
18    },
19    department: {
20      type: String,
21      required: true
22    },
23    year_of_study: {
24      type: Number,
25      required: true
26    }
27  },
28  {
29    timestamps: true
30  }
31 );
32
33 module.exports = mongoose.model('Student', StudentSchema);
```

[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Server is running on port 3000
Connected to MongoDB!

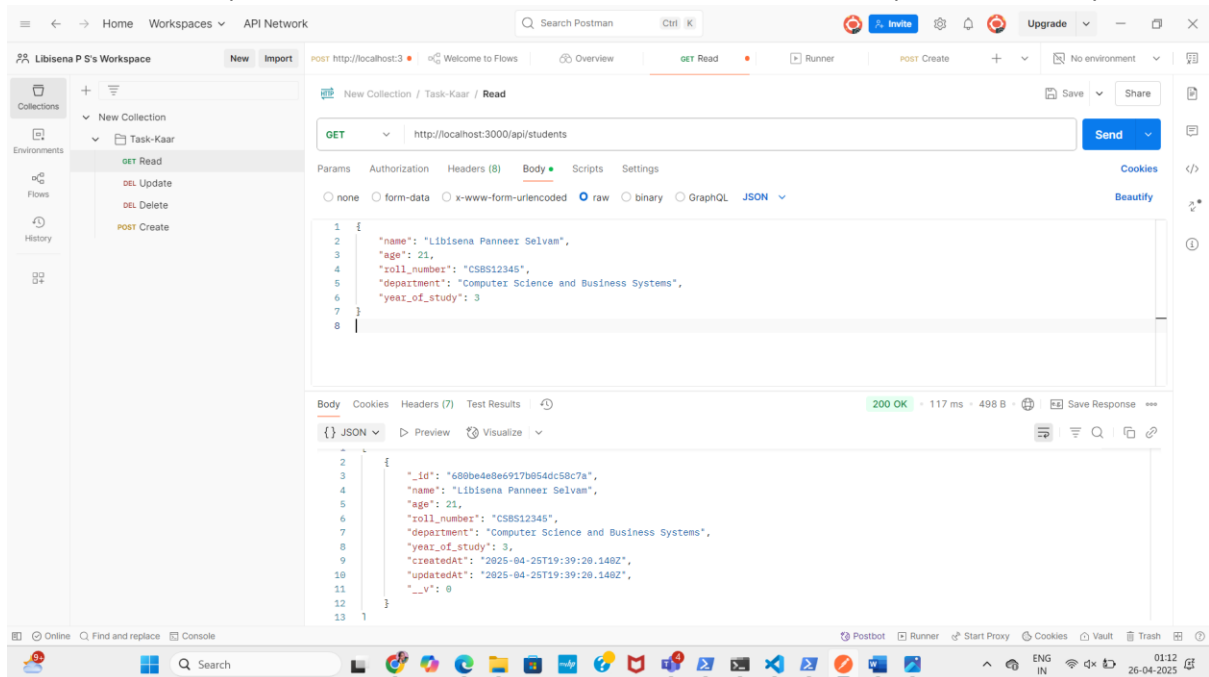
POST - Create

You send a POST request with student details (like name, age, department) to add a new student into the database.



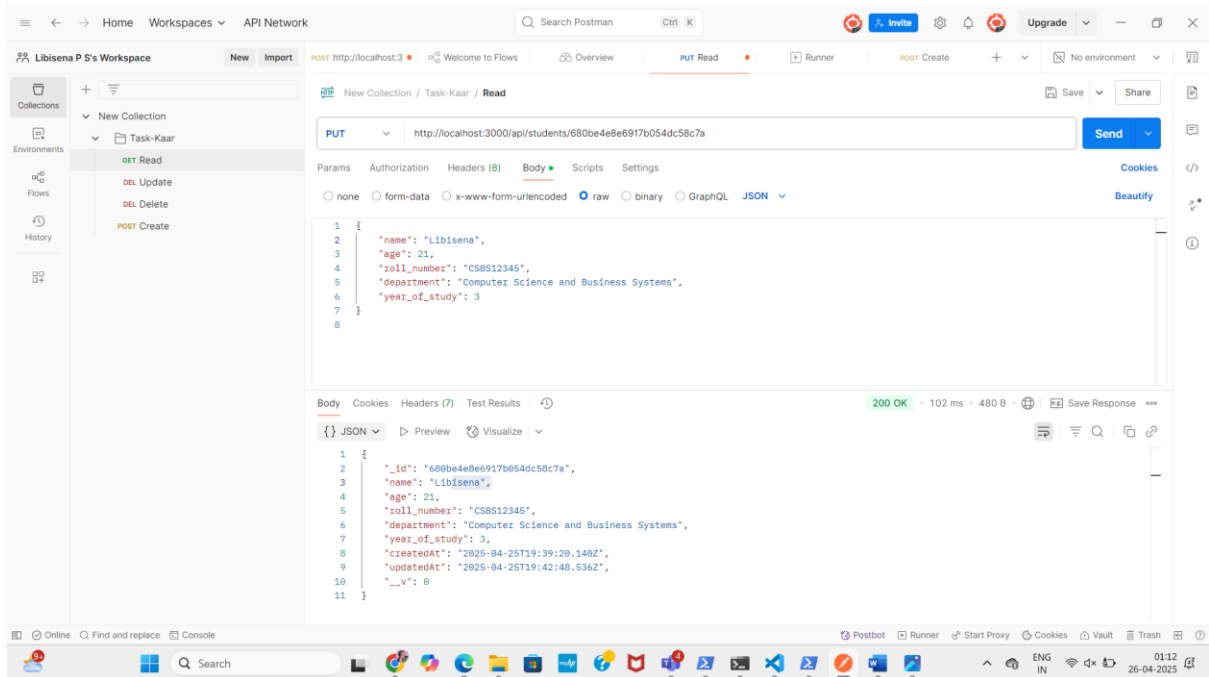
GET – Read

You send a GET request to fetch student details — either all students or a specific student by ID.



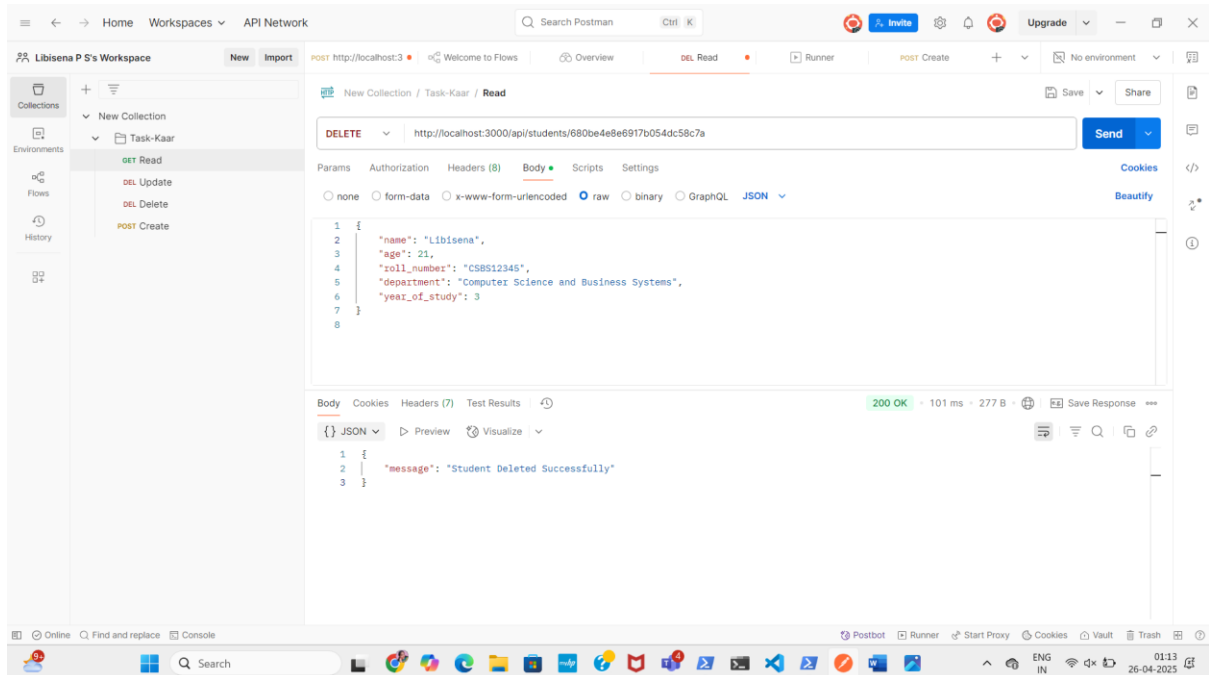
PUT – Update

You send a PUT request to modify an existing student's information based on their ID.



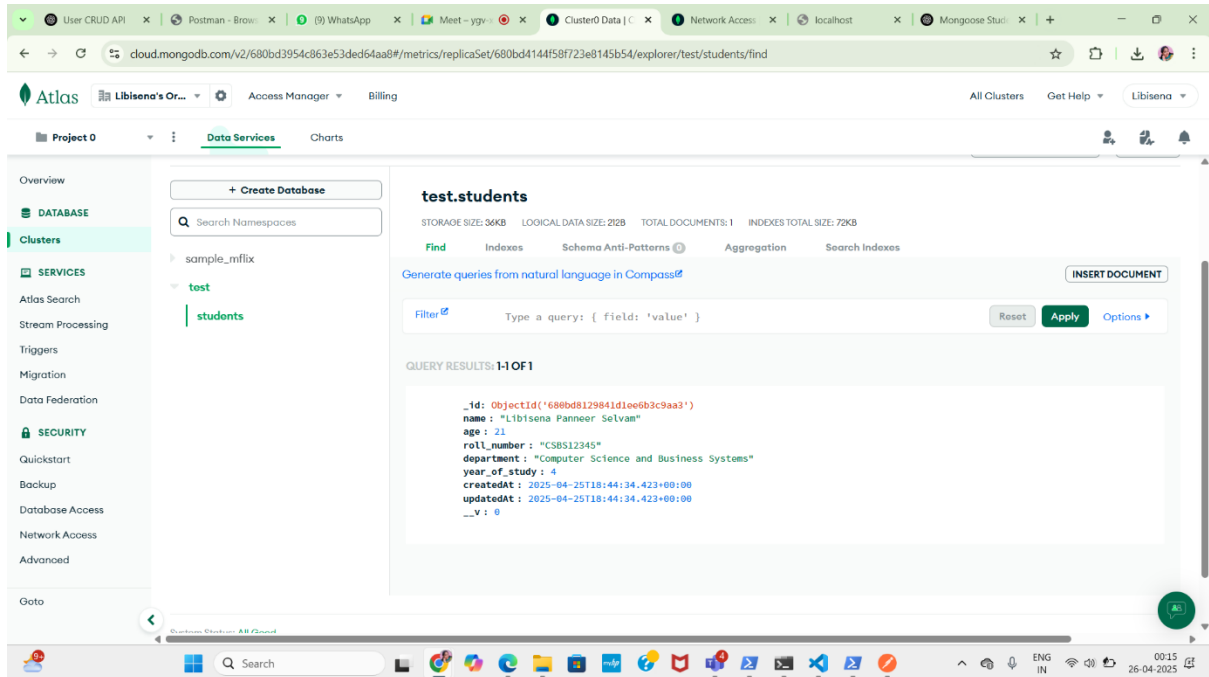
DELETE – Delete

You send a DELETE request to remove a student's record from the database using their ID.



MONGODB

Can see the student details using MongoDB



Node.js Connection with MongoDB and Performing CRUD Operations Using POST

A Student model was created using Mongoose in a Node.js environment, connected to a MongoDB database, and student details were successfully received and logged in the terminal.

