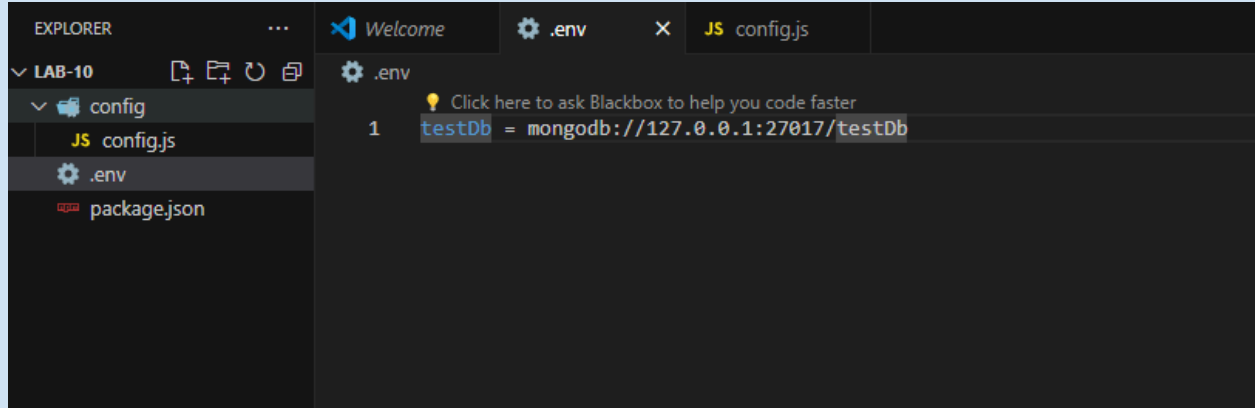


LAB-10

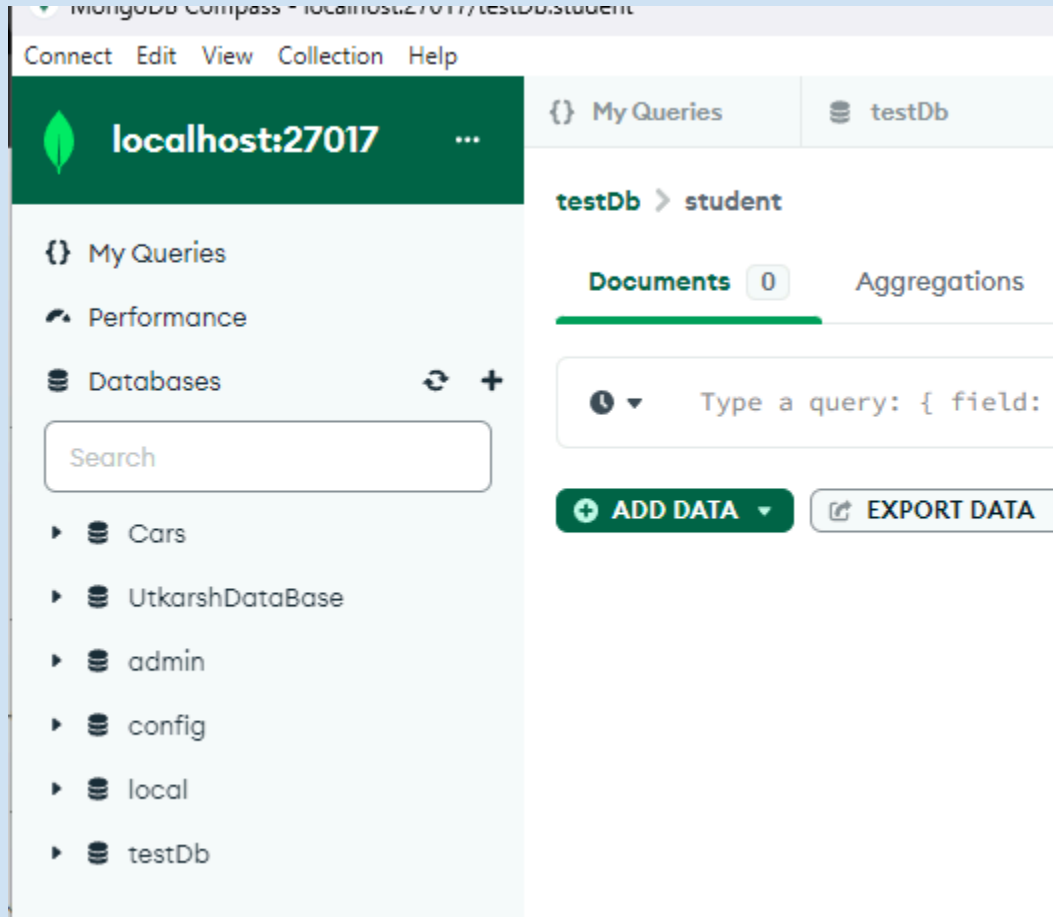
Name -Utkarsh Raj

Roll -22CS2024

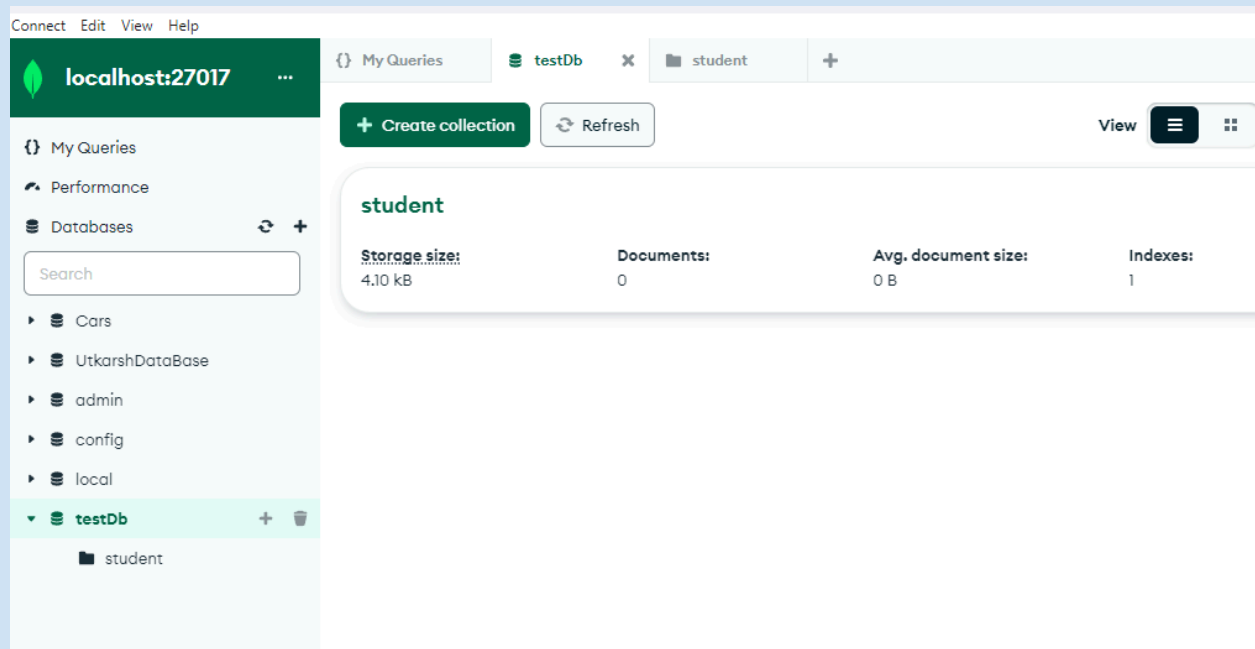
1. Connect to a MongoDB server using MongoDB Compass.



2. Create a new database named "testdb" in MongoDB Compass.



3. Create a new collection named "students" in the "testdb" database.



4. Insert ten documents into the "students" collection with the following fields: name, age, and email.



```
_id: ObjectId('660beabe5063e5ffaac1a544')
name : "Eve"
age : 23
email : "eve@example.com"
```

```
_id: ObjectId('660beabe5063e5ffaac1a545')
name : "Frank"
age : 20
email : "frank@example.com"
```



```
_id: ObjectId('660beabe5063e5ffaac1a546')
name : "Grace"
age : 22
email : "grace@example.com"
```

```
_id: ObjectId('660beabe5063e5ffaac1a547')
name : "Harry"
age : 21
```

```
_id: ObjectId('660beabe5063e5ffaac1a547')
name : "Harry"
age : 21
email : "harry@example.com"
```



```
_id: ObjectId('660beabe5063e5ffaac1a548')
name : "Ivy"
age : 19
email : "ivy@example.com"
```

```
_id: ObjectId('660beabe5063e5ffaac1a549')
name : "Jack"
age : 23
email : "jack@example.com"
```

5. View the contents of the "students" collection.

```
const student = require('../model/student');
const Todo =require('../model/student');

exports.getStudent = async(req,res)=>{
  try{
```

```

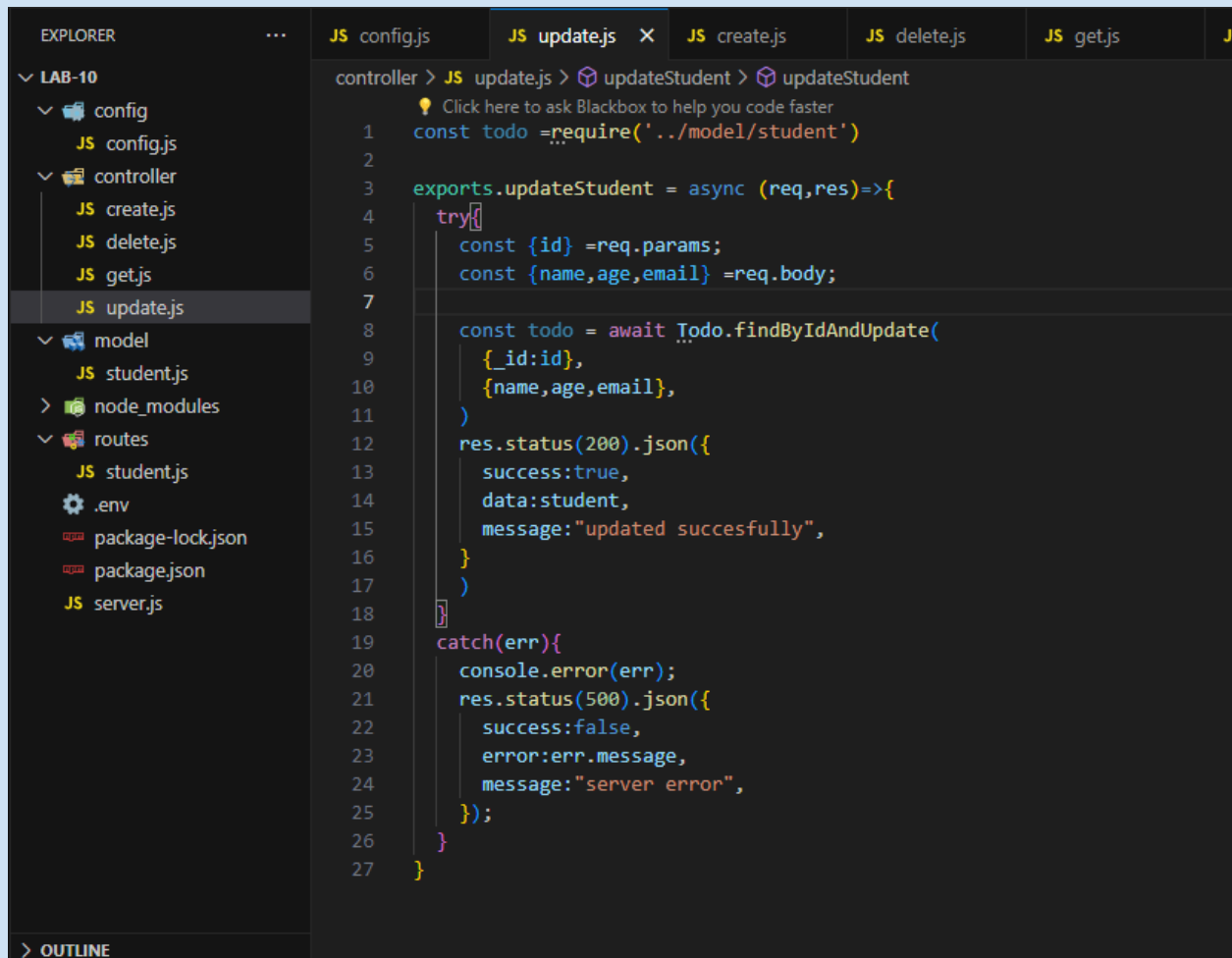
    const student= await Student.find({});
    res.status(200).json({
      success:true,
      data :student,
      message:"entire todo data is fetched",
    })
  }
  catch(err){
    console.error(err);
    res.status(500).json({
      success:false,
      error:err.message,
      message:"server error",
    });
  }
};

exports.getStudentById = async(req,res)=>{
  try{
    const id =req.params.id;
    const student = await Student.findById({_id:id});
    if(!student){
      res.status(404).json({
        success:false,
        message:"No data found at given id",
      });
    }
    res.status(200).json({
      success:true,
      data:student,
      message:`student ${id} data succesfully fetched`,
    })
  }
  catch(err){
    console.error(err);
    res.status(500).json({
      success:false,
      error:err.message,
      message:"server error",
    });
  }
};

```

```
    })  
  }  
};
```

6. Update the age of a specific student in the "students" collection.



The screenshot shows a VS Code editor with a project named 'LAB-10'. The Explorer sidebar on the left shows the file structure: `config` (containing `config.js`), `controller` (containing `create.js`, `delete.js`, `get.js`, and `update.js`), `model` (containing `student.js`), `node_modules`, `routes` (containing `student.js`), `.env`, `package-lock.json`, `package.json`, and `server.js`. The main editor window displays the `update.js` file in the `controller` directory. The code defines an `updateStudent` function using Express.js middleware. It requires the `student` model, uses `async` and `await` to find and update a student's age, and returns a JSON response with the updated student data or an error message. The function is wrapped in a `try-catch` block to handle potential errors.

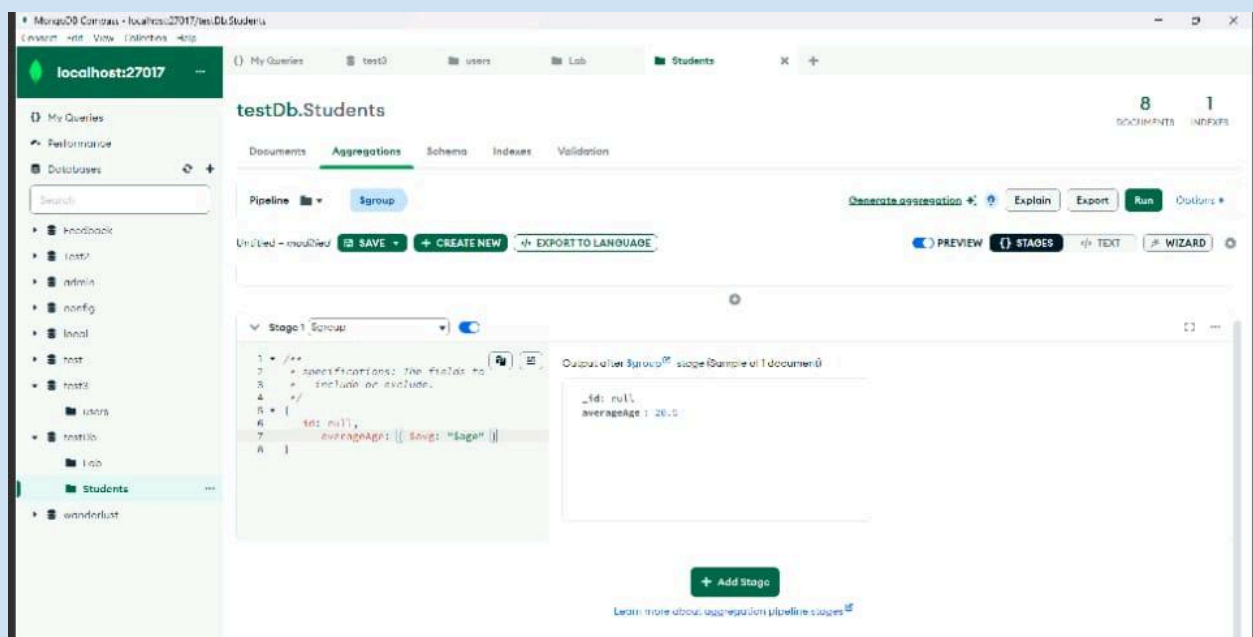
```
controller > JS update.js > updateStudent > updateStudent  
Click here to ask Blackbox to help you code faster  
1  const todo = require('../model/student')  
2  
3  exports.updateStudent = async (req,res)=>{  
4    try{  
5      const {id} =req.params;  
6      const {name,age,email} =req.body;  
7  
8      const todo = await Todo.findByIdAndUpdate(  
9        {_id:id},  
10       {name,age,email},  
11      )  
12      res.status(200).json({  
13        success:true,  
14        data:student,  
15        message:"updated succesfully",  
16      })  
17    }  
18  }  
19  catch(err){  
20    console.error(err);  
21    res.status(500).json({  
22      success:false,  
23      error:err.message,  
24      message:"server error",  
25    });  
26  }  
27 }
```

7. Delete a document from the "students" collection based on a specific condition.

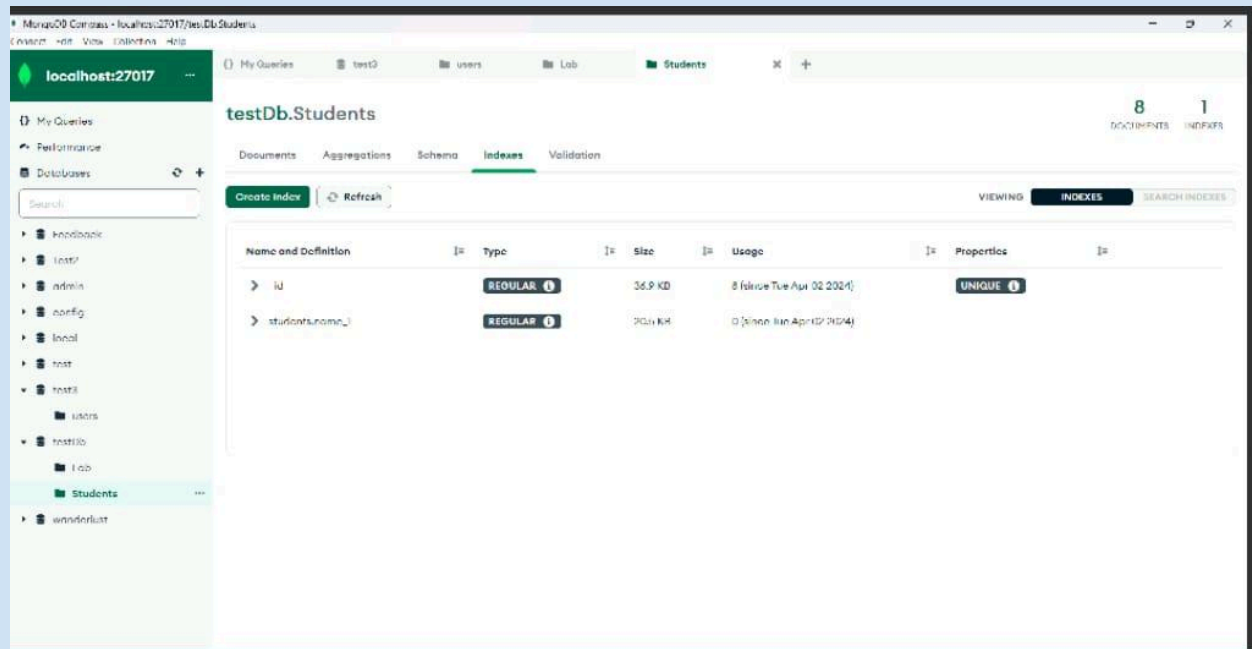
The screenshot shows a VS Code editor with a project named 'LAB-10'. The Explorer sidebar on the left shows a file structure with folders for 'config', 'controller', 'model', 'node_modules', and 'routes'. The 'controller' folder is expanded, showing files like 'config.js', 'create.js', 'delete.js', 'get.js', 'update.js', 'student.js', and 'server.js'. The 'delete.js' file is selected and its content is displayed in the main editor. The code is a Node.js script that uses Express.js to handle a DELETE request. It imports 'require' and 'Student' from local modules. The 'deleteStudent' function is an asynchronous function that takes 'req' and 'res' as arguments. It uses a 'try-catch' block to handle the deletion logic. Inside the 'try' block, it extracts the 'id' from the request parameters, finds the student by ID, and if found, returns a 200 status with a success message. If not found, it returns a 404 status with a message. The 'catch' block handles any errors by logging them and returning a 500 status with an error message.

```
controller > JS delete.js > deleteStudent > deleteStudent > student
Click here to ask Blackbox to help you code faster
1 const student = require('../model/student');
2
3 exports.deleteStudent = async (req,res)=>{
4   try{
5     const{id}=req.params;
6     const student=await Student.findByIdAndDelete(id);
7     if(!student){
8       res.status(404).json({
9         success:false,
10        message:"No data found at given id",
11      });
12    }
13    res.status(200).json({
14      success:true,
15      data:student,
16      message:"Delete succesfully",
17    })
18  }
19  catch(err){
20    console.error(err);
21    res.status(500).json({
22      success:false,
23      error:err.message,
24      message:"server error",
25    });
26  }
27 }
28 }
```

8. Use the aggregation pipeline to calculate the average age of all students in the "students" collection.



9. Create an index on the "name" field in the "students" collection.



```
const mongoose = require( 'mongoose' );

require("dotenv").config();

const dbConnect = ()=>{
  mongoose.connect(process.env.testDb,{
    useNewUrlParser :true,
    useUnifiedTopology: true,
  })
  .then(()=>{console.log( "Database Connected Successfully" )})
  .catch((error)=>{
    console.log( "Database Connection Failed!" );
    console.error(error.message);
    process.exit(1);
  });
}

module.exports = dbConnect;
const express = require('express');
const router = express.Router();
```

```

const {create}= require('../controller/create');
const{get,getStudentById} =require("../controller/get");
const{updateStudent} = require("../controller/update");
const{deleteStudent} =require("../controller/delete");
const {createStudent } = require('../controller/create');
const { getStudent } = require('../controller/get');

router.post("/createStudent",createStudent);
router.get("/getStudent",getStudent);
router.get("/getStudent/:id",getStudentById);
router.put('/updateStudent/:id',updateStudent);
router.delete("/deleteStudent/:id",deleteStudent);

module.exports =router;

```

```

const express =require('express');
const app=express();
require('dotenv').config();

const PORT = process.env.PORT || 4000;
app.use(express.json());

const studentRoutes = require('../routes/student');

app.use("/api/v1",studentRoutes)
app.listen(PORT,()=>{console.log(`server listen at ${PORT} `)});

const dbConnect = require('../config/config') ;
dbConnect();

app.get("/",(req,res)=>{
  res.send("<h1>this is home page </h1>")
});

```