CSSE3101 – ADVANCED WEB TECHNOLOGIES
MERN STACK DEVELOPMENT

Objective:

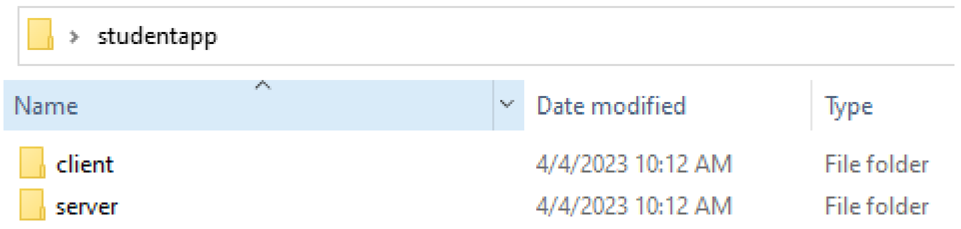
In this lab activity, you are going to learn how to set up an MERN stack React application. You will learn to create a Mongodb database and collection inside the database and make the user interface(UI) to collect the data to insert into the collection in the database.

Requirements implemented in this STUDENT INFORMATION APPLICATION:

```
PART 1 – APPLICATION SETUP
PART 2 – FRONT- END DEVELOPMENT USING REACT AND BOOTSTRAP
PART 3 – CREATING THE DATABASE USING MONGODB
PART 4 – SERVER-SIDE DEVELOPMENT USING NodeJS and Express
PART 5 – CONNECTING YOUR APPLICATION TO MONGODB
PART 6 – CREATING MODELS
PART 7 – CREATING EXPRESS ROUTES (CRUD Operations - CREATE/Post – add
student record to the database
```

PART 1 – APPLICATION SETUP

1. Create the application folder structure.



```
> studentapp

Name                 Date modified        Type
client               4/4/2023 10:12 AM    File folder
server               4/4/2023 10:12 AM    File folder
```

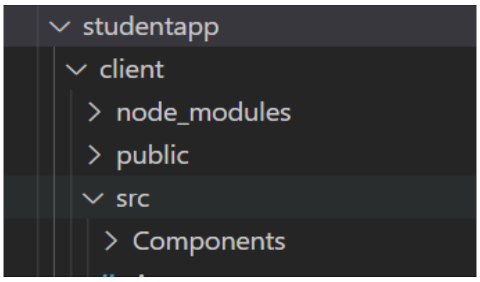
2. Open the **studentapp** folder in Visual Studio (VS) Code. Open a new terminal and do the following:
   a. Inside the folder, create a folder named server.
   b. Inside the folder, create a react app using the command:

      npx create-react-app client

   c. Install the required dependencies for this application using the command **npm install**. Other dependencies will be added later.
      npm install bootstrap
      npm install react-router-dom
   d. Start your client server using the command **npm start.**

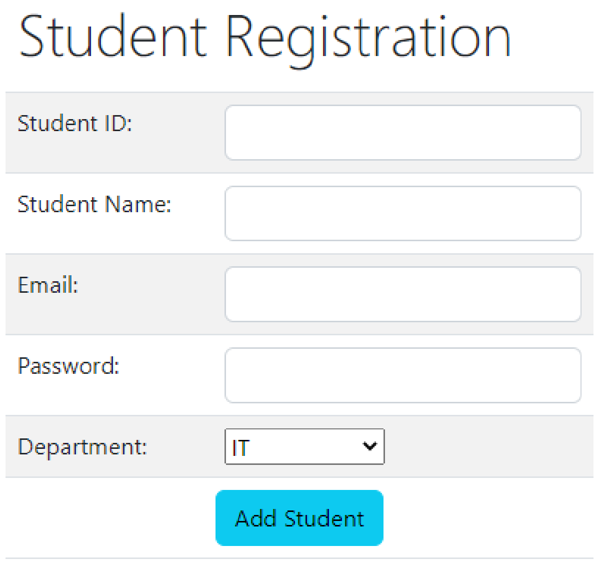
PART 2 – FRONT END DEVELOPMENT USING REACT AND BOOTSTRAP

1. Create a new folder named **Components** in the src folder.



```
studentapp
  client
    node_modules
    public
    src
      Components
```

2. In the Components folder, create a new file: **StudentRegister.js.** This component will allow the user to add or register a new student and will be saved to the database.
3. Create the student registration form as shown below. Use Bootstrap classes and define your own CSS to achieve the desired UI. Use Bootstrap table to create the form layout.



Student Registration

Student ID:	
Student Name:	
Email:	
Password:	
Department:	IT
	Add Student

4. Update your **App.js, so that, it will render the StudentRegister.**
5. Import the following file in `src/index.js`

```
import 'bootstrap/dist/css/bootstrap.css'
```

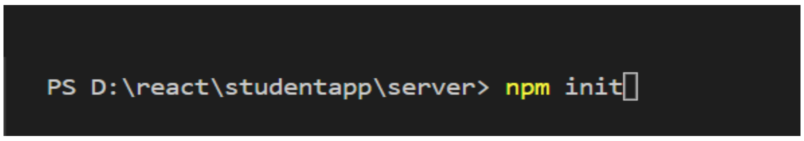
PART 3 – CREATING THE DATABASE USING MONGODB

Project Summary:

Project Name	STUDENT_INFORMATION_SYSTEM_APP
Cluster Name	studentinfosys
Database Name	studentdb
Collections	students
Database username	admin
Database password	csse3101

PART 4 – SERVER-SIDE DEVELOPMENT USING NodeJS and Express

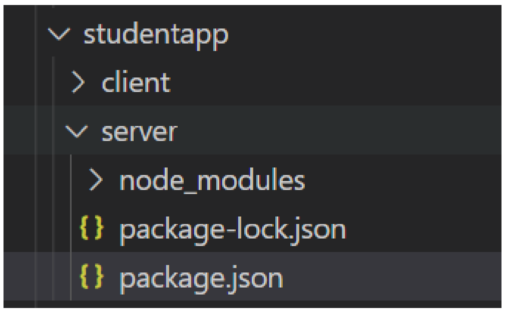
1. Go to the terminal, and in the server folder do the following:
   a. Execute the command to initialize the server. Accept the default server setting by Pressing the Enter key.
   npm init



```
PS D:\react\studentapp\server> npm init
```

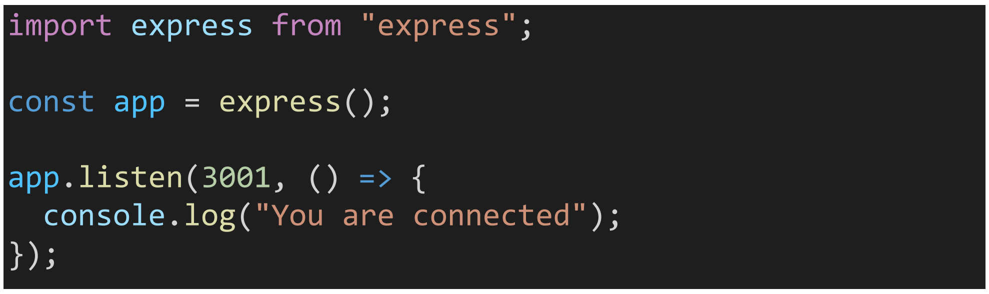
   b. Install the libraries.
   npm install express
   npm install mongoose

   After executing the commands your server folder will have the following files and folders:
2. In the server folder, create a file **index.js.** This file is the entry point for your server. Do the



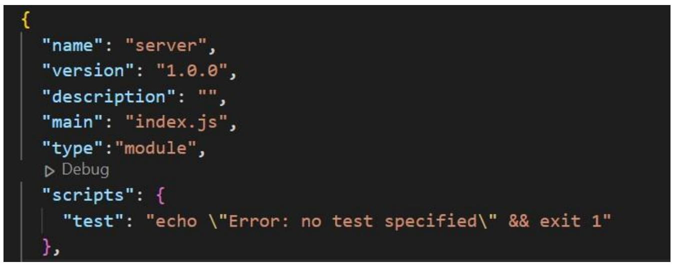
```
∨ studentapp
  > client
  ∨ server
    > node_modules
    {} package-lock.json
    {} package.json
```

   following:
   a. Write the code to create the express server.



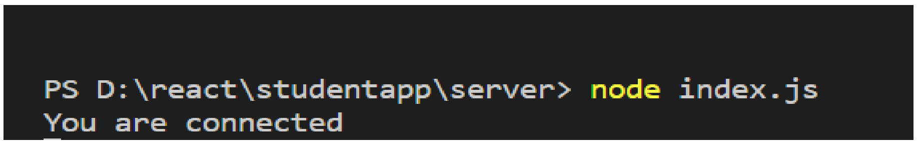
```
import express from "express";

const app = express();

app.listen(3001, () => {
  console.log("You are connected");
});
```

   b. Edit `package.json` and add the following after the main:`"type":"module",`



```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type":"module",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
```

3. From the terminal, run your server.

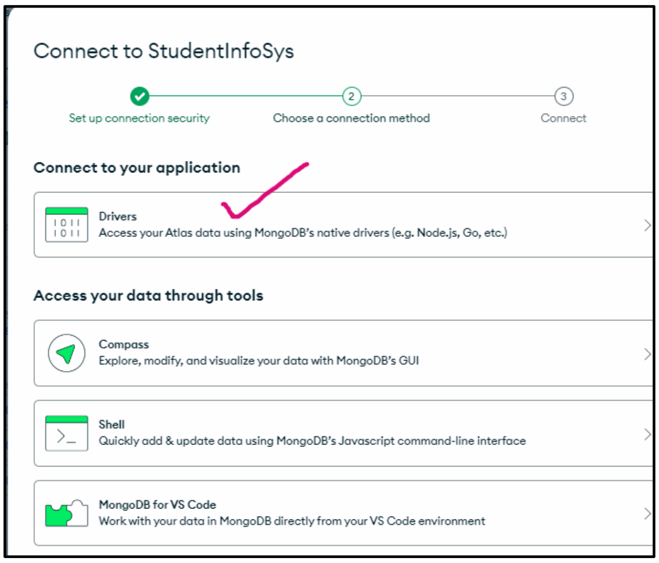


```
PS D:\react\studentapp\server> node index.js
You are connected
```

   Note: Every time you have changes in your index.js, you need to run again the server.

PART 5 – CONNECTING YOUR APPLICATION TO MONGODB

1. Login to your MongoDb Atlas account. Go to the specific project. Select Connect Your Application.



Connect to StudentInfoSys

Set up connection security — Choose a connection method — Connect

Connect to your application

Drivers
Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)

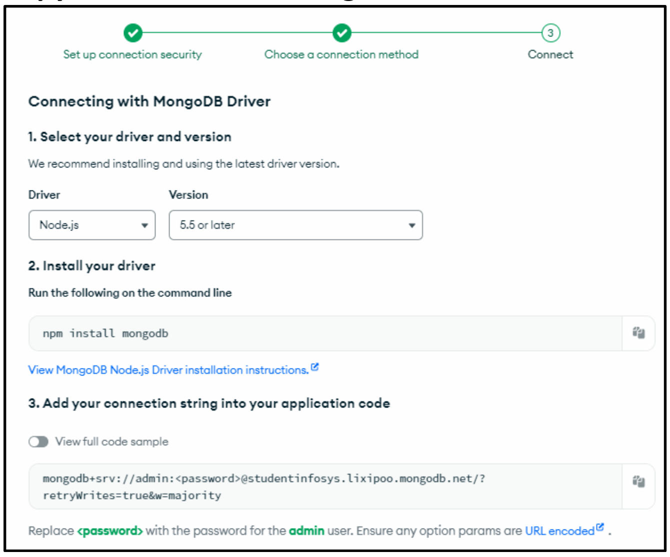
Access your data through tools

Compass
Explore, modify, and visualize your data with MongoDB's GUI

Shell
Quickly add & update data using MongoDB's Javascript command-line interface

MongoDB for VS Code
Work with your data in MongoDB directly from your VS Code environment

2. **Copy the connection string.**



Set up connection security — Choose a connection method — Connect

Connecting with MongoDB Driver

1. Select your driver and version

We recommend installing and using the latest driver version.

Driver: Node.js
Version: 5.5 or later

2. Install your driver

Run the following on the command line

```
npm install mongodb
```

View MongoDB Node.js Driver installation instructions.

3. Add your connection string into your application code

View full code sample

```
mongodb+srv://admin:<password>@studentinfosys.lixipoo.mongodb.net/?
retryWrites=true&w=majority
```

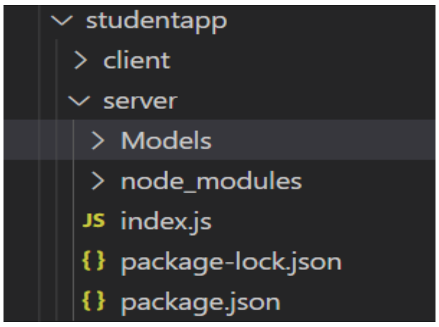
Replace **<password>** with the password for the **admin** user. Ensure any option params are URL encoded .

3. In **index.js**, do the following:
   a. Import mongoose library.
   b. Create a connectionString variable and assign the connection string generated from MongoDB Atlas.
   c. Here is an example of a connection string (specify values for password and name of the database):
   mongodb+srv://admin:<password>@studentinfosysapp.rr1cedt.mongodb.net/<name of the database>?retryWrites=true&w=majority
   d. Complete the connection code as shown below:

```
import express from "express"; import
mongoose from "mongoose";

const app = express();

//Database connectionconst
connectString =
  "mongodb+srv://admin:admin12345@studentinfosysapp.5pqj5zu.mo
ngodb.net/studentDb?retryWrites=true&w=majority";

mongoose.connect(connectString);

app.listen(3001, () => { console.log("You are
  connected");
});
```

PART 6 – CREATING MODELS

1. In the server folder, create the folder **Models**. This is where the database model is saved.



```
studentapp
  client
  server
    Models
    node_modules
    index.js
    package-lock.json
    package.json
```

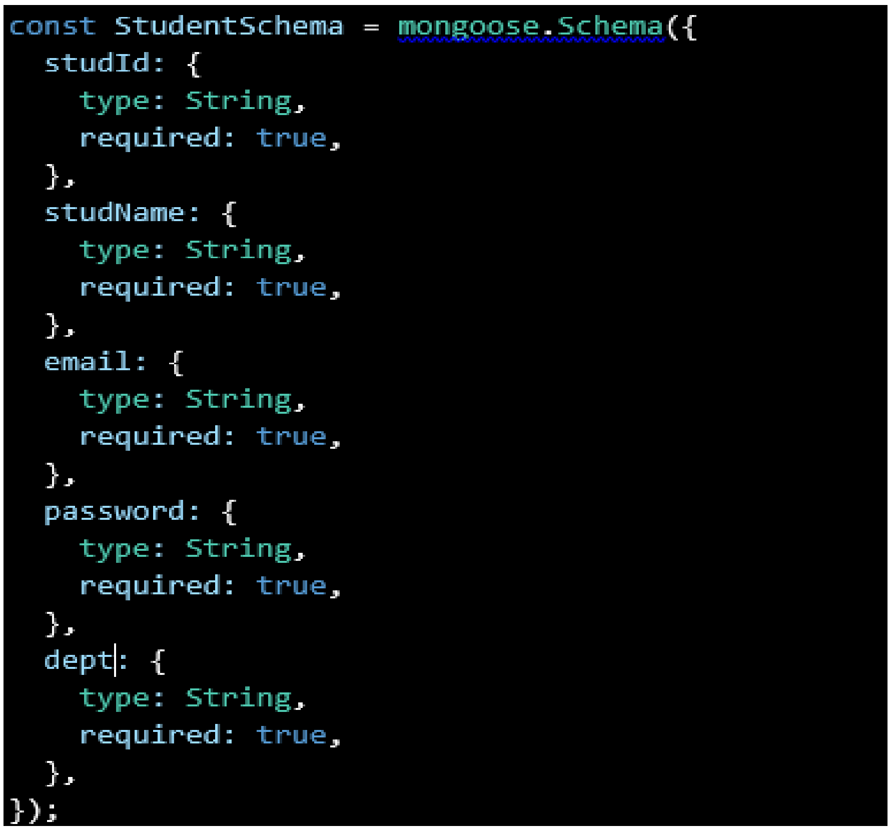
2. Create a model that will save the student information. Create a new file: **Student.js** inside the Models folder and do the following:
   a. Import mongoose.

   ```
   import mongoose from "mongoose";
   ```

   b. Create a schema.

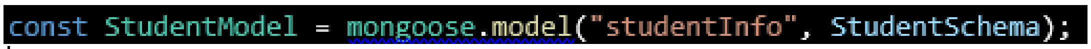
   ```
   const StudentSchema = mongoose.Schema({});
   ```

c. Add the fields in the schema.



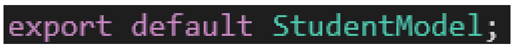
```
const StudentSchema = mongoose.Schema({
  studId: {
    type: String,
    required: true,
  },
  studName: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
  },
  password: {
    type: String,
    required: true,
  },
  dept: {
    type: String,
    required: true,
  },
});
```

d. Create a model variable and bind the MongoDb collection with the schema.
Syntax:
`mongoose.model(<Collectionname>,  <CollectionSchema>)`



```
const StudentModel = mongoose.model("studentInfo", StudentSchema);
```

e. Export the model.



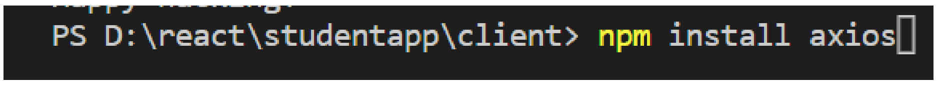
```
export default StudentModel;
```

PART 7– CREATING EXPRESS ROUTES

1. In the client folder, do the following:

   a. Go to the terminal and install axios.

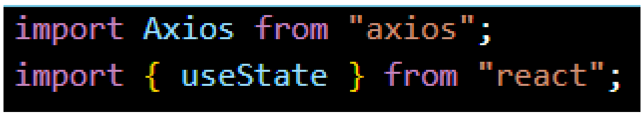
```
npm install axios
```



```
PS D:\react\studentapp\client> npm install axios
```

   b. Update the **StudentRegister.js** and do the following:

      i. Import the following:



```
import Axios from "axios";
import { useState } from "react";
```

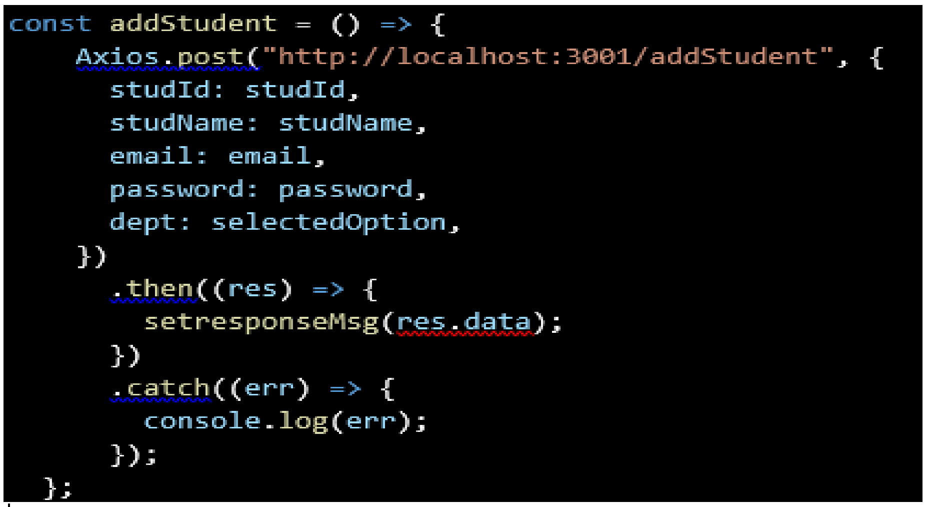
      ii. Create state variables using useState hook for each of the field in the form.

      iii. Update each state variable to be assigned with the value of the respectiveform elements by adding an event handler for each of the form controls.

      iv. Create a function **addStudent** using arrow function syntax. This function uses Axios to submit a request to the server passing along the data to theserver.

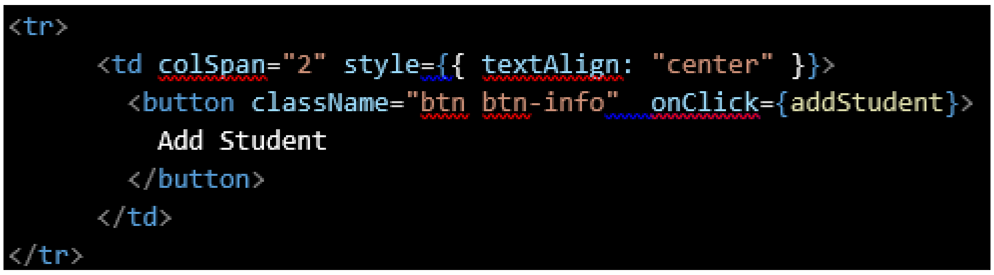
Syntax:

```
 const nameofmethod = ()=>{
     Axios.post("Server route that handles
therequest",{
          JSON format data as part of the request
body
     })
      .then(()=>{
          Action if successful
     })
     .catch(()=>{
          Action if error is encountered
     })
}
```



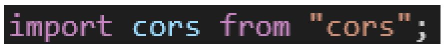
```
const addStudent = () => {
    Axios.post("http://localhost:3001/addStudent", {
      studId: studId,
      studName: studName,
      email: email,
      password: password,
      dept: selectedOption,
    })
      .then((res) => {
        setresponseMsg(res.data);
      })
      .catch((err) => {
        console.log(err);
      });
  };
```

v. Create the event handler for the button to call the **addStudent** function.

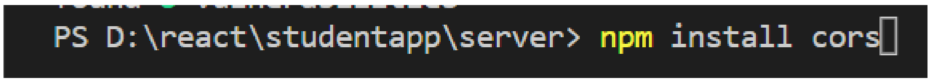


```
<tr>
      <td colSpan="2" style={{ textAlign: "center" }}>
        <button className="btn btn-info"  onClick={addStudent}>
          Add Student
        </button>
      </td>
</tr>
```

2. Go to the terminal, and in the server install `cors`.
   npm install cors
3. In the server folder, update **index.js** by doing the following.
   a. Import cors.

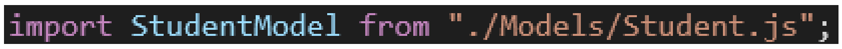


```
import cors from "cors";
```



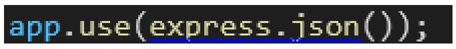
```
PS D:\react\studentapp\server> npm install cors
```

   b. Import the needed model.



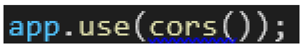
```
import StudentModel from "./Models/Student.js";
```

   c. Mount the needed middleware functions.
      i. Parse the incoming requests with JSON payloads



```
app.use(express.json());
```

      ii. Enable Cross-Origin Resource Sharing (CORS)



```
app.use(cors());
```

   d. Create the Express POST route for adding new student.

BACK-END (server folder)

index.js

```
import express from "express";
import mongoose from "mongoose";
import StudentModel from "./Models/Student.js";
import cors from "cors";

//declare an express object
const app = express();

//middleware
app.use(cors());
app.use(express.json());

//database connection
const connectString =
  "mongodb+srv://admin:b606110@studentinfocluster.0hmagz7.mongodb.net/studentDb?ret
ryWrites=true&w=majority";

mongoose.connect(connectString);

//express POST route for adding new student
app.post("/addStudent", async (req, res) => {

  const student = new StudentModel({
    studId: req.body.studId,
    studName: req.body.studName,
    email: req.body.email,
    password: req.body.password,
    department: req.body.department,
  });

  await student.save();
  res.send("Record Successfully Added!");
});
```

Required Submission.

Once you complete the lab activity, you are required to upload the database model file `student.js`, `server` folder and `src` folder of the client app.