FULL STACK DEVELOPMENT - E-FILE

Name : Shahid Firozabad
Roll No : 2305029
Class : TY - BCA
Batch : 02
Subject : Full Stack Development

CONTENT

	• Raw Queries	
10	• Sequelize Association (1:1, 1:M) • Advanced M:N Associations	07
11	• Installation of React js • Components (Build-in and Custom) • Props • States	07
12	• Hooks (useState, useReducer, useContext, useRef, useEffect, useMemo, useCallback, etc.)	07
13	• Routes in React JS • Navigation	07
14	• Redux • Dispatch	07
15	• Integrate Node with React JS	07

WEEK 01

1) URL Parameter Extractor: Parsing Query Date.

Code

```
const { URL } = require('url');

const myUrl = new URL('http://www.google.com/?name=shahid');

console.log('Host:', myUrl.host);

console.log('Name:', myUrl.searchParams.get('name'));
```

Output

```
PS C:\Users\TYBCA\Desktop\2305029> node url
Host: www.google.com
Name: shahid
PS C:\Users\TYBCA\Desktop\2305029>
```

2) Simple HTTP Server: Basic Node.js Server Setup

code

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

var server = http.createServer(function(req,res){
})
server.listen(5000);
console.log("Server is on : 5000");
```

Output

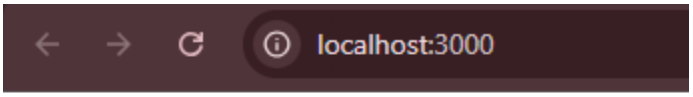
```
Name: shahid
PS C:\Users\TYBCA\Desktop\2305029> node server1
Server is on : 5000
```

3) Express Hello World: Simple Web Server Setup

Code

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

const app = express();

const port = 3000;

app.get("/", (req,res) => {

    res.send("Hello World");

})

app.listen(port, () => {

    console.log("App is listening on port: ", port);

    console.log("Visit: http://localhost:3000");

})
```

Output



Hello World

4) Multi-Page HTTP Server: Dynamic Page Navigation

Code

```
var http = require('http');

var server = http.createServer(function (req, res) {

    const Pages = (title, content) => {

        return `

        <html>

        <body>

         </h1>${title}</h1>

         <h2>${content}</h2>

         <button onClick="location.href='/'">Home</button>

          <button
onclick="location.href='/student'">Student</button>

          <button onclick="location.href='/admin'">Admin</button>

    </body>

              </html>       `

    }

    if (req.url == "/") {

        res.writeHead(200, { 'Content-Type': 'text/html' });

        res.write(Pages('HomePage', 'Home page here'));

        res.end();

    }
```

```
    else if (req.url == '/student') {

        res.writeHead(200, { 'Content-Type': 'text/html' });

        res.write(Pages('Student', 'Student page here'));

        res.end();

    }

    else if (req.url == "/admin") {

        res.writeHead(200, { 'Content-Type': 'text/html' });

        res.write(Pages('Admin', 'Admin page here'));

        res.end();

    }

    else {

        res.end('Invalid Request');

    }

});

server.listen(5000);

console.log("Server is running on port : 5000");

console.log("Go to http://localhost:5000");
```

Output

WEEK 02

5) File Path Resolver: Joining Paths in Node.js

Code

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

const filePath = '2305029/message.txt';

const fullPath = path.join(__dirname,filePath);

console.log('Full path', fullPath);
```

Output

```
PS C:\Users\TYBCA\Desktop\2305029> node path
Full path C:\Users\TYBCA\Desktop\2305029\2305029\message.txt
PS C:\Users\TYBCA\Desktop\2305029>
```

7) System Resource Checker: Free Memory & Platform Info

Code

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

const freeMemBytes = os.freemem();

const freeMemGb = (freeMemBytes / (1024 ** 3)).toFixed(2);

console.log('Platform: ', os.platform());

console.log('Free Memory: ', freeMemGb, "TB");
```

Output

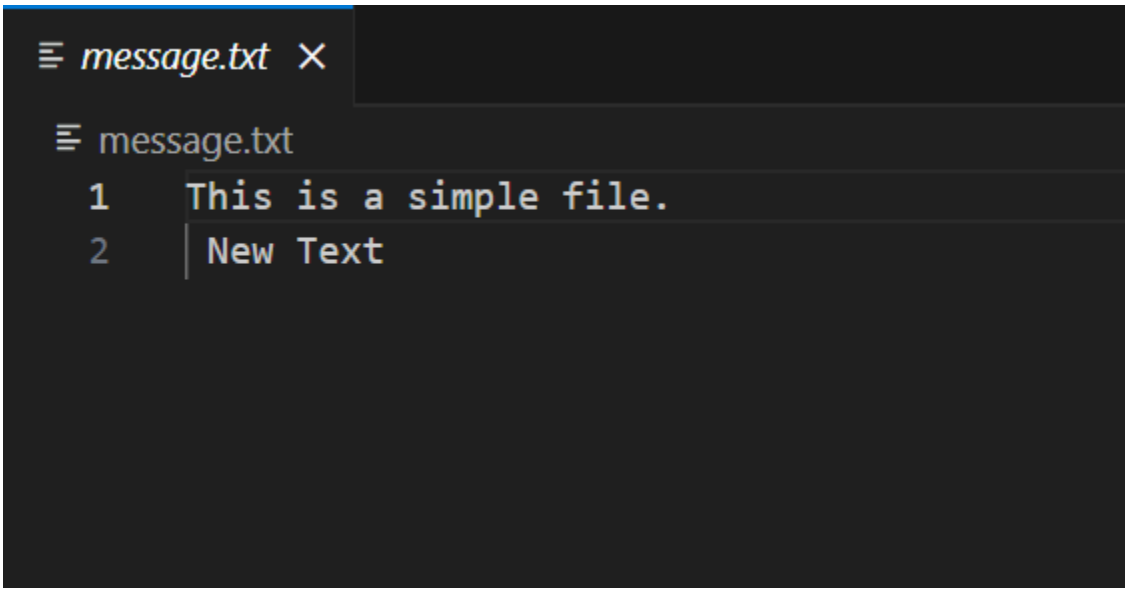
```
PS C:\Users\TYBCA\Desktop\2305029> node os
Platform:  win32
Free Memory:  1.53 TB
PS C:\Users\TYBCA\Desktop\2305029>
```

8) File Handler: Write and Read Operations in Node.js

Code

```
const fs = require("fs");

fs.writeFileSync('message.txt', 'This is a simple file.\n New Text');

const data = fs.readFileSync('message.txt', 'utf-8');

console.log(data);
```

Output



9) Math Operations Module: Basic Arithmetic Functions

Code

[math.js]

```
const add = (a, b) => {

  return a + b;

}

const sub = (a, b) => {

    return a - b;

}

const mul = (a, b) => {

    return a * b;

}
```

```
const div = (a, b) => {

    if (b === 0 || a === 0) {

        return "Cannot divide by zero.";

    }

    else {

        return a / b;

    }

}

module.exports.math = {

    add,

    sub,

    mul,

    div

}
```

main.js

```
// Math -----

console.log("\n","-----------------Math----------------------","\n")

const Math = require('./math');

console.log("Addition: ", Math.math.add(10,20));

console.log("Subtract: ", Math.math.sub(10,20));

console.log("Multiply: ", Math.math.mul(10,20));

console.log("Divide: ", Math.math.div(10,20));
```

Output

```
  -----------------Math---------------------

Addition:   30
Subtract:   -10
Multiply:   200
Divide:   0.5

  -----------------Date----------------
```

10) Log System with Server Output: Info, Warning, and Error Logs

Code

logs.js

```
var log = {

   info: function (info) {

      const message = "Info : " + info;

      console.log('info: ' + info);

      return message;

   },

   warning: function (warning) {

      const message = "Warning : " + warning;

      console.log('Warning' + warning);

      return message;

   },
```

```
   error: function (error) {

      const message = "Error" + error;

      console.log('Error' + error);

      return message;

   },

}

module.exports.log = log;
```

main.js

```
const { log } = require('./logs');

console.log();

console.log("--------------------Logs on Browser----------------------");

console.log();

var http = require('http');

var server = http.createServer(function (req, res) {

    const LogsInfo = () => {

        return log.info("Starting Node Js.........");

    };

    const LogsWarning = () => {

        return log.warning("Warning : ! Battery Low ! ");

    };

    const LogsError = () => {

        return log.error("Error : ! Server Crashed !");

    };
```

```
    const LogsText1 = LogsInfo();

    const LogsText2 = LogsWarning();

    const LogsText3 = LogsError();

    res.writeHead(200, { "Content-Type": "text/html" });

    res.write(`<html><body><h3>${LogsText1}</h3></body></html>`);

    res.write(`<html><body><h3>${LogsText2}</h3></body></html>`);

    res.write(`<html><body><h3>${LogsText3}</h3></body></html>`);

    res.end();

});

server.listen(5000);

console.log("Server is running on port : 5000");

console.log("Go to http://localhost:5000");
```

Output

11) Greeting Module: Personalized Hello Function

Code

greet.js

```
function sayHello(name) {

   return `Hello, ${name}`;

}

module.exports.greet = sayHello ;
```

main.js

```
// Greet

console.log("\n","--------------------Greet----------------------","\n");

const {greet} = require('./greet');

const message = greet("Shahid");

console.log(message);
```

Output

```
 --------------------Greet----------------------

Hello, Shahid
```

12) Date and Time Module: Get Current Date and Time

Code

date.js

```
const GetCurrentDateTime = () => {

    const now = new Date();

    return now.toLocaleString();

}

module.exports = GetCurrentDateTime;
```

main.js

```
//  DATE ...........

console.log("\n","-----------------Date----------------","\n")

const GetCurrentDateTime = require('./date');

console.log("Date:", GetCurrentDateTime());
```

Output

```
  -----------------Date----------------

 Date: 1/8/2025, 9:46:23 am
```

WEEK 03

13) Personalized Greeting API: Greet by Name

Code

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

const app = express();

app.get("/", (req,res) => {

    const name = req.query.name || 'Guest';

    res.send(`Hello ${name}`);

})

app.listen(3000, () => {

   console.log("App is listening on port: ", 3000);

    console.log("Visit: http://localhost:3000");

})
```

Output

Hello Guest

14) Time Out

Code

```
console.log("Started");

setTimeout(() => {
    console.log("Timeout Callback");
}, 4000);

console.log("Ended");
```

Output

```
PS C:\Users\TYBCA\Desktop\2305029> node setTimeout
Started
Ended
Timeout Callback
PS C:\Users\TYBCA\Desktop\2305029>
```

15) Synchronous

Code

```
//  Operation Executes Sequentially
const fs = require('fs');
const data = fs.readFileSync('file.txt', 'utf-8');
console.log(data);
console.log("Finished Reading File.");
```

Output

```
PS C:\Users\TYBCA\Desktop\2305029> node synchronous
Hiiiii
how are you
?

i am shahid
Finished Reading File.
PS C:\Users\TYBCA\Desktop\2305029>
```

16) Asynchronous

Code

```
const fs = require('fs');
fs.readFileSync('file.txt', 'utf-8', (err,data) => {
    if(err) throw err;
    console.log(data);
});

console.log("Started Reading File");
```

Output

```
PS C:\Users\TYBCA\Desktop\2305029> node asynchronous
Started Reading File
PS C:\Users\TYBCA\Desktop\2305029>
```

17) Concurrency

Code

```
const fs = require('fs');
console.log('Started Reading File');
fs.readFile("file.txt", 'utf-8', (err,data) => {
    if(err) throw err;
    console.log("File content,", "\n" , data);
})
console.log("This prints before file read completely.");
```

Output

```
PS C:\Users\TYBCA\Desktop\2305029> node concurrency
Started Reading File
This prints before file read completely.
File content,
 Hiiiii
how are you
?

i am shahid
PS C:\Users\TYBCA\Desktop\2305029>
```

18) Exception Handling

Code

```
function DivideNumbers(a,b) {
    try {
        if(typeof a !== 'number' || typeof b !== 'number'){
          throw new Error("Invalid input: Both must be numbers");
        }
        if(b === 0){
            throw new Error("Divison by zero ?, * NOT POSSIBLE *");
        }
        let result = a/b;
        console.log("Result: ", result);
    }catch(error){
        console.log("Error Occured: ", error.message);
    }finally {
        console.log("Execution of DivideNUmbers() is complete.\n");
    }
}

//  Calling of function | Test Cases.
DivideNumbers(10,20);
DivideNumbers(0,20);
DivideNumbers(10,0);
DivideNumbers(100,50);
```

Output

```
PS C:\Users\TYBCA\Desktop\2305029> node exception
Result:  0.5
Execution of DivideNUmbers() is complete.

Result:  0
Execution of DivideNUmbers() is complete.

Error Occured:  Divison by zero ?, * NOT POSSIBLE *
Execution of DivideNUmbers() is complete.

Result:  2
Execution of DivideNUmbers() is complete.

PS C:\Users\TYBCA\Desktop\2305029>
```

19) Promises

Code

```
function fullName(callback) {
    console.log("Shahid");
    callback();
}
function details() {
    console.log("Full Stack Developer");
}

fullName(details);
```

Output

```
PS C:\Users\TYBCA\Desktop\2305029> node promises
Shahid
Full Stack Developer
PS C:\Users\TYBCA\Desktop\2305029>
```

20) Multiple Event

Code

```
const EventEmitter = require("events");
const colors = require('colors');
const myEmitter = new EventEmitter();

myEmitter.on('start', (data) => {
    console.log();
    console.log('1 -->'.yellow,'Listener 01 triggered for "start": ', data);
    myEmitter.emit('process', `Processed: ${data}`);
});
myEmitter.on('start', (data) => {
    console.log();
    console.log('2 -->'.yellow,'Listener 02 triggered for "start":Logging audit:
', data);
});

myEmitter.on('process', (info) => {
    console.log();
    console.log('3 -->'.yellow,'Processing Complete: ', info);
    myEmitter.emit('end', 'All Tasks Done !');
});

myEmitter.on('end', (msg) => {
    console.log();
    console.log('4 -->'.yellow,'End event recieved: ', msg);
});

myEmitter.emit('start', 'Data Recieved from user Input');
console.log();
```

Output

```
PS C:\Users\TYBCA\Desktop\2305029> node MultiEvnt

 1 --> Listener 01 triggered for "start":  Data Recieved from user Input

 3 --> Processing Complete:  Processed: Data Recieved from user Input

 4 --> End event recieved:  All Tasks Done !

 2 --> Listener 02 triggered for "start":Logging audit:  Data Recieved from user Input

PS C:\Users\TYBCA\Desktop\2305029>
```

21) Argument Event

Code

```
const EventEmitter = require('events');
const { emit } = require('process');
const emitter = new EventEmitter();

emitter.on('userJoined', (username, userID) => {
    console.log(`${username}, (${userID}) has joined the chat`);
});

emitter.emit('userJoined', 'Shahid', 29);
```

Output

```
PS C:\Users\TYBCA\Desktop\2305029> node argEvent
Shahid, (29) has joined the chat
PS C:\Users\TYBCA\Desktop\2305029>
```

22) Error Handling

Code

```
const EventEmitter = require('events')
const emitter = new EventEmitter();
emitter.on('error', (err) => {
    console.log();
    console.error('An err occured', err.message);
    console.log();

})
emitter.emit('error', new Error('Something went Wrong'));
```

Output

```
PS C:\Users\TYBCA\Desktop\2305029> node ErrorHandle

An err occured Something went Wrong

PS C:\Users\TYBCA\Desktop\2305029>
```

23) Stimulate Asynchronous Using setInterval

Code

```
console.log("Start");

setTimeout(function() {
    console.log("Executed after 2 seconds.");
}, 2000);

console.log("End");

console.log("Print Name");

function greet(name) {
 console.log("My name is: ", name);
}

setTimeout(greet, 3000 , "Shahid");

console.log("Set Interval");

setInterval(function () {
    console.log("This prints every 2 seconds");
}, 2000);
console.log("Interval Functions Ends");

function display(name){
    console.log(name);
}
setInterval(display, 4000, "Shahid");
```

Output

```
PS C:\Users\TYBCA\Desktop\2305029> node setInterval
Start
End
Print Name
Set Interval
Interval Functions Ends
Executed after 2 seconds.
This prints every 2 seconds
My name is:  Shahid
Shahid
This prints every 2 seconds
This prints every 2 seconds
Shahid
This prints every 2 seconds
This prints every 2 seconds
Shahid
This prints every 2 seconds
This prints every 2 seconds
Shahid
This prints every 2 seconds
This prints every 2 seconds
Shahid
This prints every 2 seconds
This prints every 2 seconds
Shahid
This prints every 2 seconds
This prints every 2 seconds
Shahid
This prints every 2 seconds
This prints every 2 seconds
Shahid
This prints every 2 seconds
This prints every 2 seconds
```

24) Coloring the text in the console.

Code

```
const colors = require('colors');
colors.disable();
console.log("Red".red);
console.log("green".green);
console.log("blue".blue);
console.log("blue".blue.bold);
console.log("yellow".yellow.underline);
```

Output

```
PS C:\Users\TYBCA\Desktop\2305029> node colors
Red
green
blue
blue
yellow
PS C:\Users\TYBCA\Desktop\2305029>
```

24) Disable of the color function.

```
const colors = require('colors');
colors.disable();
console.log("Red".red);
console.log("green".green);
console.log("blue".blue);
console.log("blue".blue.bold);
console.log("yellow".yellow.underline);
```

Output

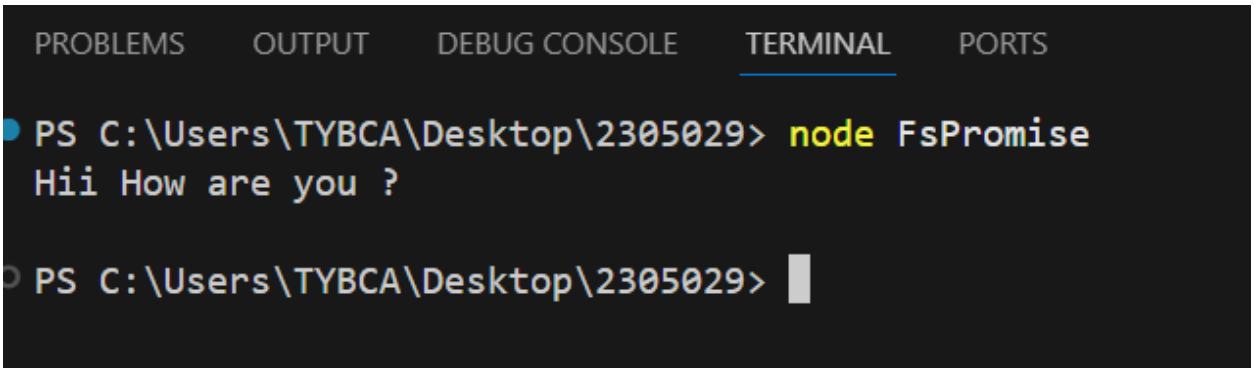
```
PS C:\Users\TYBCA\Desktop\2305029> node colors
Red
green
blue
blue
yellow
PS C:\Users\TYBCA\Desktop\2305029>
```

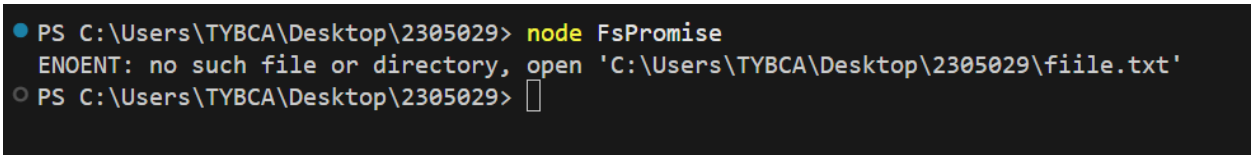
25) File System Reading Using promises.

Code FsPromise.js

```
var fs = require('fs');
var promise = new Promise(function(resolve, reject) {
    fs.readFile("file.txt", 'utf8', function(error, data) {
        if(error) {
            return reject(error);
        }
        resolve(data);
    });
});

promise.then(function(result) {
    console.log(result);
}, function(error) {
    console.error(error.message);
})
```

Output





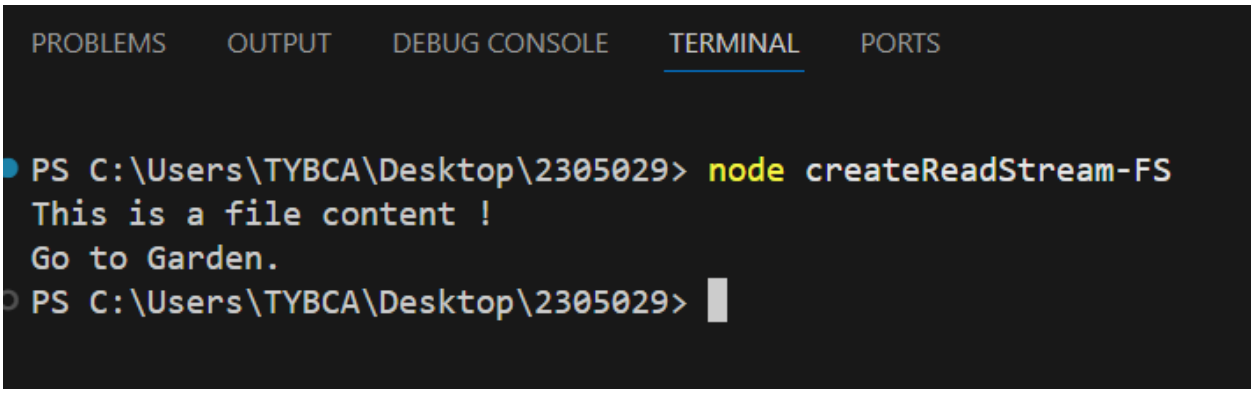
WEEK 04

25) Create ReadStream in File System.

Code

```
const fs = require('fs');
let streamin = fs.createReadStream('log.txt', 'utf8');
let filedata = '';

streamin.on('data', function(data) {
    filedata += data;
});

streamin.on('end', function() {
    console.log(filedata);
});

streamin.on('error', function(){
    console.log("Error Reading the file.");
});
```

Output



26) Create Write Stream in File System.

Code

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

const streamin = fs.createReadStream('log.txt', 'utf-8');
const streamout = fs.createWriteStream('NewLog.txt', 'utf-8');

streamin.pipe(streamout);

streamin.on('data', function(data) {
    console.log('transferred: %d' , data.length);
});

streamin.on('end', function() {
    console.log('Finished');
});

streamin.on('error', function() {
    console.log('Error Copying');
});
```

Output

```
PS C:\Users\TYBCA\Desktop\2305029> node createWriteStr
transferred: 39
Finished
PS C:\Users\TYBCA\Desktop\2305029>
```

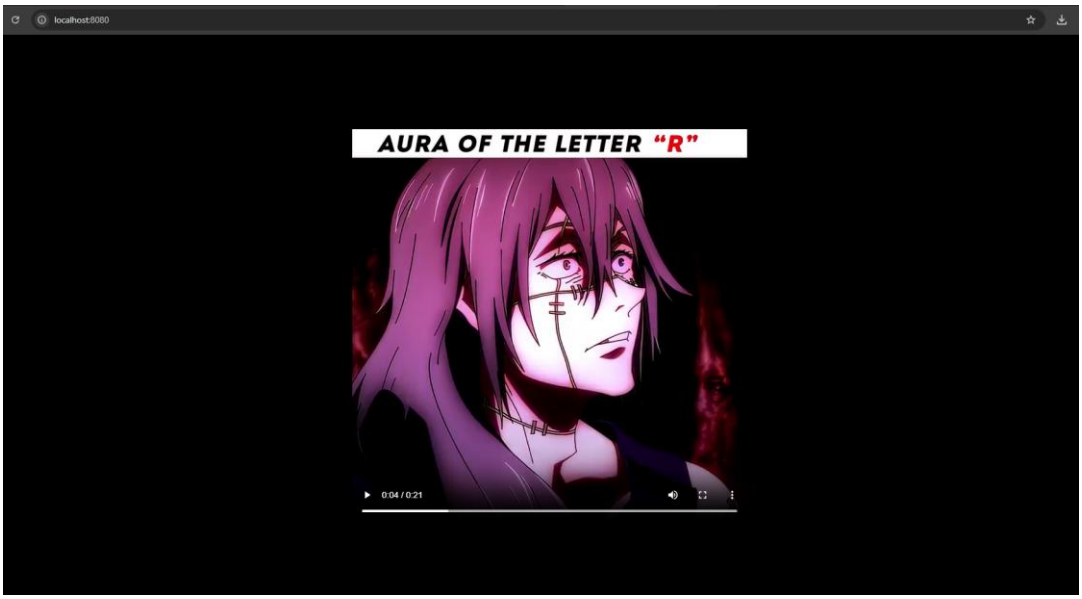
WEEK 05

27) Video Streaming using http Server in browser.

Code

```
const http = require('http');
const fs = require('fs');

const serv = http.createServer(func);
serv.listen(8080);
console.log("Server is running.");

function func(req, res) {
    const path = 'video.mp4';
    const stat = fs.statSync(path);
    const fileSize = stat.size;

    const head = {
        'Content-Length': fileSize,
        'Content-Type': 'video/mp4'
    };
    res.writeHead(200, head);
    const stream = fs.createReadStream(path);
    stream.pipe(res);
}
```

Output

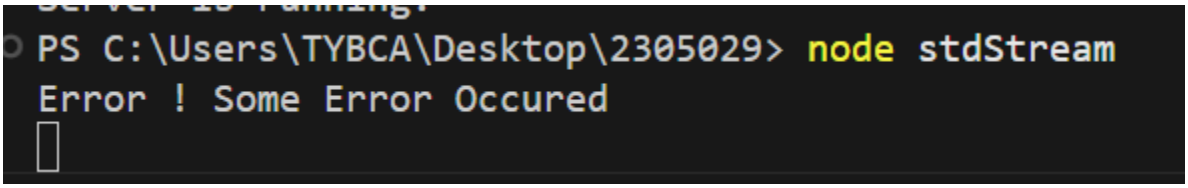


28) std Stream

Code

```
process.stdin.on("data", data => {
    data = data.toString().toUpperCase();
    process.stdout.write(data + "\n");
});

process.stderr.write("Error ! Some Error Occured\n");
```

Output



WEEK 07

29) Node and Express connection to MySQL Workbench Database & API Calling

Codes

Config.json

```
{
  "development": {
    "username": "root",
    "password": null,
    "database": "tybcafsd29",
    "host": "127.0.0.1",
    "dialect": "mysql"
  }
}
```

migrations/Migration.js

```
'use strict';
/** @type {import('sequelize-cli').Migration} */
module.exports = {
  async up(queryInterface, Sequelize) {
    await queryInterface.createTable('Students', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      name: {
        type: Sequelize.STRING
      },
      email: {
        type: Sequelize.STRING
      },
      age: {
        type: Sequelize.INTEGER
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
  async down(queryInterface, Sequelize) {
    await queryInterface.dropTable('Students');
  }
};
```

Models/index.js

```
'use strict';

const fs = require('fs');
const path = require('path');
const Sequelize = require('sequelize');
const basename = path.basename(__filename);
const env = process.env.NODE_ENV || 'development';
const config = require(__dirname + '/../config/config.json')[env];
const db = {};

let sequelize;
if (config.use_env_variable) {
  sequelize = new Sequelize(process.env[config.use_env_variable], config);
} else {
  sequelize = new Sequelize(config.database, config.username, config.password,
config);
}

fs
  .readdirSync(__dirname)
  .filter(file => (
    file.indexOf('.') !== 0 &&
    file !== basename &&
    file.slice(-3) === '.js' &&
    file.indexOf('.test.js') === -1
  ))
  .forEach(file => {
    const model = require(path.join(__dirname, file))(sequelize,
Sequelize.DataTypes);
    db[model.name] = model;
    console.log(`Loaded model: ${model.name}`);
  });

Object.keys(db).forEach(modelName => {
  if (db[modelName].associate) {
    db[modelName].associate(db);
  }
});

db.sequelize = sequelize;
db.Sequelize = Sequelize;

module.exports = db;
```

models/student.js

```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class Student extends Model {
    /**
     * Helper method for defining associations.
     * This method is not a part of Sequelize lifecycle.
     * The `models/index` file will call this method automatically.
     */
    static associate(models) {
      // define association here
    }
  }
  Student.init({
    name: DataTypes.STRING,
    email: DataTypes.STRING,
    age: DataTypes.INTEGER
  }, {
    sequelize,
    modelName: 'Student',
  });
  return Student;
};
```

App.js

```
const express = require('express');
const { Student } = require('./models');

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

// Basic validation middleware for student data
function validateStudent(req, res, next) {
  const { name, email } = req.body;
  if (!name || !email) {
    return res.status(400).json({ error: 'Name and email are required' });
  }
  next();
}

// POST - Create student
app.post('/students', validateStudent, async (req, res) => {
  try {
    const student = await Student.create(req.body);
    res.status(201).json(student);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// GET - All students
app.get('/students', async (req, res) => {
  try {
    const students = await Student.findAll();
    res.json(students);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// GET - One student by ID
app.get('/students/:id', async (req, res) => {
  try {
    const student = await Student.findByPk(req.params.id);
    if (student) {
      res.json(student);
    } else {
      res.status(404).json({ error: 'Student not found' });
    }
```

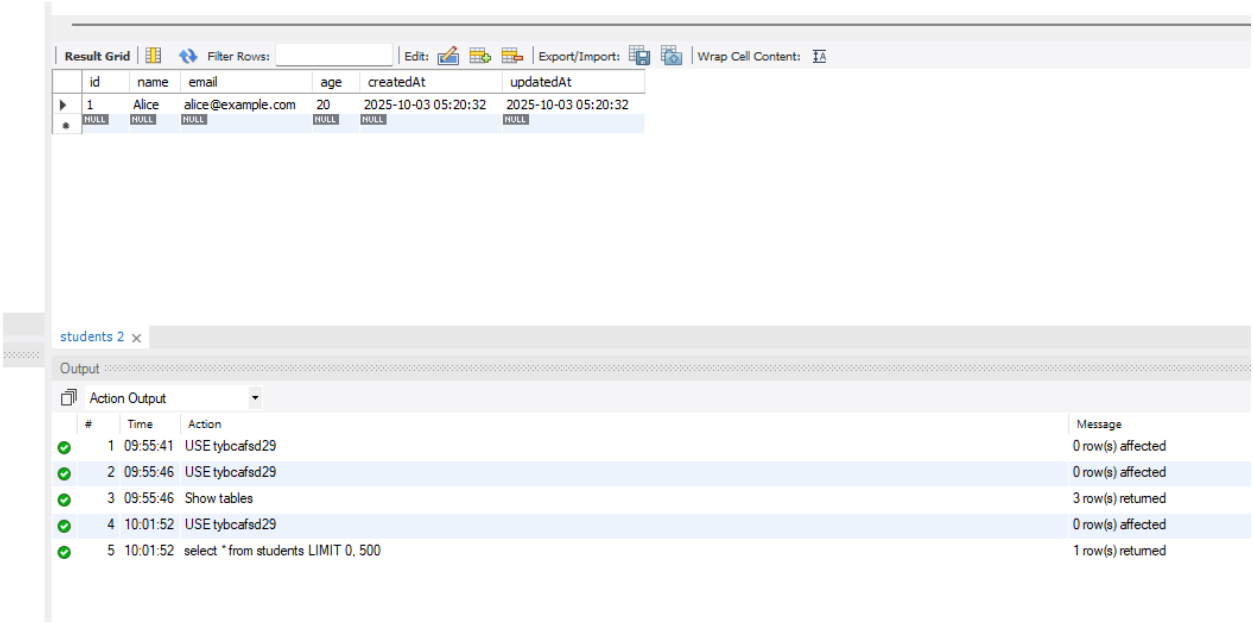
```
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// PUT - Update student by ID
app.put('/students/:id', validateStudent, async (req, res) => {
  try {
    const [updated] = await Student.update(req.body, { where: { id: req.params.id
} });
    if (updated) {
      res.json({ message: 'Updated successfully' });
    } else {
      res.status(404).json({ error: 'Student not found' });
    }
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// DELETE - Delete student by ID
app.delete('/students/:id', async (req, res) => {
  try {
    const deleted = await Student.destroy({ where: { id: req.params.id } });
    if (deleted) {
      res.status(204).send(); // No content on successful delete
    } else {
      res.status(404).json({ error: 'Student not found' });
    }
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// Start server
app.listen(3000, () => {
  console.log('Server is running on http://localhost:3000');
});
```

Db.js connection details

```
const { Sequelize } = require('sequelize');
const sequelize = new Sequelize('tybcafsd29', 'root', '', {
  host: '127.0.0.1',
  dialect: 'mysql',
  port: 3306,
  logging: false,
});
module.exports = sequelize;
```

Output

```
PS C:\Users\TYBCA\Desktop\2305029\DB-codes> node app.js
Loaded model: Student
Loaded model: Users
Server is running on http://localhost:3000
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	email	age	createdAt	updatedAt
1	Alice	alice@example.com	20	2025-10-03 05:20:32	2025-10-03 05:20:32
NULL	NULL	NULL	NULL	NULL	NULL

students 2

Output

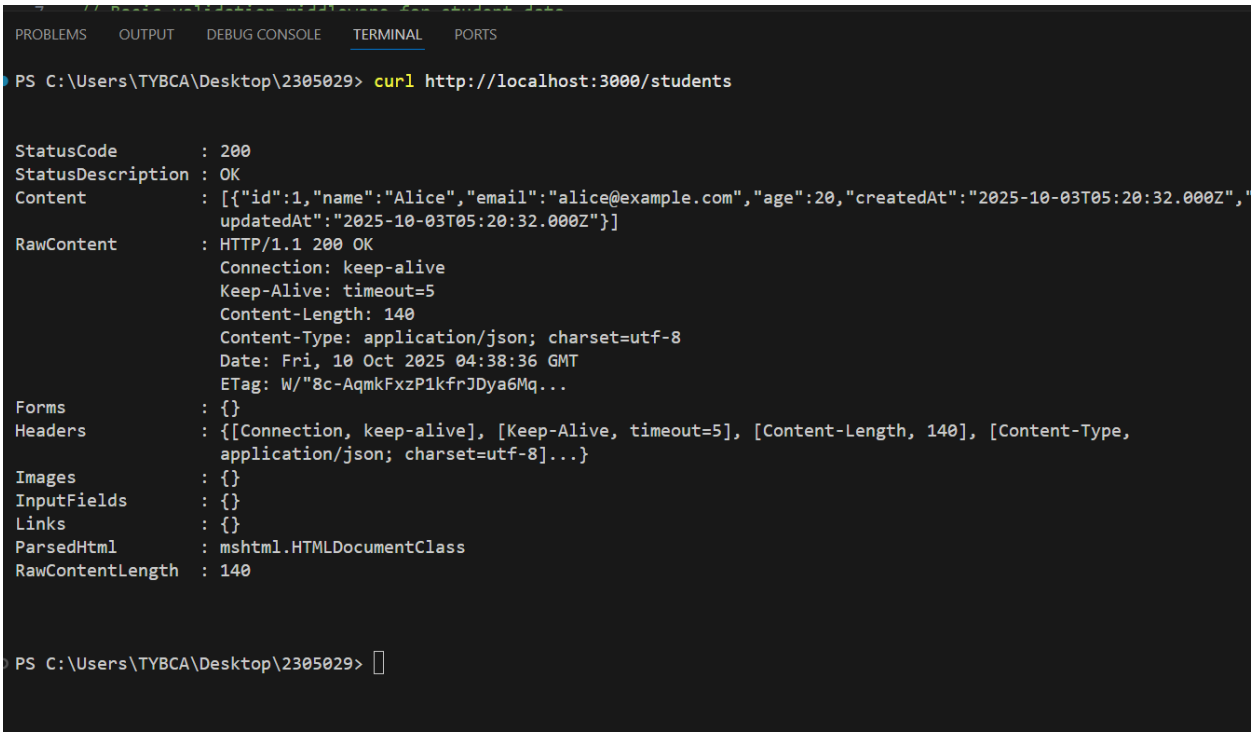
Action Output

#	Time	Action	Message
1	09:55:41	USE tybcafsd29	0 row(s) affected
2	09:55:46	USE tybcafsd29	0 row(s) affected
3	09:55:46	Show tables	3 row(s) returned
4	10:01:52	USE tybcafsd29	0 row(s) affected
5	10:01:52	select * from students LIMIT 0, 500	1 row(s) returned

Home page

/students page

API using CURL Command



WEEK 09 + WEEK 10

30) The Week 9 project on *Model Querying, Validation, and Raw Queries* in Node.js using Sequelize and Express, describing how it evolves from ORM-based querying to adding validations and constraints, and finally to using raw SQL queries.

Folder Structure

```
week9/
├── db.js
├── package.json
├── readme.md
├── ModelQueryingFinders/
│   ├── index.js
│   ├── model/
│   │   └── user.js
│   └── queries/
│   └── userQueries.js
├── RawQueries/
│   ├── index.js
│   ├── model/
│   │   └── user.js
│   └── queries/
│   └── userQueries.js
├── ValidationConstraints/
│   ├── index.js
│   ├── model/
│   │   └── user.js
│   └── queries/
│   └── userQueries.js
```

- .env

```
DB_NAME=tybcafsd29
DB_USER=root
DB_PASS=
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DIALECT=mysql
```

- Db.js

```
// db.js
const { Sequelize } = require('sequelize');
const path = require('path');

require('dotenv').config({ path: path.resolve(__dirname, '.env') });

console.log('DB_NAME:', process.env.DB_NAME);
console.log('DB_USER:', process.env.DB_USER);
console.log('DB_PASS:', process.env.DB_PASS);
console.log('DB_HOST:', process.env.DB_HOST);
console.log('DB_DIALECT:', process.env.DB_DIALECT);
console.log('DB_PORT:', process.env.DB_PORT);

const sequelize = new Sequelize(
  process.env.DB_NAME,
  process.env.DB_USER,
  process.env.DB_PASS,
  {
    host: process.env.DB_HOST,
    dialect: process.env.DB_DIALECT,
    port: process.env.DB_PORT,
    logging: false
  }
);

module.exports = sequelize;
```

- **Users Model**
  File: week09/ModelQueryingFinders/model/user.js
  Purpose: Define users table schema

  Code

```
'use strict';
const { DataTypes } = require('sequelize');
const sequelize = require('../../db');

const Users = sequelize.define('Users', {
  name: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  email: {
    type: DataTypes.STRING,
    allowNull: false,
    unique: true,
  },
  age: {
    type: DataTypes.INTEGER,
  },
});

module.exports = Users;
```

- **User Queries**
  File: week09/ModelQueryingFinders/queries/userQueries.js
  Purpose: CRUD operations for users table

  Code

```
const Users = require('../model/users');

async function createUser(payload) { return await
Users.create(payload); }
async function getAllUsers() { return await Users.findAll(); }
async function findUserById(id) { return await Users.findByPk(id); }
async function findUsersByName(name) { return await Users.findAll({
where: { name } }); }
async function updateUser(id, payload) {
  const [updated] = await Users.update(payload, { where: { id } });
  return updated;
}
async function deleteUser(id) { return await Users.destroy({ where: {
id } }); }

module.exports = { createUser, getAllUsers, findUserById,
findUsersByName, updateUser, deleteUser };
```

- **Express Server**
  File: week09/ModelQueryingFinders/index.js
  Purpose: Setup server and define API routes

  Code

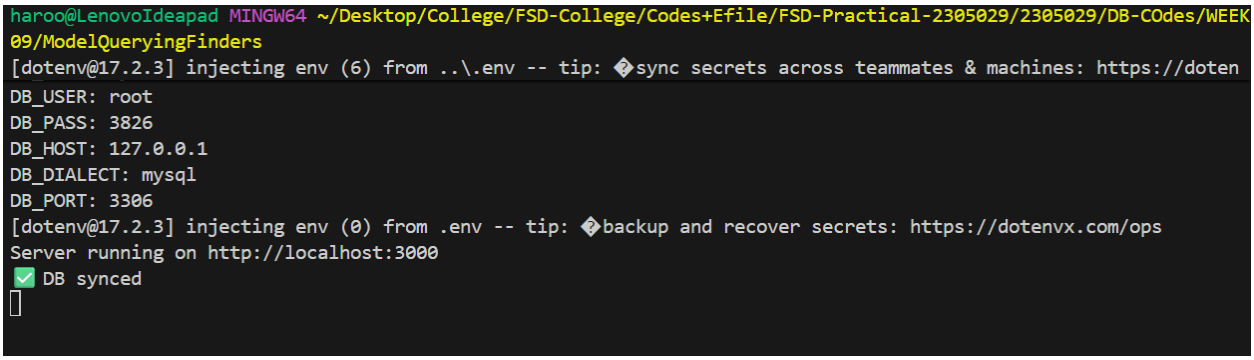
```
const express = require('express');
const Users = require('./model/user');
const sequelize = require('../../db');

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

// Sync DB
sequelize.sync({ force: false }).then(() => console.log('✅ DB
synced'));

// Routes
app.get('/users', async (req, res) => {
  try {
    const users = await Users.findAll();
    res.json(users);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.get('/users/:id', async (req, res) => {
  try {
    const user = await Users.findByPk(req.params.id);
    if (!user) return res.status(404).json({ error: 'User not found' });
    res.json(user);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
```

```
});

app.post('/users', async (req, res) => {
  try {
    const newUser = await Users.create(req.body);
    res.json(newUser);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.put('/users/:id', async (req, res) => {
  try {
    const [updated] = await Users.update(req.body, { where: { id:
req.params.id } });
    if (!updated) return res.status(404).json({ error: 'User not found' });
    res.json({ message: 'User updated' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.delete('/users/:id', async (req, res) => {
  try {
    const deleted = await Users.destroy({ where: { id: req.params.id }
});
    if (!deleted) return res.status(404).json({ error: 'User not found' });
    res.json({ message: 'User deleted' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

const PORT = 3000;
```

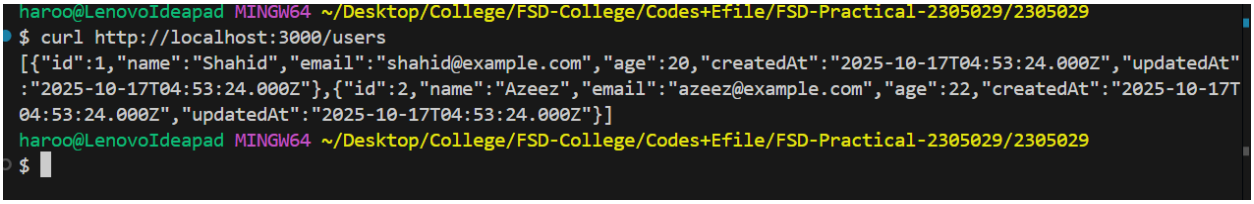
app.listen(PORT, () => console.log(`Server running on http://localhost:${PORT}`));

- **Outputs / Postman Requests**
  Purpose: Capture API responses for documentation

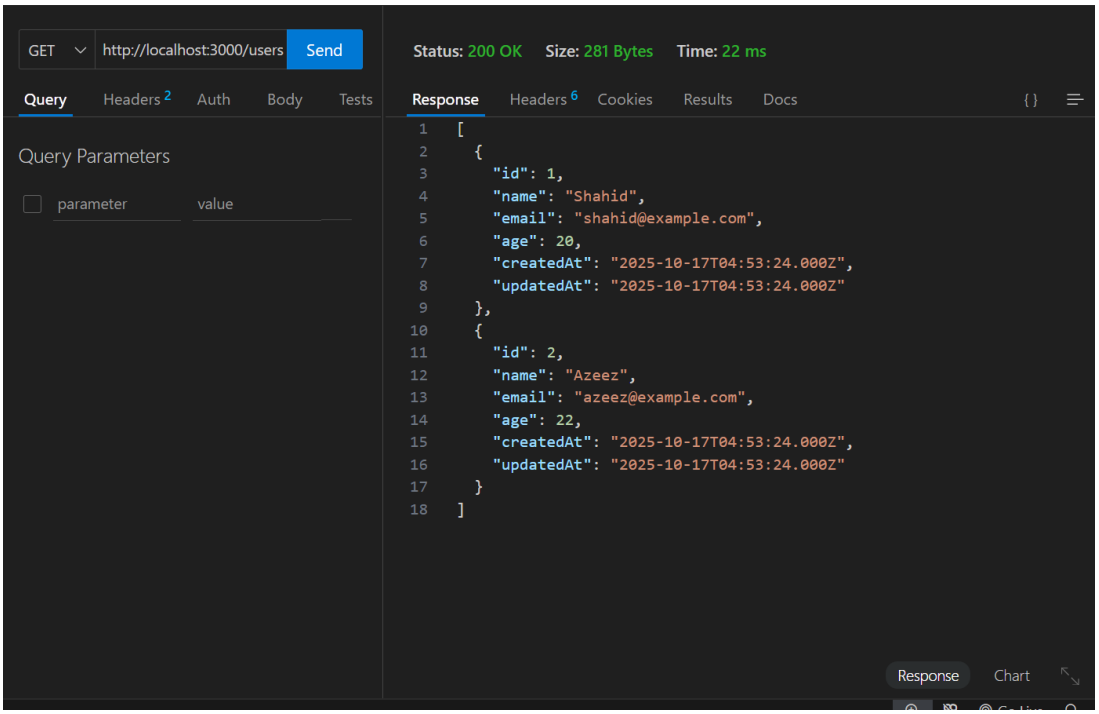
Server Start



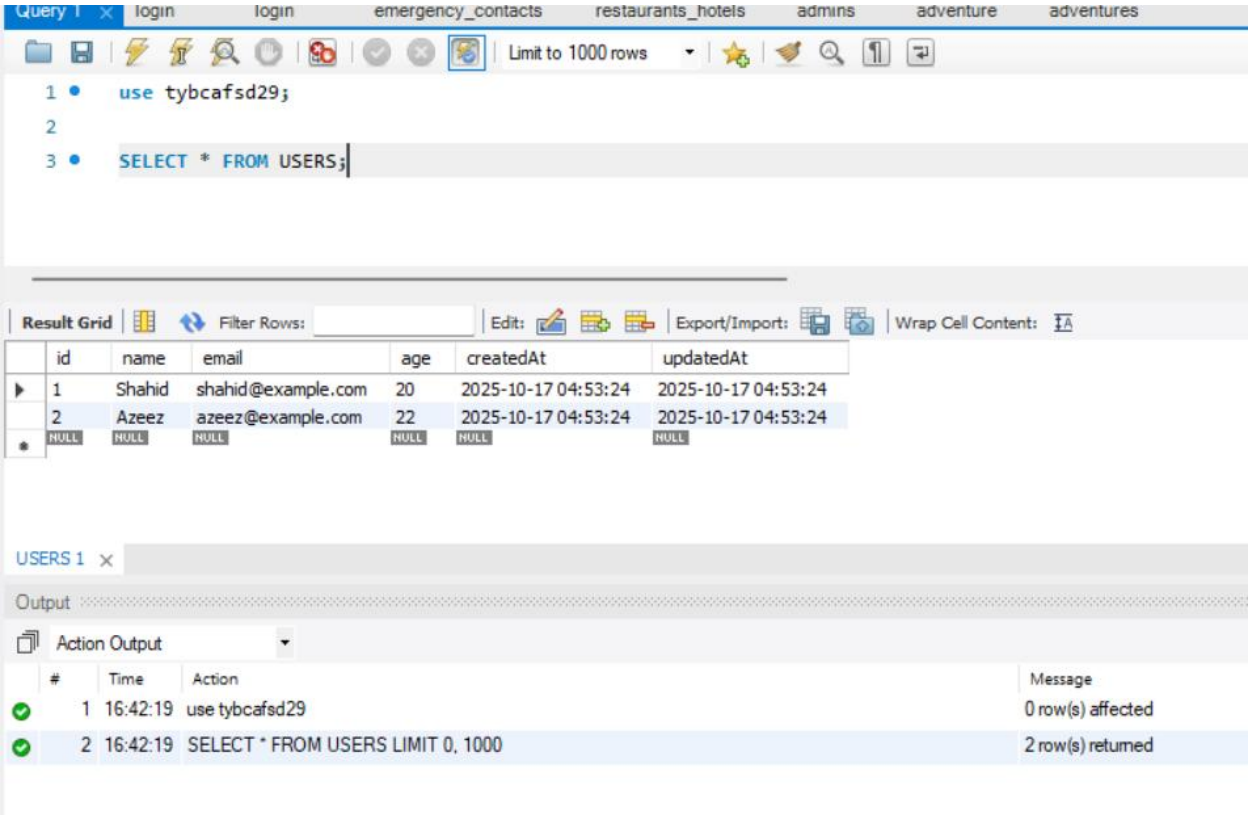
API Using CURL Command



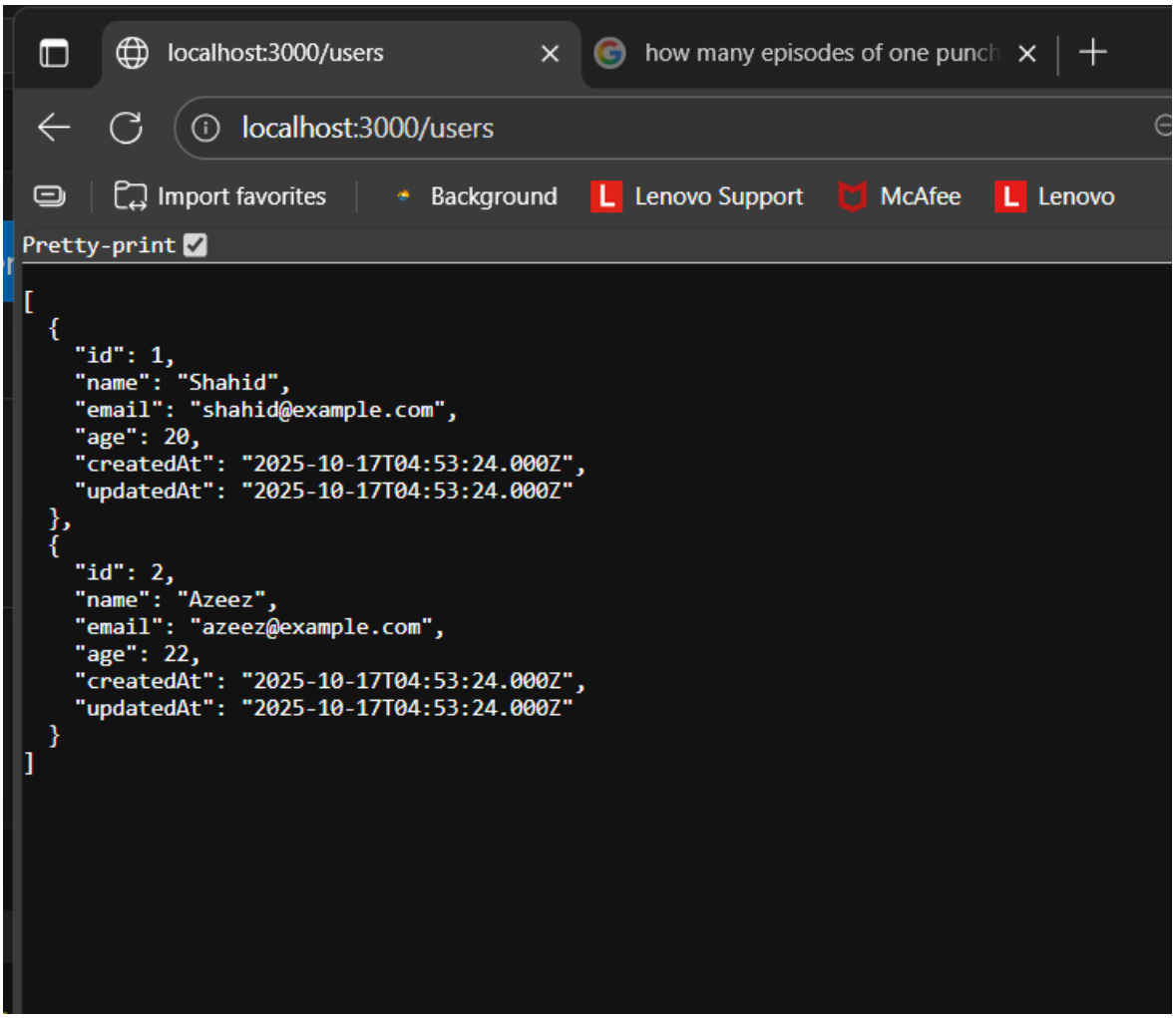
API Using ThunderClient Postman



API in MySQL Workbench

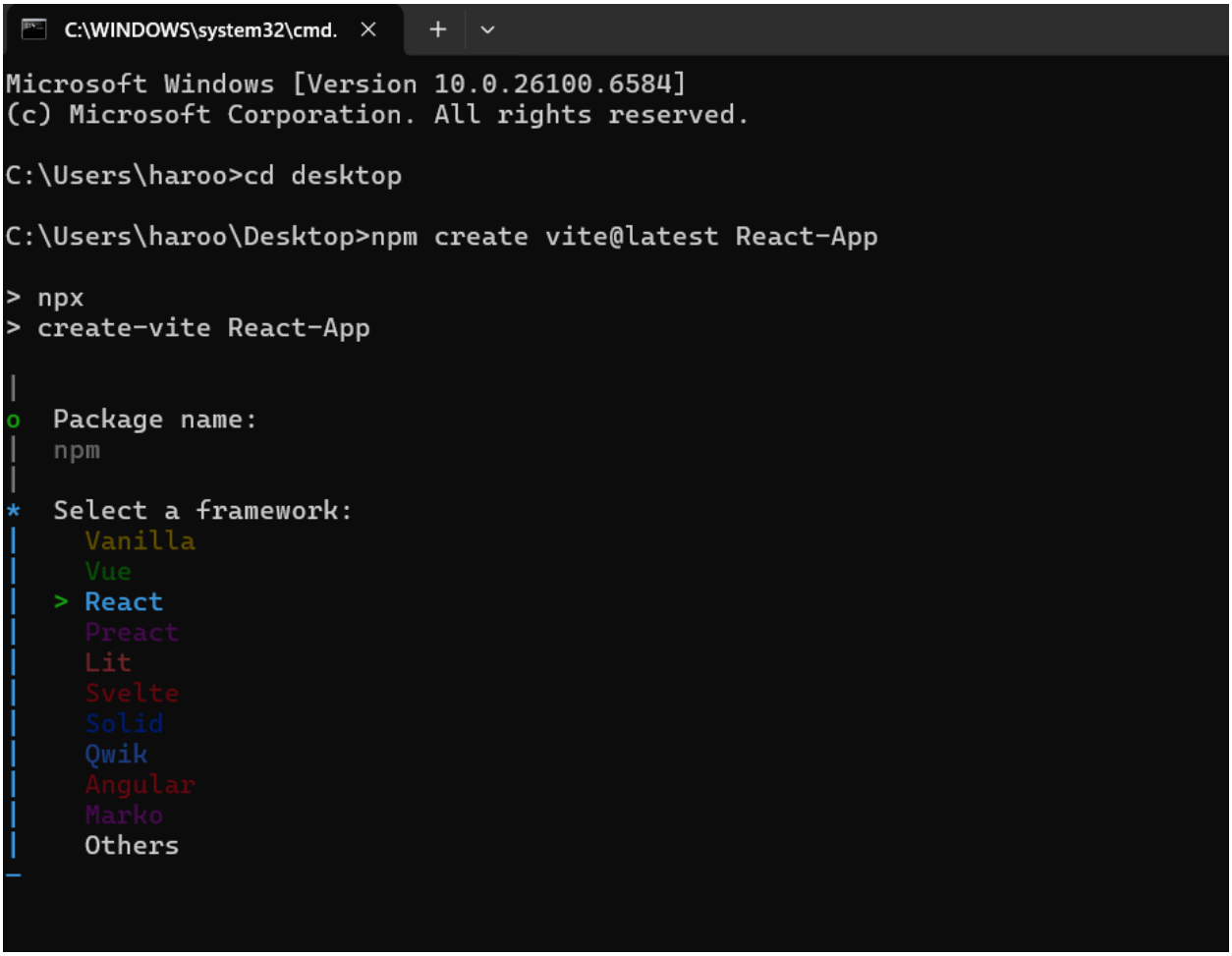


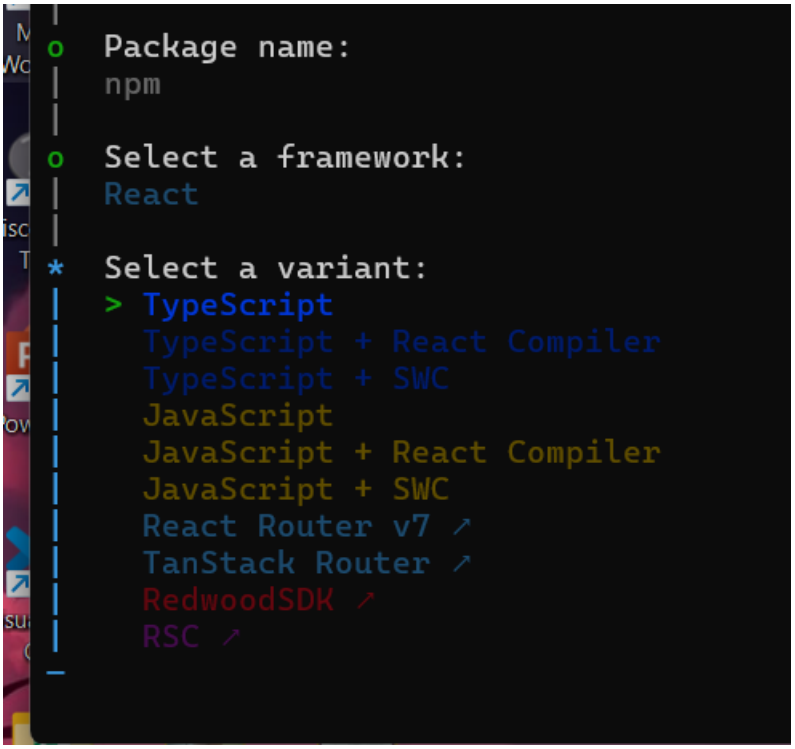
API in the Browser

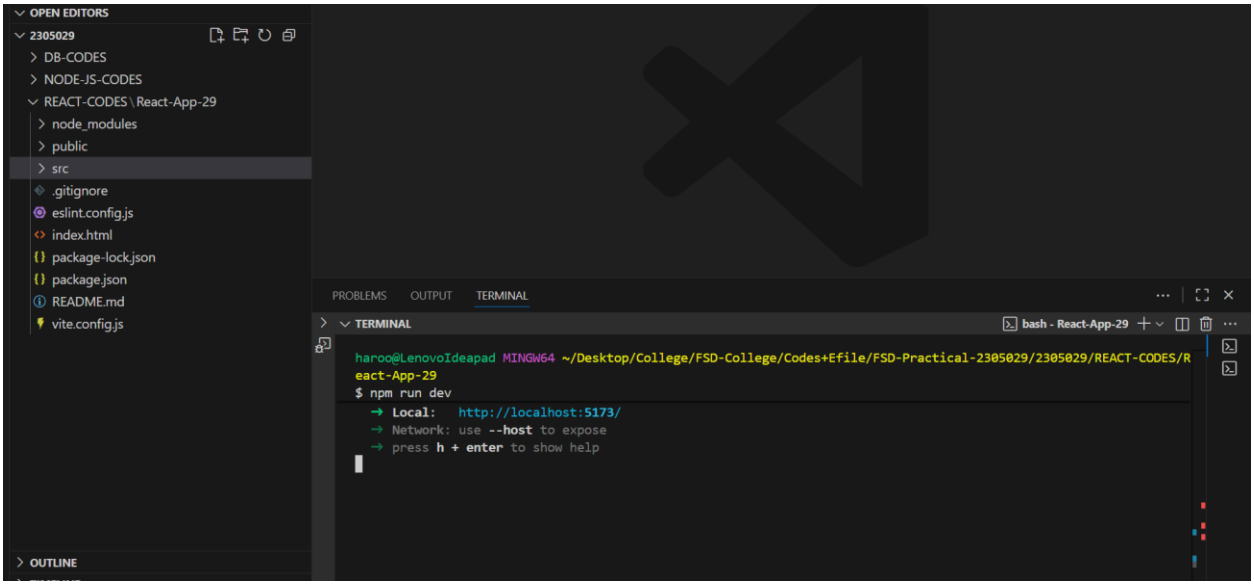


WEEK 11

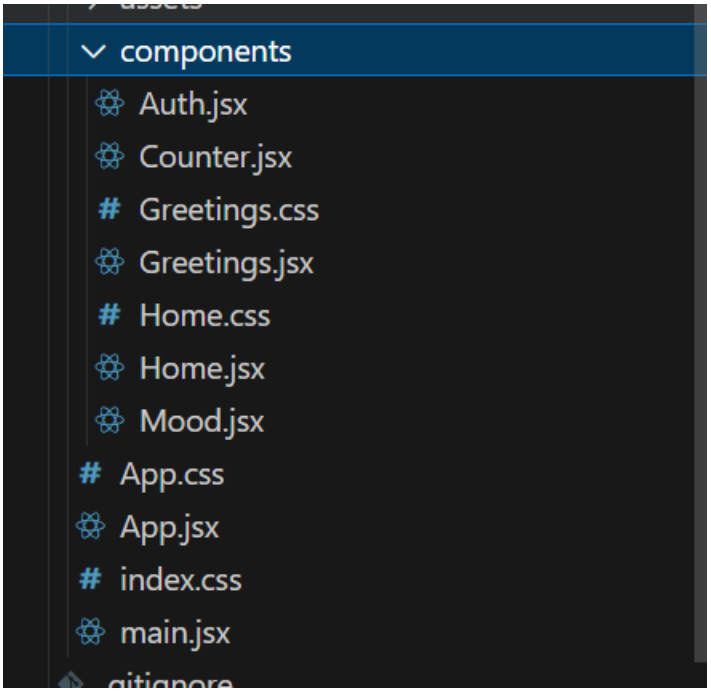
- Installation of React




Package name:
npm
Select a framework:
React
Select a variant:
> TypeScript
TypeScript + React Compiler
TypeScript + SWC
JavaScript
JavaScript + React Compiler
JavaScript + SWC
React Router v7 ↗
TanStack Router ↗
RedwoodSDK ↗
RSC ↗


OPEN EDITORS
2305029
DB-CODES
NODE-JS-CODES
REACT-CODES \ React-App-29
node_modules
public
src
.gitignore
eslint.config.js
index.html
package-lock.json
package.json
README.md
vite.config.js
OUTLINE
PROBLEMS
OUTPUT
TERMINAL
TERMINAL
bash - React-App-29
harop@LenovoIdeapad MINGW64 ~/Desktop/College/FSD-College/Codes+Efile/FSD-Practical-2305029/2305029/REACT-CODES/React-App-29
$ npm run dev
➜ Local: http://localhost:5173/
➜ Network: use --host to expose
➜ press h + enter to show help

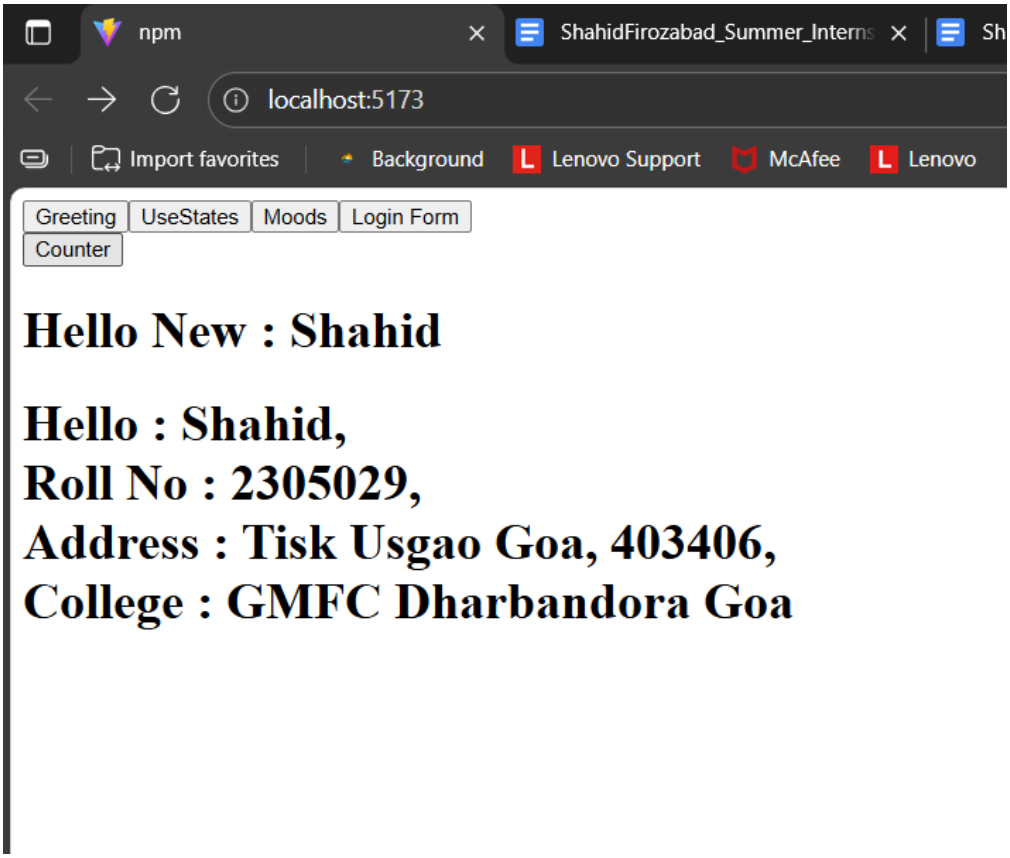
- Components



- Props

```
import React from "react";
const Greetings = (props) => {
    const name = "Shahid";
    const Rollno = 2305029;
    const Address = "Tisk Usgao Goa, 403406";
    const College = "GMFC Dharbandora Goa";

    return (
        <>
        <a href="/Counter"><button>Counter</button></a>
        <h1>Hello New : {props.name}</h1>
        <h1>Hello : {name},<br></br> Roll No : {Rollno},<br />Address :
{Address},<br />College : {College}</h1>
        </>
    )

}
const App = () => {

    return (
        <>
        <Greetings name="Shahid" />
        </>
    )
```

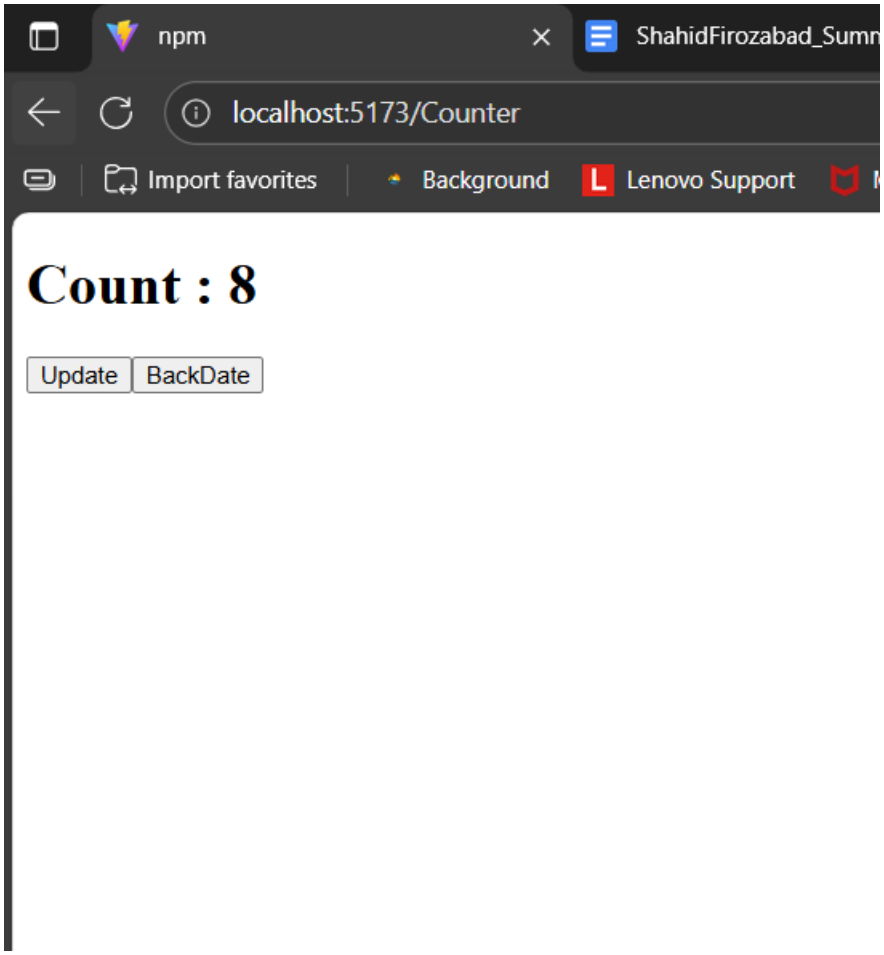
```
}
export default App;
```



- Use States
- Counter Button using states

```
import { use, useState } from "react";
const Counter = () => {
const [count, setCount] = useState(0);

return (
    <>
    <div className="Counter-container">
        <h1>Count : {count}</h1>
        <button onClick={() => setCount(count + 1)}>Update</button>
        <button onClick={() => setCount(count - 1)}>BackDate</button>
    </div>
    </>
)
}

export default Counter;
```

Output


npm
ShahidFirozabad_Sumn
localhost:5173/Counter
Import favorites
Background
Lenovo Support
Count : 8
Update
BackDate

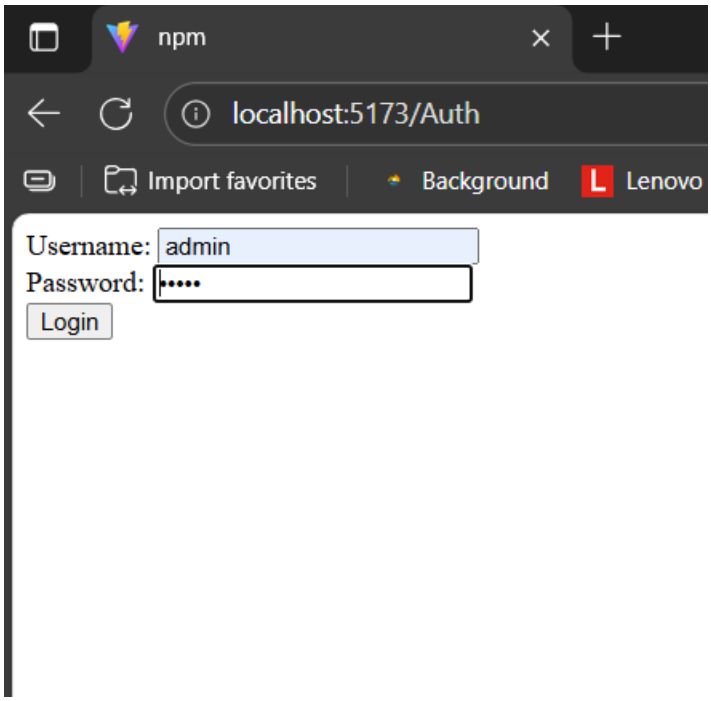
WEEK 12

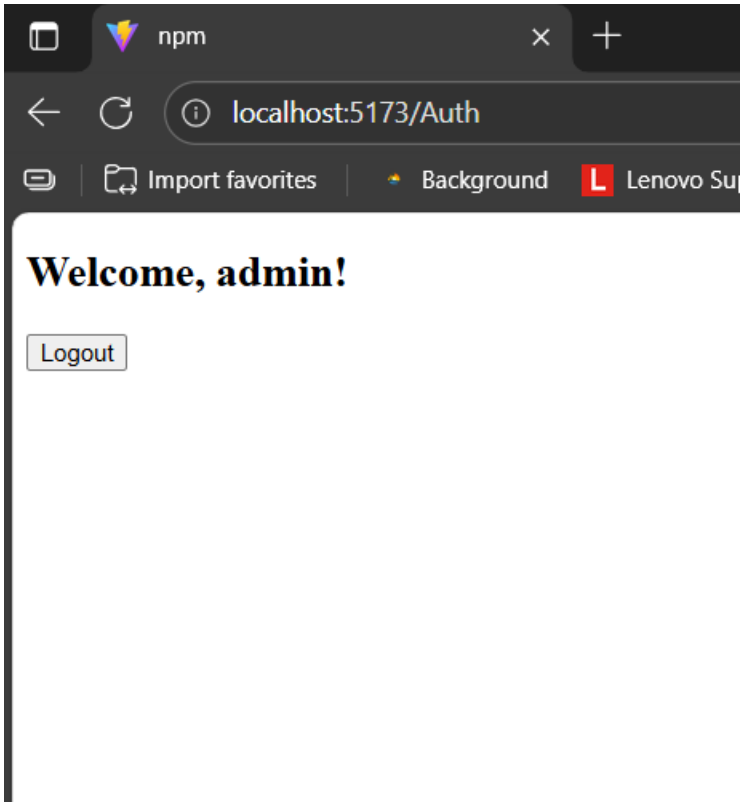
- **Hooks**
  - useState
  - **Authentication Login & Signup**

```jsx
// LoginForm.jsx
import React, { useState } from "react";

function LoginForm() {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  const handleSubmit = (e) => {
    e.preventDefault();
    if (username === "admin" && password === "1234") {
      setIsLoggedIn(true);
    } else {
      alert("Invalid credentials!");
    }
  };

  const handleLogout = () => {
    setUsername("");
    setPassword("");
    setIsLoggedIn(false);
  };

  return (
    <div>
      {isLoggedIn ? (
        <>
          <h2>Welcome, {username}!</h2>
          <button onClick={handleLogout}>Logout</button>
        </>
      ) : (
        <form onSubmit={handleSubmit}>
          <div>
            <label>Username: </label>
            <input
              type="text"
              value={username}
              onChange={(e) => setUsername(e.target.value)}
```

```
            />
          </div>

          <div>
            <label>Password: </label>
            <input
              type="password"
              value={password}
              onChange={(e) => setPassword(e.target.value)}
            />
          </div>

          <button type="submit">Login</button>
        </form>
      )}
    </div>
  );
}

export default LoginForm;
```

Output





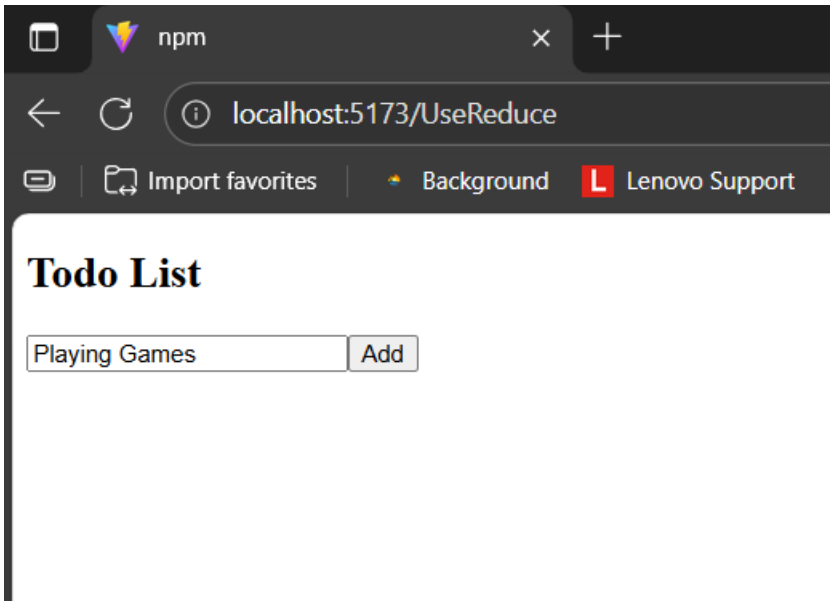
- useReducer

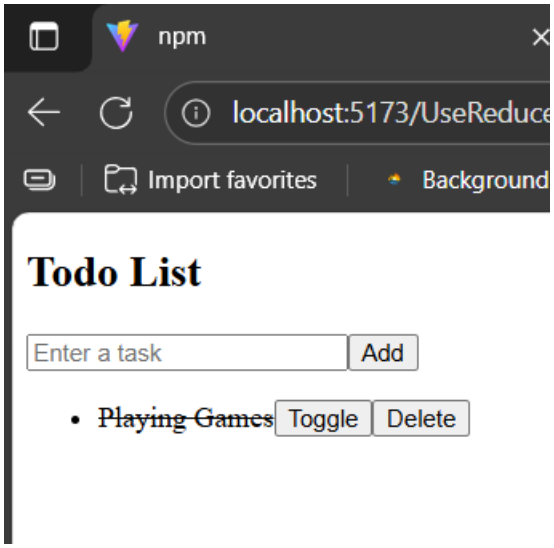
```
import React, { useReducer } from "react";

// Reducer function
function todoReducer(state, action) {
  switch (action.type) {
    case "add":
      return [...state, { id: Date.now(), text: action.text, completed:
false }];
    case "toggle":
      return state.map(todo =>
        todo.id === action.id ? { ...todo, completed: !todo.completed } :
todo
      );
    case "delete":
      return state.filter(todo => todo.id !== action.id);
    default:
      return state;
  }
}
```

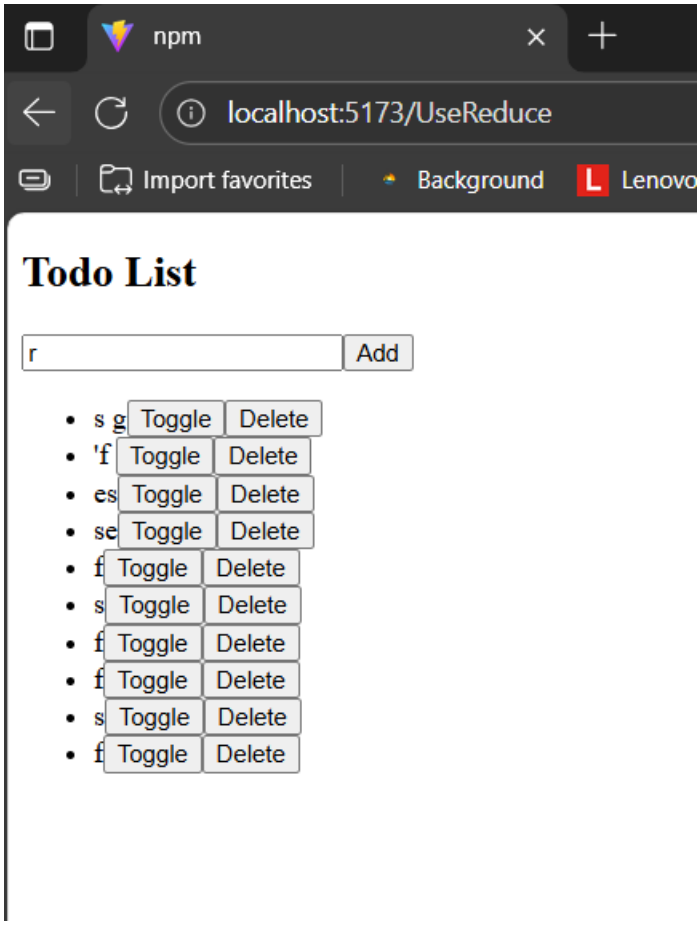
```
function TodoApp() {
  const [todos, dispatch] = useReducer(todoReducer, []);
  const [text, setText] = React.useState("");

  const handleSubmit = (e) => {
    e.preventDefault();
    if (text.trim() === "") return;
    dispatch({ type: "add", text });
    setText("");
  };

  return (
    <div>
      <h2>Todo List</h2>
      <form onSubmit={handleSubmit}>
        <input
          value={text}
          onChange={(e) => setText(e.target.value)}
          placeholder="Enter a task"
        />
        <button type="submit">Add</button>
      </form>

      <ul>
        {todos.map(todo => (
          <li
            key={todo.id}
            style={{ textDecoration: todo.completed ? "line-through" :
"none" }}
          >
            {todo.text}
            <button onClick={() => dispatch({ type: "toggle", id: todo.id
})}>
              Toggle
            </button>
            <button onClick={() => dispatch({ type: "delete", id: todo.id
})}>
              Delete
            </button>
          </li>
        ))}
      </ul>
    </div>
  );
```

```
}

export default TodoApp;
```

Outputs







- useContext

- useRef
- useEffect
- useMemo
- useCallback

WEEK 13

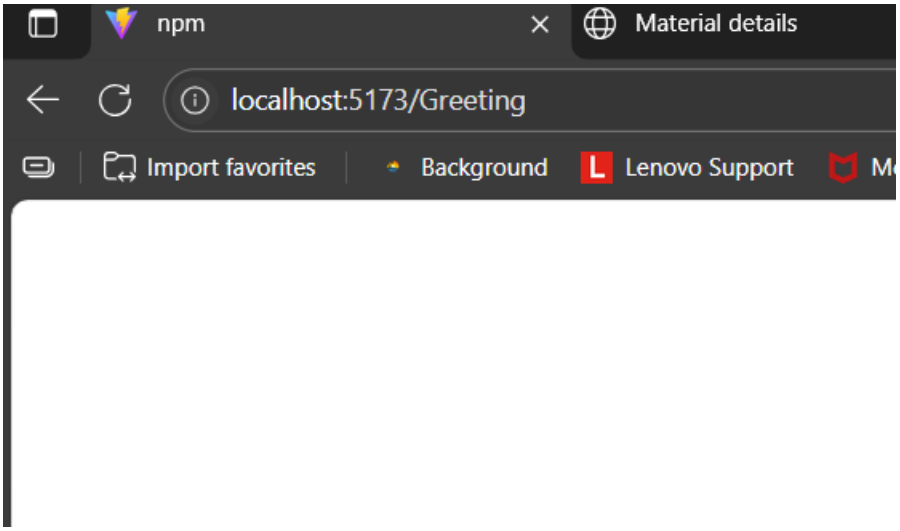
- **Routing**
  - Browser Router

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import { BrowserRouter, Routes, Route } from 'react-router-dom'; // ☑
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <BrowserRouter>
    <App />
  </BrowserRouter>,
)
```

  - App.jsx

```
import { useState } from 'react'
import { Routes, Route } from 'react-router-dom'
import Home from "./components/Home.jsx";
import Greetings from "./components/Greetings.jsx";
import Counter from "./components/Counter.jsx"
import Moods from "./components/Mood.jsx";
import Auth from "./components/Auth.jsx";
import UseReduce from "./components/UseReducer.jsx";
import UseContext from "./components/UseContext.jsx";

function App() {

  return (
    <>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/Greetings" element={<Greetings />} />
        <Route path="/Counter" element={<Counter />} />
        <Route path="/Moods" element={<Moods />} />
        <Route path="/Auth" element={<Auth />} />
        <Route path="/UseReduce" element={<UseReduce />} />
        <Route path="/UseContext" element={<UseContext />} />
      </Routes>
    </>
  )
}

export default App;
```

- Route image

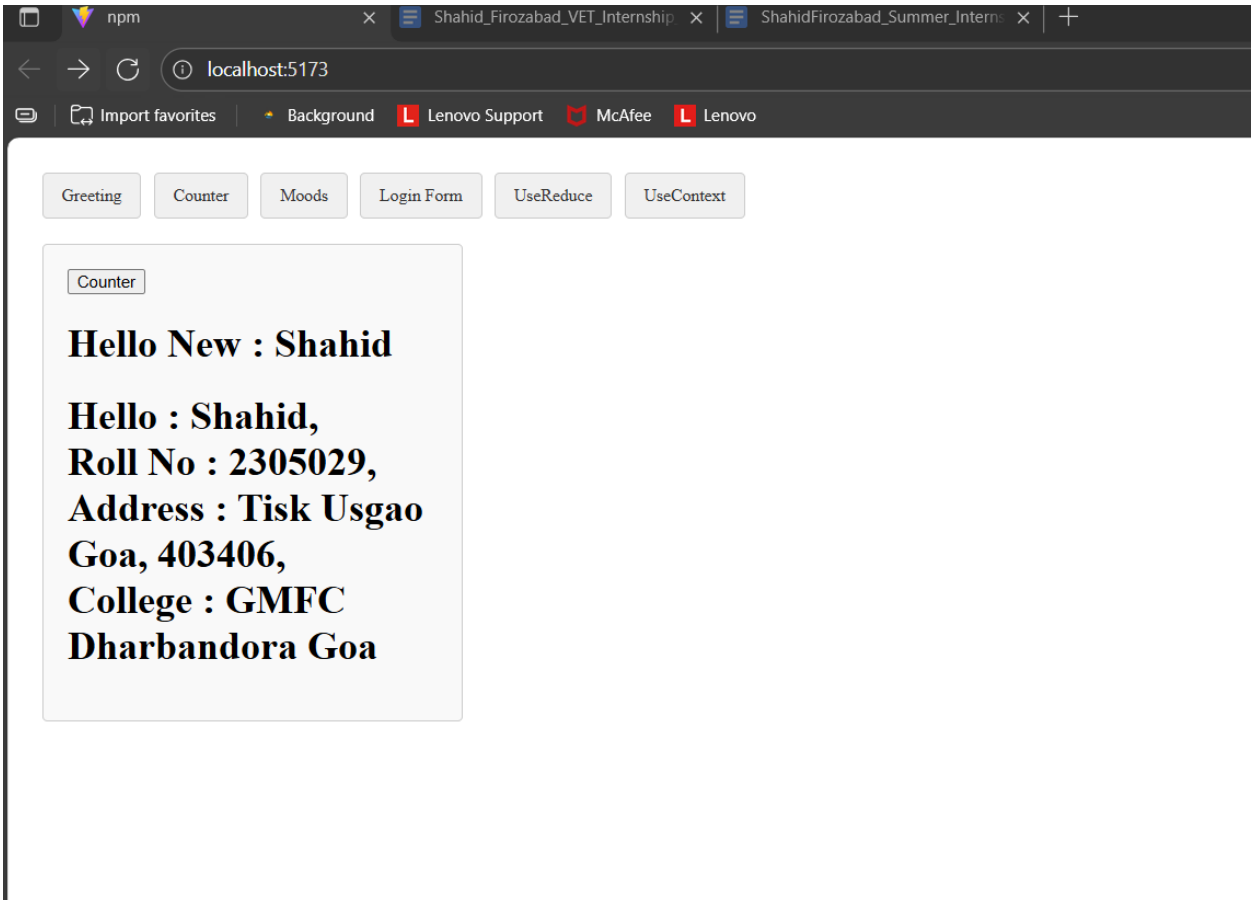


- **Navigation**

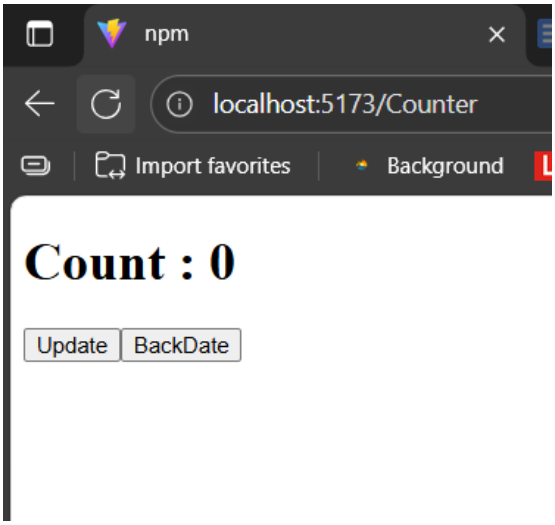
```
import React from "react";
import "./Home.css";
import Greetings from "./Greetings";
import { Link } from 'react-router-dom';

const Home = () => {
    return (
        <div className="home-container">
            <nav className="nav-bar">
                <Link to="/Greeting" className="nav-button">Greeting</Link>
                <Link to="/Counter" className="nav-button">Counter</Link>
                <Link to="/Moods" className="nav-button">Moods</Link>
                <Link to="/Auth" className="nav-button">Login Form</Link>
                <Link to="/UseReduce" className="nav-button">UseReduce</Link>
                <Link to="/UseContext" className="nav-button">UseContext</Link>
            </nav>
            <div className="greetings">
                <Greetings name="Shahid" />
            </div>
        </div>
    )
}
```

export default Home;

Ouput (Navbar)



Count Route + Navgitation



WEEK 14

- **Redux**

```
// components/infoSlice.jsx
import { createSlice } from "@reduxjs/toolkit";

const infoSlice = createSlice({
  name: "info",
  initialState: {
    course: "Bachelor of Computer Applications (BCA)",
    duration: "3 Years / 6 Semesters",
    contact: "ramkrishna@gmail.com"
  },
  reducers: {
    updateContact(state, action) {
      state.contact = action.payload;
    }
  }
});

export const { updateContact } = infoSlice.actions;
export default infoSlice.reducer; // default export is reducer (not a
component)
```

// components/Store.jsx

import { configureStore } from "@reduxjs/toolkit";

import infoReducer from "./infoSlice";

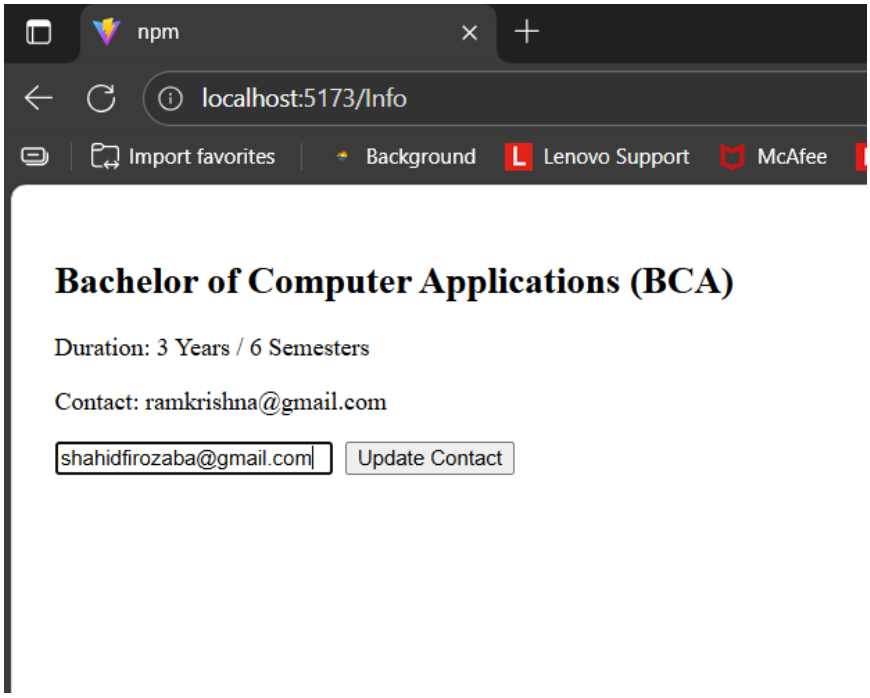
const store = configureStore({

  reducer: { info: infoReducer }

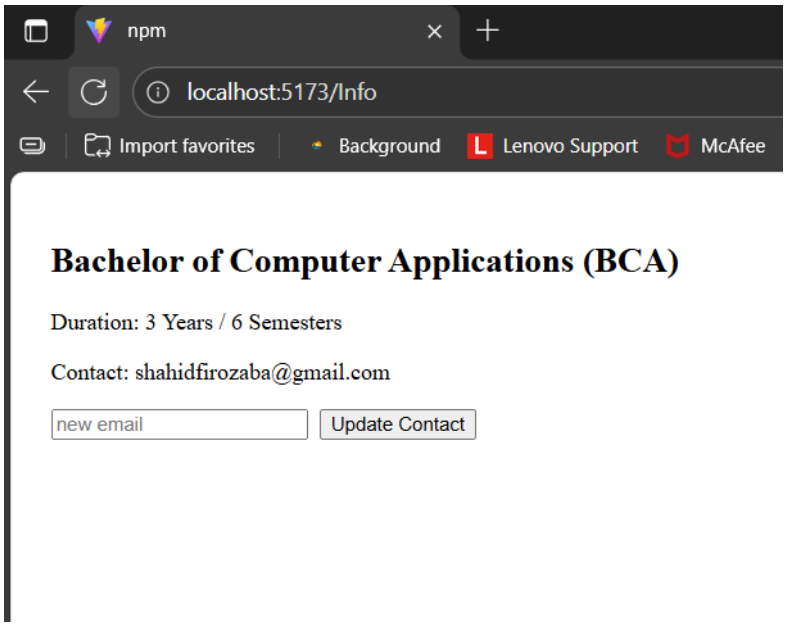
});

export default store;

Before updating Email

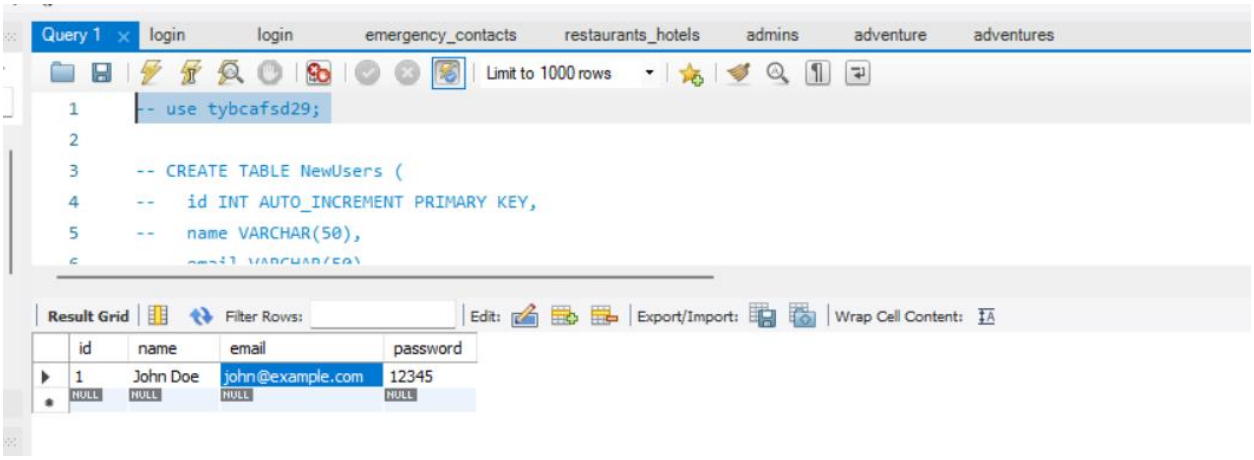


After Adding



WEEK 15

➢ Data Base



➢ Integrate Node with React JS

- **Server.js**

```
// server/server.js
const express = require("express");
const mysql = require("mysql2");
const cors = require("cors");
const bodyParser = require("body-parser");

const app = express();
const PORT = 5000;

// Middleware
app.use(cors());           // allow requests from frontend
app.use(bodyParser.json()); // parse JSON

// MySQL connection
const db = mysql.createConnection({
  host: "localhost",
  user: "root",           // your MySQL username
  password: "3826",    // your MySQL password
  database: "tybcafsd29"  // your database
});

db.connect((err) => {
  if (err) throw err;
  console.log("Connected to MySQL database tybcafsd29");
```

```
});

// Login route
app.post("/login", (req, res) => {
  const { email, password } = req.body;
  if (!email || !password) {
    return res.status(400).json({ message: "Please enter email and password"
});
  }

  const query = "SELECT * FROM NewUsers WHERE email = ? AND password = ?";
  db.query(query, [email, password], (err, results) => {
    if (err) return res.status(500).json({ message: "Database error", error:
err });

    if (results.length > 0) {
      res.json({ message: "Login successful", user: results[0] });
    } else {
      res.status(401).json({ message: "Invalid credentials" });
    }
  });
});

app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

Output

```
Node.js v22.12.0

haroo@LenovoIdeapad MINGW64 ~/Desktop/College/FSD-College/Codes+Efile/FSD-Practical-2305029/2305029/React-Codes/Re
act-App-29/Backend
$ node server
Server running on port 5000
Connected to MySQL database tybcafsd29
```

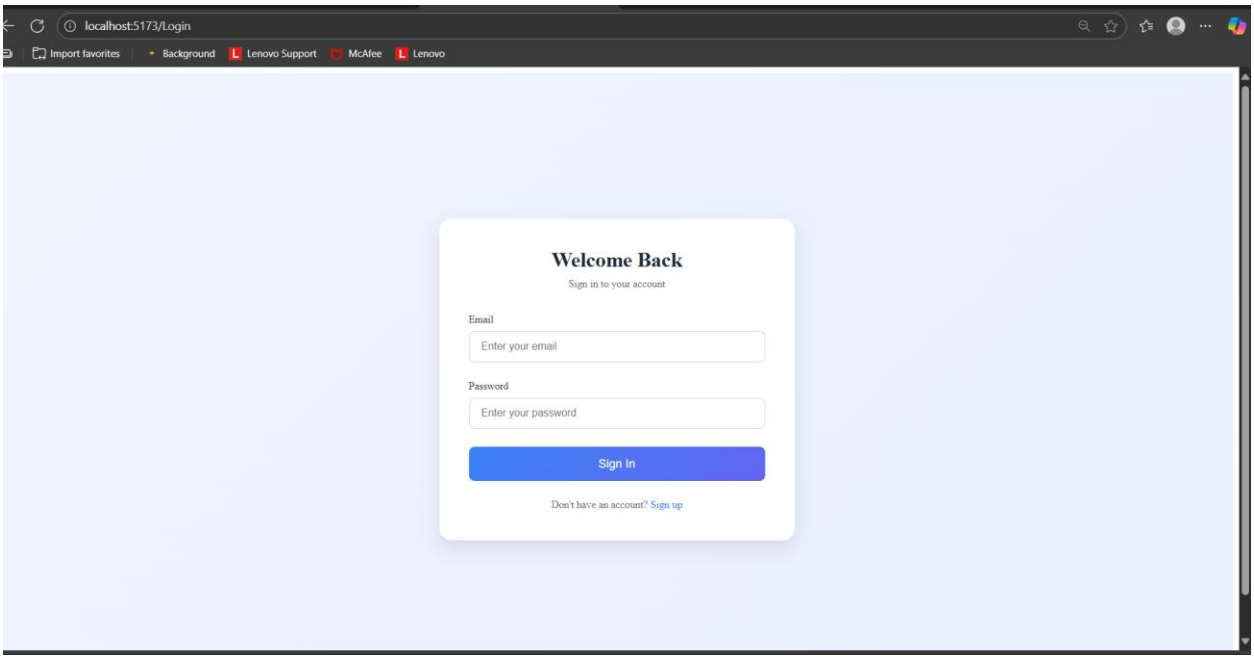
- **Login.jsx**

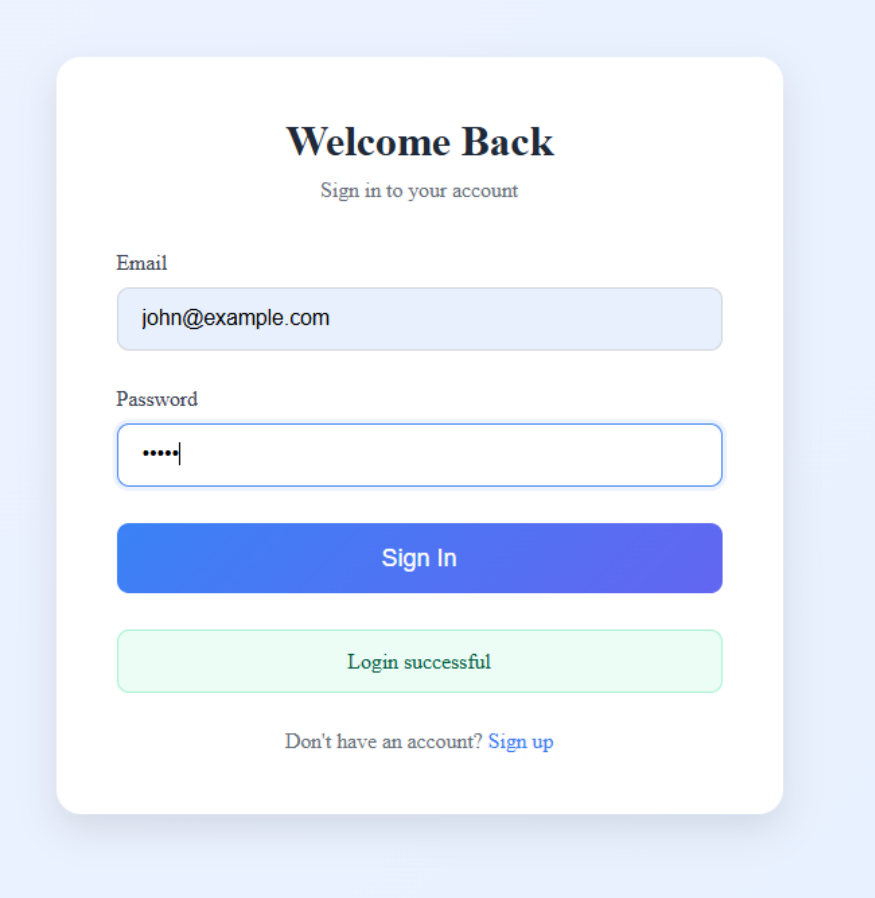
```
// src/components/Login.jsx
import React, { useState } from "react";
import axios from "axios";
import "./Login.css"; // We'll create this CSS file

function Login() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [message, setMessage] = useState("");
  const [isLoading, setIsLoading] = useState(false);

  const handleLogin = async (e) => {
    e.preventDefault();
    setIsLoading(true);
    try {
      const res = await axios.post("http://localhost:5000/login", { email,
password });
      setMessage(res.data.message);
    } catch (err) {
      if (err.response) setMessage(err.response.data.message);
      else setMessage("Server error");
    } finally {
      setIsLoading(false);
    }
  };

  return (
    <div className="login-container">
      <div className="login-card">
        <div className="login-header">
          <h2>Welcome Back</h2>
          <p>Sign in to your account</p>
        </div>

        <form onSubmit={handleLogin} className="login-form">
          <div className="input-group">
            <label>Email</label>
            <input
              type="email"
              placeholder="Enter your email"
              value={email}
              onChange={(e) => setEmail(e.target.value)}
```

```
            required
            className="login-input"
          />
        </div>

        <div className="input-group">
          <label>Password</label>
          <input
            type="password"
            placeholder="Enter your password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            required
            className="login-input"
          />
        </div>

        <button
          type="submit"
          disabled={isLoading}
          className="login-button"
        >
          {isLoading ? (
            <div className="loading-spinner">
              <div className="spinner"></div>
              Signing in...
            </div>
          ) : (
            "Sign In"
          )}
        </button>
      </form>

      {message && (
        <div className={`message ${message.includes("error") || message ===
"Server error" ? "error" : "success"}`}>
          {message}
        </div>
      )}

      <div className="login-footer">
        <p>
          Don't have an account?{" "}
          <a href="#" className="signup-link">Sign up</a>
        </p>
```

```
        </div>
      </div>
    </div>
  );
}

export default Login;
```

Output




Welcome Back
Sign in to your account
Email
john@example.com
Password
Sign In
Login successful
Don't have an account? Sign up