**Tutorial – 1**

1. Design html page to get below given output using bootstrap css.

**CODE :**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>UI</title>

<link rel="stylesheet" href="../css/bootstrap.min.css">

<script src="../js/bootstrap.bundle.min.js"></script>

<style type="text/css">

.primary{

background: #1900ff; color: #1900ff;

}

.warning{

background: #feff00; color: #feff00;

}

.success{

background: #008100; color: #008100;

}

.cyan{

background: #8CF08C; color: #8CF08C;

}

.oreng{

background: #ffa700; color: #ffa700;

}

.gray{

background: #808080; color: #808080;

}

</style>

</head>

<body>

<div class="container">

<div class="row mt-4">

<div class="mr-2 col-6 col-sm-6 col-md-6 col-lg-6 col-xl-6">

<div class="primary text-center">6</div>

</div>

<div class="ml-2 col-6 col-sm-6 col-md-6 col-lg-6 col-xl-6">

<div class="primary text-center">6</div>

</div>

</div>

<div class="row mt-2">

<div class="mr-2 col-4 col-sm-4 col-md-4 col-lg-4 col-xl-4">

<div class="warning p-2 text-center">4</div>

</div>

<div class="mr-2 col-4 col-sm-4 col-md-4 col-lg-4 col-xl-4">

<div class="warning p-2 text-center">4</div>

</div>

<div class="mr-2 col-4 col-sm-4 col-md-4 col-lg-4 col-xl-4">

<div class="warning p-2 text-center">4</div>

</div>

</div>

<div class="row mt-2">

<div class="mr-2 col-8 col-sm-8 col-md-8 col-lg-8 col-xl-8">

<div class="success p-2 text-center">8</div>

</div>

<div class="mr-2 col-4 col-sm-4 col-md-4 col-lg-4 col-xl-4">

<div class="cyan p-2 text-center">4</div>

</div>

</div>

<div class="row mt-2">

<div class="mr-2 col-3 col-sm-3 col-md-3 col-lg-3 col-xl-3">

<div class="oreng p-2 text-center">3</div>

</div>

<div class="mr-2 col-3 col-sm-3 col-md-3 col-lg-3 col-xl-3">

<div class="oreng p-2 text-center">3</div>

</div>

<div class="mr-2 col-3 col-sm-3 col-md-3 col-lg-3 col-xl-3">

<div class="oreng p-2 text-center">3</div>

</div>

<div class="mr-2 col-3 col-sm-3 col-md-3 col-lg-3 col-xl-3">

<div class="oreng p-2 text-center">3</div>

</div>

</div>

<div class="row mt-2">

<div class="mr-2 col-12 col-sm-12 col-md-12 col-lg-12 col-xl-12">

<div class="gray text-center">12</div>

</div>

</div>

</div>

</body>

</html>

**::OUTPUT::**

**Icon

Description automatically generated**

1. Use a Bootstrap class to style the table properly and get the following output (with padding and horizontal dividers).
2. Add zebra-stripes to the table.
3. Add borders on all sides of the table and cells.
4. Enable a hover state on table rows.
5. Make the table more compact by cutting cell padding in half.
6. Use contextual classes to add the following:

* Green color to the table row containing John.
* Red color to the table row containing Mary.
* Orange color to the last table row.

**CODE :**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>UI</title>

<link rel="stylesheet" href="../css/bootstrap.min.css">

<script src="../js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container">

<table class="mt-4 table table-striped table-bordered table-hover table-sm">

<thead>

<tr>

<th scope="col">Firstname</th>

<th scope="col">Lastname</th>

<th scope="col">Email</th>

</tr>

</thead>

<tbody>

<tr class="table-success">

<td>John</td>

<td>Doe</td>

<td>john@example.com</td>

</tr>

<tr class="table-danger">

<td>Mary</td>

<td>Moe</td>

<td>mary@example.cpm</td>

</tr>

<tr class="table-warning">

<td>July</td>

<td>Dooley</td>

<td>july@example.com</td>

</tr>

</tbody>

</table>

</div>

</body>

</html>

**::OUTPUT::**

**Table

Description automatically generated**

1. Bootstrapping with Buttons.

Use a Bootstrap class to style the button properly with a red color.

b.      Change the size of the buttons in the following order: large, medium, small and xsmall.

c.       Make the button span the entire width of the parent element.

d.      Use a Bootstrap class to disable the button.

**CODE :**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>UI</title>

<link rel="stylesheet" href="../css/bootstrap.min.css">

<script src="../js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container">

<div class="mt-4 text-center">

<button type="button" class="col-12 col-sm-4 col-md-4 col-lg-6 col-xl-6 btn btn-danger btn-block" disabled>Danger</button>

</div>

</div>

</body>

</html>

**::OUTPUT::**

**A picture containing background pattern

Description automatically generated**

1. Style the below given html form using bootstrap to get the output shown below.

**CODE :**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Exercise #6: Simple form</title>

<link rel="stylesheet" href="../css/bootstrap.min.css">

<script src="../js/bootstrap.bundle.min.js"></script>

<style type="text/css">

.border{

border-color: #E2E2E2;

border-bottom-left-radius: 5px;

border-bottom-right-radius: 5px;

border-bottom-width: 5px;

border-top-left-radius: 5px;

border-top-right-radius: 5px;

}

.font-weight{

font-weight: bold;

}

.margin-right{

margin-right: 5px;

}

</style>

</head>

<body>

<form class="form col-12 p-4" action="#" >

<div class="mt-2">

<label class="font-weight col-12" for="first\_name">First name:</label>

<input class="mt-1 col-12 border p-1" type="text" name="first\_name" id="first\_name"/>

</div>

<div class="mt-2">

<label class="font-weight col-12" for="last\_name">Last name:</label>

<input class="mt-1 col-12 border p-1" type="text" name="last\_name" id="last\_name"/>

</div>

<div class="mt-2">

<label><input class="margin-right" type="radio" name="gender" value="male"/>male</label>

<label><input class="margin-right" type="radio" name="gender" value="female"/>female</label>

</div>

<div class="mt-2">

<label class="font-weight col-12" for="birth\_date">Date of birth:</label>

<input class="mt-1 col-12 border p-1" type="date" name="birth\_date" id="birth\_date"/>

</div>

<div class="mt-3">

<input class="btn pl-2 pr-2 btn-primary text-center" type="submit" value="Add"/>

</div>

</form>

</body>

</html>

**::OUTPUT::**

**A picture containing text

Description automatically generated**

**Tutorial-2**

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, chat or text message

Description automatically generated

Text

Description automatically generated

1. Write a MongoDB query to display all the documents in the collection restaurants.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine for all the documents in the collection restaurant.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine, but exclude the field \_id for all the documents in the collection restaurant.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Write a MongoDB query to display the fields restaurant\_id, name, borough and zip code, but exclude the field \_id for all the documents in the collection restaurant.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Write a MongoDB query to display all the restaurant which is in the borough Bronx.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Write a MongoDB query to display the first 5 restaurant which is in the borough Bronx.

**CODE : AND : OUTPUT :**

Text

Description automatically generated

1. Write a MongoDB query to display the next 5 restaurants after skipping first 5 which are in the borough Bronx.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Write a MongoDB query to find the restaurants who achieved a score more than 90.

**CODE : AND : OUTPUT :**

Text

Description automatically generated

1. Write a MongoDB query to find the restaurants that achieved a score, more than 80 but less than 100.

**CODE : AND : OUTPUT :**

Text

Description automatically generated

1. Write a MongoDB query to find the restaurants which locate in latitude value less than -95.754168.

**CODE : AND : OUTPUT :**

Text

Description automatically generated

1. Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168.

**CODE : AND : OUTPUT :**

Text

Description automatically generated

1. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than -65.754168. Note : Do this query without using $and operator.

**CODE : AND : OUTPUT :**

Text

Description automatically generated

1. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American ' and achieved a grade point 'A' not belongs to the borough Brooklyn. The document must be displayed according to the cuisine in descending order.

**CODE : AND : OUTPUT :**

Text

Description automatically generated

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'ces' as last three letters for its name.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which belong to the borough Staten Island or Queens or Bronxor Brooklyn.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which are not belonging to the borough Staten Island or Queens or Bronxor Brooklyn.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil'.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Write a MongoDB query to arrange the name of the restaurants in ascending order along with all the columns

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Write a MongoDB query to arrange the name of the cuisine in ascending order and for that same cuisine borough should be in descending order

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Find out how many times each cuisine is offered at various restaurants.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Find out how many times each cuisine is offered at various restaurants in descending order.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Which cuisine is highly offered among all restaurants?

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Find out the top 5 highly offered cuisines among all restaurants?

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

**Tutorial – 3 - CRUD with Nodejs**

1. Create and Emit a custom event that checks whether the age of the person is greater than 18 or not depending on the date of birth passed to the event.

**CODE ::**

var events = require('events')

var em = new events.EventEmitter();

em.on('ageEvent', function(data) {

var dob = new Date(data);

var mn = Date.now() - dob.getTime();

var age\_dt = new Date(mn);

var year = age\_dt.getUTCFullYear();

var age = Math.abs(year - 1997);

if (age >= 18) {

console.log("Eligible...");

} else {

console.log("Not Eligible...");

}

});

em.emit('ageEvent', "09-02-2002");

**:: OUTPUT ::**

**Text

Description automatically generated**

1. Create a node script that gets the parameters using the GET method from the form.html file and log it on the console.

**CODE ::**

**from\_get.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Get API</title>

</head>

<body>

<form method="get" action="http://localhost:3000/submit">

Name : <input type="text" name="uname"><br>

Branch : <input type="text" name="branch"><br>

<input type="submit" name="submit" value="SUBMIT">

</form>

</body>

</html>

**get.js**

var http = require('http')

var url = require('url')

var querystring = require('querystring')

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

query = url.parse(req.url).query;

uname = querystring.parse(query)['uname'];

branch = querystring.parse(query)['branch'];

console.log('Name : ', uname);

console.log('Branch : ', branch);

}).listen(3000);

**:: OUTPUT ::**

**Text

Description automatically generated with low confidence**

**Text

Description automatically generated**

1. Create a node script that gets the parameters using POST method from the form.html file and log it on console.

**CODE ::**

**🡪form\_post :**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>POST API</title>

</head>

<body>

<form method="post" action="http://localhost:3000/submit">

Name : <input type="text" name="uname"><br>

Branch : <input type="text" name="branch"><br>

<input type="submit" name="submit" value="SUBMIT">

</form>

</body>

</html>

**🡪post.js ::**

var http = require('http')

var querystring = require('querystring')

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

var dt = "";

req.on('data', function(chunk) {

dt += chunk

})

req.on('end', function() {

un = querystring.parse(dt)['uname'];

var br = querystring.parse(dt)['branch'];

console.log('Name : ', un);

console.log('Branch : ', br);

})

}).listen(3000)

**:: OUTPUT ::**

**Text

Description automatically generated**

**Text

Description automatically generated**

1. Create mongodb for students and nodejs that connects to mongodb using mongojs module.(install monojs and nodemon packages).

[Student document must contain: s\_id,s\_name, s\_branch, s\_city, s\_mobilenos, s\_add]

**CODE ::**

var mongoose = require('mongoose')

var express = require('express')

mongoose.connect('mongodb://127.0.0.1:27017/students').then(()=>{

console.log('Connection Success...');

var app = express();

app.listen(3000, ()=>{

console.log('Server Connected...');

})

}).catch((e)=>{

console.log('Connection Error...');

});

**:: OUTPUT ::**

1. Modify the above created script to insert static data in student db.

**CODE ::**

**:: OUTPUT ::**

1. Create a nodejs that fetches all the documents from student db and logs it on the console.

**CODE ::**

**:: OUTPUT ::**

1. Modify the program 6, in order to save records from the form.html file.

**CODE ::**

**:: OUTPUT ::**

1. Create a nodejs script to update student records based on the student id.

**CODE ::**

**:: OUTPUT ::**

1. Create a noejs script to delete student records based on student id.

**CODE ::**

**:: OUTPUT ::**

**Tutorial – 04 (Creating APIS using Mongoose)**

**🡺package.json**

{

"name": "demoapi",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"start": "node index",

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"body-parser": "^1.19.1",

"express": "^4.17.2",

"mongoose": "^6.2.0",

"nodemon": "^2.0.15"

}

}

**🡺index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<h1>Hosted Successfully..!!</h1> </body>

</html>

**🡺index.js**

var mongoose = require("mongoose")

var express = require("express")

var route = require('./routes')

var bodyParser = require('body-parser')

var app = express()

mongoose.connect("mongodb://127.0.0.1:27017/SPElectronics").then(() => {

console.log("Router Connected...!!")

app.use(bodyParser.urlencoded({

extended: false

}))

app.use('/api', route)

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

res.sendFile('index.html', {

root: \_\_dirname

})

})

app.listen((process.env.PORT || 3000), () => {

console.log('Server Started Successfully...!!')

})

}).catch((e) => {

console.log(e.toString())

})

**🡺 Models 🡺 Accessorie.js**

var mongoose = require('mongoose');

var accessoriesSchema = mongoose.Schema({

name: String,

price: String

})

module.exports = mongoose.model("accessories", accessoriesSchema)

Text

Description automatically generatedGraphical user interface, text, application

Description automatically generated

1. Create a NodeAPI to insert records into products database using mongoose library.

**CODE ::**

var express = require('express');

var router = express.Router();

var Accessorie = require('./Models/Accessorie')

//to insert data to the SPElectronics data base in accessories table

router.post("/accessories", async (req, res) => {

const accessorie = new Accessorie({

name: req.body.name,

price: req.body.price

})

await accessorie.save((err, msg) => {

if (err) {

res.status(500).json({

"error": err

})

} else {

res.status(200).json({

"My-message": msg

})

}

})

})

module.exports = router;

**:: OUTPUT ::**

**Graphical user interface, text, application, website

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated**

1. Create a NodeAPI to get products from the database using mongoose library.

**CODE ::**

var express = require('express');

var router = express.Router();

var Accessorie = require('./Models/Accessorie')

//to fetch data from the SPElectronics data base in accessories table

router.get('/accessories', async (req, res) => {

const accessorie = await Accessorie.find()

res.send(accessorie)

})

module.exports = router;

**:: OUTPUT ::**

**Graphical user interface, text, application, email, website

Description automatically generatedGraphical user interface, text, application, website

Description automatically generated**

1. Create a NodeAPI to update products from the database using mongoose library.

**CODE ::**

var express = require('express');

var router = express.Router();

var Accessorie = require('./Models/Accessorie')

// to update data for the SPElectronics data base in accessories table

router.patch('/accessories/:id', async (req, res) => {

const accessorie = await Accessorie.findOne({

\_id: req.params.id

})

accessorie.name = req.body.name

accessorie.price = req.body.price

await accessorie.save((err, msg) => {

if (err) {

res.status(500).json({

error: err

})

} else {

res.status(200).json({

msg: msg

})

}

})

})

module.exports = router;

**:: OUTPUT ::**

**\\||//**

**\\//**

**\/**

**Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, website

Description automatically generated**

1. Create a NodeAPI to delete products from the database using mongoose library.

**CODE ::**

var express = require('express');

var router = express.Router();

var Accessorie = require('./Models/Accessorie')

//to delete data for the SPElectronics data base in accessories table

router.delete("/accessories/:name", async (req, res) => {

await Accessorie.deleteOne({

name: req.params.name

}, (err, msg) => {

if (err) {

res.status(500).json({

error: err

})

} else {

res.status(200).json({

msg: msg

})

}

})

const accessorie = await Accessorie.find()

res.send(accessorie)

})

module.exports = router;

**:: OUTPUT ::**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, website

Description automatically generated**

**Tutorial – 05**

🡪Models🡪Accessories.js

var mongoose = require('mongoose');

var accessoriesSchema = mongoose.Schema({

name:String,

price:String

})

module.exports = mongoose.model("accessories",accessoriesSchema)

🡪index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<h1>Hosted Successfully..!!</h1>

</body>

</html>

🡪index.js

var mongoose = require("mongoose")

var express = require("express")

var route = require('./routes')

var bodyParser = require('body-parser')

var app = express()

mongoose.connect("mongodb://127.0.0.1:27017/SPElectronics", { useUnifiedTopology: true }).then(() => {

console.log("Router Connected...!!")

app.use(bodyParser.urlencoded({ extended: false }))

app.use('/api', route)

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

res.sendFile('index.html', { root: \_\_dirname })

})

app.listen((process.env.PORT || 3000), () => {

console.log('Server Started Successfully...!!')

})

}).catch((e) => {

console.log(e.toString())

})

🡪routes.js

var express = require('express');

var router = express.Router();

var Accessorie = require('./Models/Accessorie')

//to fetch data from the SPElectronics data base in accessories table

router.get('/accessories', async (req, res) => {

const accessorie = await Accessorie.find()

res.send(accessorie)

})

//to insert data to the SPElectronics data base in accessories table

router.post("/accessories", async (req, res) => {

const accessorie = new Accessorie({

name: req.body.name,

price: req.body.price

})

await accessorie.save((err, msg) => {

if (err) {

res.status(500).json({

"error": err

})

}

else {

res.status(200).json({

"My-message": msg

})

}

})

})

// to update data for the SPElectronics data base in accessories table

router.patch('/accessories/:id',async (req,res)=>{

const accessorie = await Accessorie.findOne({\_id:req.params.id})

accessorie.name = req.body.name

accessorie.price = req.body.price

await accessorie.save((err,msg)=>{

if(err){

res.status(500).json({

error:err

})

}

else{

res.status(200).json({

msg:msg

})

}

})

})

//to delete data for the SPElectronics data base in accessories table

router.delete("/accessories/:name",async(req,res)=>{

await Accessorie.deleteOne({name:req.params.name},(err,msg)=>{

if(err){

res.status(500).json({

error:err

})

}

else{

res.status(200).json({

msg:msg

})

}

})

const accessorie = await Accessorie.find()

res.send(accessorie)

})

module.exports = router;

**Tutorial – 6 (Working with Typescript)**

1. **Install typescript.**

**Under the cmd terminal you can type 🡺 npm i -g typescript**

1. **Create an arrow function that calculates the sum of n natural numbers. n is passed as a parameter.**

**CODE :**

var sum = (n: number) => {

return (n\*(n + 1) / 2)

}

console.log(sum(15));

**::OUTPUT::**

****

1. **Create three arrow functions that demonstrate usage of default parameter, optional parameter and rest parameter.**

**CODE :**

**\*\*Default Parameter\*\***

console.log("Default Parameter");

let user = (fname: string = 'Bhargav', lname: string) => {

var fristname = fname;

var lastname = lname;

console.log(fristname);

console.log(lastname);

}

user('Bk', 'patel')

**\*\*Optional Parameter\*\***

console.log("Optional Parameter");

let student = (fname: String, erno ? : number) => {

var fristname = fname;

var enrollmentno = erno;

console.log(fristname);

console.log(enrollmentno);

}

student('Bhargav', 30)

**\*\*Rest Parameter\*\***

console.log("Rest Parameter");

let staff = (fname: String, ...id: number[]) => {

var fristname = fname;

var idd = id;

console.log(fristname);

console.log(idd);

}

staff('jj', 2, 5, 8)

**::OUTPUT::**

**Text

Description automatically generated**

1. **Create an interface called student with name, city, branch properties and display method. Also, create an object to utilize the student interface.**

**CODE :**

interface student {

name: string,

city: string,

branch: string,

display: () => {}

}

var obj: student = {

name: "Bhargav",

city: "Amreli",

branch: "Computer Engineering",

display: (): string => { return "Name : " + obj.name + "\n" +

"City : " + obj.city + "\n" +

"Branch : " + obj.branch }

}

console.log(obj.display())

**::OUTPUT::**

**Text

Description automatically generated**

1. **Demonstrate the usage of single level and multiple inheritance of interface.**

**CODE :**

**//Single Inheritance**

interface base {

name: string

}

interface derived extends base {

enrollment: string,

display: () => {}

}

var student: derived = {

name: "Bhargav Vasani",

enrollment: "20SOECE13031",

display: (): string => { return "Name : " + student.name + "\n" + "Enrollment : " + student.enrollment }

}

console.log(student.display())

**//Multiple Inheritance**

interface A {

name: string

}

interface B {

enrollment: string

}

interface marks extends A, B {

CGPA: number

}

var result: marks = {

name: "Bhargav Vasani",

enrollment: "20SOECE13031",

CGPA: 8.7

}

console.log("Name : " + result.name + "\n" + "Enrollment Number: " + result.enrollment + "\n" + "CGPA : " + result.CGPA)

**::OUTPUT::**

**Text

Description automatically generated**

1. **Define a class Clock with three private integer data members hour, min and sec.**

**Define a no**

**argument constructor to initialize time value to 12:00:00.**

**Define a three argument constructor to initialize the time.**

**Define a methods to**

**a. Increment time to the next second.**

**b. Display the time value.**

**c. Return the hour (getHour():number)**

**d. Return the minute (getMinute():number)**

**e. Return the seconds (getSeconds():number)**

**CODE :**

class clock {

private hours: number

private min: number

private sec: number

constructor() {

this.hours = 12

this.min = 0

this.sec = 0

}

incSec() {

return this.sec++

}

getHour(): number {

return this.hours

}

getMinute(): number {

return this.min

}

getSeconds(): number {

return this.sec

}

Display() {

this.incSec()

console.log(this.getHour() + ":" + this.getMinute() + ":" + this.getSeconds())

}

}

var c = new clock()

c.Display();

**::OUTPUT::**

**Text

Description automatically generated**

1. **The employee interface for a company contains employee code, name, designation and basic pay. The employee is given a house rent allowance (HRA) of 10% of the basic pay and dearness allowance (DA) of 45% of the basic pay. The total pay of the employee is calculated as Basic Pay + HRA + DA. Write a class to define the details of the employee. Write a constructor to assign the required initial values. Add a method to calculate HRA, DA and total pay and print them. Create objects for three different employees and calculate HRA, DA and total pay.**

**CODE :**

**::OUTPUT::**

**Tutorial - 7**

**Create a component that displays data the from the live API created using node(refer prev tutorials).**

**Make sure to use following things:**

1. **ngFor**
2. **bootstrap (any)**
3. **Pagination**
4. **Sorting**
5. **Searching**
6. **Service**
7. **Interface**

**Code :**

🡪App.module.ts

import{ NgModule} from'@angular/core';

import{ BrowserModule} from'@angular/platform-browser';

import{ AppRoutingModule} from'./app-routing.module';

import{ AppComponent} from'./app.component';

import{ PostsComponent} from'./posts/posts.component';

import{ HttpClientModule} from'@angular/common/http';

import{NgxPaginationModule} from'ngx-pagination';

import{ OrderModule} from'ngx-order-pipe';

@NgModule({

declarations:[

AppComponent,

PostsComponent

],

imports:[

BrowserModule,

AppRoutingModule,

HttpClientModule,

NgxPaginationModule,

OrderModule

],

providers:[],

bootstrap:[AppComponent]

})

exportclassAppModule{ }

🡪Post.service.ts

import{ Injectable} from'@angular/core';

import{HttpClient} from'@angular/common/http';

@Injectable({

providedIn:'root'

})

exportclassPostService{

url= "https://jsonplaceholder.typicode.com/posts"

constructor(private\_http:HttpClient) { }

getData(){

returnthis.\_http.get(this.url);

}

}

🡪Posts.component.ts

import{ Component, OnInit} from'@angular/core';

import{ PostService} from'../Service/post.service';

@Component({

selector:'app-posts',

templateUrl:'./posts.component.html',

styleUrls:['./posts.component.css']

})

exportclassPostsComponentimplementsOnInit{

data:any= []

p: number= 1;

key= 'id';

reverse:boolean=false;

constructor(private\_postService:PostService) { }

ngOnInit(): void{

this.\_postService.getData().subscribe(res=>{

this.data=res;

console.log(this.data)

})

}

sort(key:string){

this.key= key;

this.reverse=!this.reverse;

}

}

🡪Posts.component.html

<tableclass="table">

<thead>

<th(click)="sort('id')">Id</th>

<th(click)="sort('title')">Title</th>

<th(click)="sort('body')">Body</th>

</thead>

<tbody>

<tr\*ngFor="let item of data | paginate: { itemsPerPage: 10, currentPage: p } | orderBy:key:reverse">

<td>{{item.id}}</td>

<td>{{item.title}}</td>

<td>{{item.body}}</td>

</tr>

</tbody>

</table>

<pagination-controls(pageChange)="p = $event"></pagination-controls>

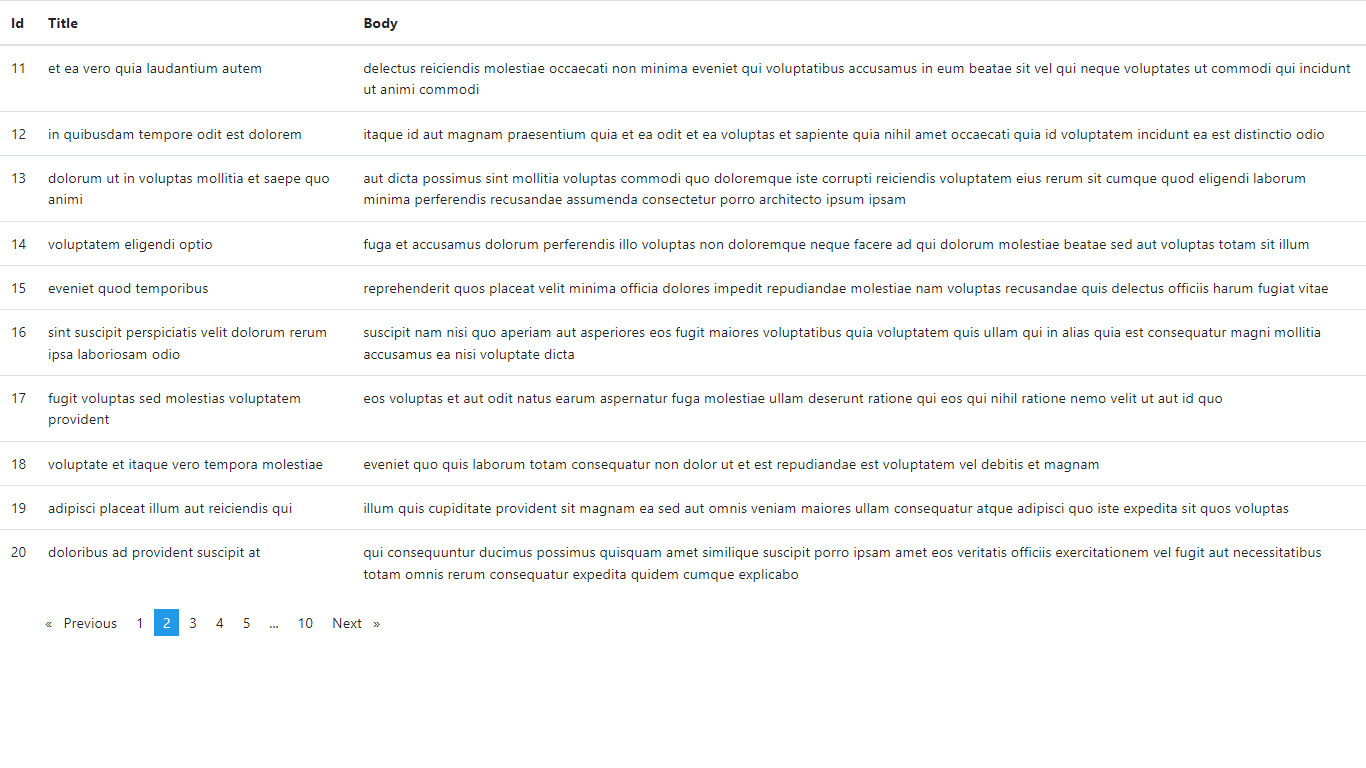
**::Output::**

**A picture containing text

Description automatically generated**

**Text

Description automatically generated**

****

**Text

Description automatically generated with medium confidence**

**Activity\_Bootstrap-4\_Portfolio**

1. Design a single page portfolio as per the below output using bootstrap 4 flexes.

**:: CODE ::**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<link rel="stylesheet" href="../css/bootstrap.min.css">

<script src="../js/bootstrap.bundle.min.js"></script>

<title>Portfolio</title>

<style>

.test {

padding: 60px 0px !important;

}

</style>

</head>

<body>

<!-- header -->

<div class="container">

<div class="row">

<div class="col-md-4 col-lg-4">

<img src="../image/bhargav.jpg" alt="John Doe" class="img-thumbnail" height="100">

</div>

<div class="col-md-8 col-lg-8">

<div class="d-flex flex-column">

<!-- name -->

<div class="bg-dark d-flex text-white justify-content-between align-items-center p-5">

<div>

<h1 class="display-4">John Aderson</h1>

</div>

<div><i class="fa fa-facebook"></i></div>

<div><i class="fa fa-google-plus"></i></div>

<div><i class="fa fa-linkedin"></i></div>

<div><i class="fa fa-youtube"></i></div>

</div>

<!-- designation -->

<div class="text-white text p-4" style="background:#000">

Web and Mobile Developer

</div>

<!-- button -->

<div class="d-flex text-white text-center">

<div class="bg-success p-4 flex-fill test">

<i class="fa fa-home"></i>

<div>Home</div>

</div>

<div class="bg-danger p-4 flex-fill test">

<i class="fa fa-home"></i>

<div>Resume</div>

</div>

<div class="bg-warning p-4 flex-fill test">

<i class="fa fa-home"></i>

<div>Work</div>

</div>

<div class="bg-primary p-4 flex-fill test">

<i class="fa fa-home"></i>

<div>Contact</div>

</div>

</div>

</div>

</div>

</div>

</div>

<div class="container">

<div class="bg-success p-3 text-white">

<h1 class="resume">My Resume</h1>

<p>It is a long established fact that a reader will be distracted by the readable content of a page when

looking at its layout.

</p>

</div>

<div class="my\_skill mt-3">

<div class="p-2">

<h3>My Skill</h3>

<p>It is a long established fact that a reader will be distracted by the readable content of a page when

looking at its layout.

</p>

</div>

<div class="p-2">

<h4>HTML</h4>

<div class="progress">

<div class="progress-bar " role="progressbar" style="width: 90%" aria-valuenow="90"

aria-valuemin="0" aria-valuemax="100">90%</div>

</div>

</div>

<div class="p-2">

<h4>CSS</h4>

<div class="progress">

<div class="progress-bar progress-bar-striped bg-danger" role="progressbar" style="width: 80%"

aria-valuenow="25" aria-valuemin="0" aria-valuemax="100">80%</div>

</div>

</div>

<div class="p-2">

<h4>Node</h4>

<div class="progress">

<div class="progress-bar bg-primary" role="progressbar" style="width: 60%" aria-valuenow="50"

aria-valuemin="0" aria-valuemax="100">60%</div>

</div>

</div>

<div class="p-2">

<h4>Angular</h4>

<div class="progress">

<div class="progress-bar bg-primary" role="progressbar" style="width: 60%" aria-valuenow="75"

aria-valuemin="0" aria-valuemax="100">60%</div>

</div>

</div>

</div>

</div>

<div class="container">

<div class="p-2">

<h3>Where I Worked?</h3>

<p>It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. </p>

</div>

<div class="worked">

<div class="row row-cols-1 row-cols-md-3 g-4">

<div class="col">

<div class="card mb-3">

<div class="card-header text-center"><strong>WEB DEVELOPER</strong></div>

<div class="card-body text-center">

<p class="card-text text-center">This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer. </p>

<p class=" bg-dark p-2 text-white">As a Front End Developer</p>

<p class=" bg-dark p-2 text-white">As a Front End Developer</p>

</div>

<div class="card-footer text-center"><strong>Date: 2015-20-18</strong></div>

</div>

</div>

<div class="col">

<div class="card mb-3">

<div class="card-header text-center"><strong>WEB DEVELOPER</strong></div>

<div class="card-body text-center">

<p class="card-text text-center">This is a longer card with supporting text below as a natural lead-in

to additional content. This content is a little bit longer.

</p>

<p class=" bg-dark p-2 text-white">As a Front End Developer</p>

<p class=" bg-dark p-2 text-white">As a Front End Developer</p>

</div>

<div class="card-footer text-center"><strong>Date: 2015-20-18</strong></div>

</div>

</div>

<div class="col">

<div class="card mb-3 text-center">

<div class="card-header text-center"><strong>WEB DEVELOPER</strong></div>

<div class="card-body">

<p class="card-text text-center">This is a longer card with supporting text below as a natural lead-in

to additional content. This content is a little bit longer.

</p>

<p class=" bg-dark p-2 text-white">As a Front End Developer</p>

<p class=" bg-dark p-2 text-white">As a Front End Developer</p>

</div>

<div class="card-footer text-center"><strong>Date: 2015-20-18</strong></div>

</div>

</div>

</div>

</div>

</div>

<div class="container">

<div class="download p-4 mb-5 bg-dark text-center">

<button class="btn btn-primary"><i class="fa fa-download"></i> Download</button>

</div>

</div>

</body>

</html>

**:: OUTPUT ::**

**Graphical user interface, application, website

Description automatically generated**

**Hands on for Querying Documents**

****

****

Text

Description automatically generated

1. Find all the result of the given collection.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Show the result in pretty format?

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Get only CE data as a output.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Get only CE data as a output with only name field.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Get the CE data without \_ID field in it.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. set the filter to “active : A” and get only the first field with “active : A” value.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Do the same as 6 question but with different method.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Do the same as 6 question but this time, get the 2nd field with active : A by skipping the 1st field.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

**Hands on with Aggregation & Grouping**

1. List the category of books.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Find the total count of documents for each category.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Find the documents having copies greater than 30 according to the categories.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated with medium confidence**

1. Find the documents by grouping according to isbn number and sort by the same.

**CODE : AND : OUTPUT :**

**Graphical user interface, text

Description automatically generated**

1. Find the documents by grouping to categories and sorting it by category name only for those documents having available status true.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Find the average no. of books based on the categories.

**CODE : AND : OUTPUT :**

Text

Description automatically generated

1. Find the min and max no. of books based on the categories.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Find the total count of documents for each category.

**CODE : AND : OUTPUT :**

A picture containing text

Description automatically generated

1. Find the total count of not available books for each category.

**CODE : AND : OUTPUT :**

Graphical user interface, text

Description automatically generated

1. Find the total no. of copies of books for each category.

**CODE : AND : OUTPUT :**

Text

Description automatically generated

1. Sort the above output in descending order of copies.

**CODE : AND : OUTPUT :**

Text

Description automatically generated

1. Find the documents by grouping category and isbn number along with the count of each group.

**CODE : AND : OUTPUT :**

Text

Description automatically generated

1. Find the documents by grouping category and isbn number along with the count of each group. But the output should be displayed as follows:

Expected output:

{ "Category" : "history", "ISBN" : 32145, "total\_docs" : 1 }

{ "Category" : "history", "ISBN" : 1234, "total\_docs" : 2 }

{ "Category" : "fiction", "ISBN" : 1234, "total\_docs" : 1 }

{ "Category" : "programming", "ISBN" : 1234, "total\_docs" : 1 }

{ "Category" : "programming", "ISBN" : 12354, "total\_docs" : 1 }

{ "Category" : "gk", "ISBN" : 1234, "total\_docs" : 1 }

{ "Category" : "novel", "ISBN" : 1234, "total\_docs" : 1 }

**CODE : AND : OUTPUT :**

A screen shot of a computer

Description automatically generated with medium confidence

1. Find the documents by grouping category and isbn number along with the total no of copies for each group. Display the group having highest copies.

Expected output:

{ "Category" : "history", "ISBN" : 32145, "total\_copies" : 50 }

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

**Hands-on with Node.js**

1. Install Nodejs to set up the project.

Follow the step : 🡺

Step-1 : Frist off all you can download a NodeJS set up file from the browser follow this link : [Download | Node.js (nodejs.org)](https://nodejs.org/en/download/) or [**https://nodejs.org/dist/v14.15.4/node-v14.15.4-x64.msi**](https://nodejs.org/dist/v14.15.4/node-v14.15.4-x64.msi)

Step-2 : Ones download this file than install the NodeJS in your system and than after set environment variable path as a globally from any location.

EX 🡪 C:\Program Files\MongoDB\Server\4.4\bin

1. Create a function expression in node js to count the number of items in an array.

**CODE ::**

var arr = [10,20,30,40,50];

console.log(arr.length)

**:: OUTPUT ::**

****

1. Create a module that contains three functions and export the same using nodejs.

**CODE ::**

var sum = (a, b) => {

return a + b;

}

var print = (s) => {

return s;

}

var sqr = (a) => {

return a \* a;

}

console.log("Let's Start..!")

module.exports = {

sum,

print,

sqr

};

**:: OUTPUT ::**

****

1. Create a module to import the above created function and utilize it.

**CODE ::**

var fun = require('./function\_export')

console.log(fun.sum(9, 17));

console.log(fun.print("Welcome to Node...!!!"));

console.log(fun.sqr(7));

**:: OUTPUT ::**

**Text

Description automatically generated**

1. Create a server using http module and listen to port 3000.

**CODE ::**

var http = require('http')

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

res.writeHead(200, {

"Content-Type": "text/html"

});

res.write("Welcome to Http Module...!!!")

res.end();

})).listen(3000);

console.log("Server Started...!!!");

**:: OUTPUT ::**

****

1. Display three different users(Student, admin and default) html pages as per the requested url.

**CODE ::**

var http = require('http')

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

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

res.writeHead(200, {

"Content-Type": "text/html"

});

res.write("Welcome Default Page...!!!")

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

res.writeHead(200, {

"Content-Type": "text/html"

});

res.write("Welcome Student Page...!!!")

} else {

res.writeHead(200, {

"Content-Type": "text/html"

});

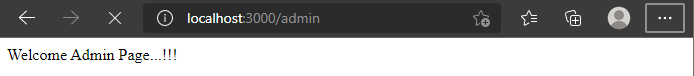
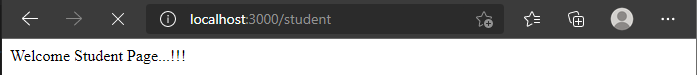
res.write("Welcome Admin Page...!!!")

}

})).listen(3000);

console.log("Server Started...!!!");

**:: OUTPUT ::**

****

1. Create and import the class type object to display the full name of the user.

**CODE ::**

**name.js**

var name = {

fname: "Bhargav",

lname: "Vasani"

};

module.exports = name;

**import\_class.js**

var http = require('http')

var result = require('./name')

var url = require('url')

var querystring = require('querystring')

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

var query = url.parse(req.url).query;

var fname = querystring.parse(query)['fname'];

var lname = querystring.parse(query)['lname'];

console.log(result.fname, result.lname)

console.log("First name", result.fname)

console.log("Last name", result.lname)

})).listen(3000);

console.log("Server Started...!!!");

**:: OUTPUT ::**

**Text

Description automatically generated**

**Hands on with Querying using Regex Expression**

Text

Description automatically generated

Text

Description automatically generated

1. Find the documents whose description starts with "M".

**CODE : AND : OUTPUT :**

**A screenshot of a computer

Description automatically generated with medium confidence**

1. Find the documents whose description end with "n".

**CODE : AND : OUTPUT :**

****

1. Find the documents whose description contains "line" word".

**CODE : AND : OUTPUT :**

**A screenshot of a computer

Description automatically generated with medium confidence**

1. Find the documents whose description contains second character "i".

**CODE : AND : OUTPUT :**

****

1. Find the documents where "sku" fields contains "xyz", ignore the case sensitivity.

**CODE : AND : OUTPUT :**

****

1. Find the documents where any line from the description starts with ‘S’.

**CODE : AND : OUTPUT :**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Hands-On for Querying with filters**

1. Create a database named "Stocks".

**CODE : AND : OUTPUT :**

****

1. Create an empty collection named "inventory".

**CODE : AND : OUTPUT :**

****

1. Insert below records in the inventory collection all together.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Select all documents from inventory collection.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Selects from the inventory collection all documents where the status equals "D".

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Retrieves all documents from the inventory collection where status equals either "A" or "D".

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Retrieve documents in the inventory collection where the status equals "A" and qty is less than ($lt (Links to an external site.)Links to an external site.) 30.

**CODE : AND : OUTPUT :**

****

1. Retrieve all documents in the collection where the status equals "A" or qty is less than ($lt (Links to an external site.)Links to an external site.) 30.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Update qty to 25 where item is notebook

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Update all status to 'B' where status is 'D'.

**CODE : AND : OUTPUT :**

A screenshot of a computer

Description automatically generated with medium confidence

1. Write a query to demonstrate usage of upsert.

**CODE : AND : OUTPUT :**

Text

Description automatically generated

1. Delete the document where qty is 100

**CODE : AND : OUTPUT :**



1. Delete all the documents from inventory collection.

**CODE : AND : OUTPUT :**



1. Rename the collection.

**CODE : AND : OUTPUT :**

Text

Description automatically generated

1. Delete the database.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

**Practice Basic CRUD**

1. Create a database named "Students".

**CODE : AND : OUTPUT :**

****

1. Create an empty collection named "studentData".

**CODE : AND : OUTPUT :**

****

1. Insert only one record with appropriate fields.

**CODE : AND : OUTPUT :**

****

1. Try to insert 5 records together using single query.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Fetch all the documents in formatted manner.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Fetch the document based on the filter criteria.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Delete one document from the "studentData".

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Delete the documents based on the filter criteria.

**CODE : AND : OUTPUT :**

**Text

Description automatically generated**

1. Delete all the documents from the "studentData: collection

**CODE : AND : OUTPUT :**

**A screenshot of a computer

Description automatically generated with medium confidence**

1. Drop the collection.

**CODE : AND : OUTPUT :**

** Graphical user interface, text

Description automatically generated**

**Practice with events and string interpolation**

1. Create a variable, array, function, object and an array of objects in the class file and utilize the same in html to display the data.

**CODE : AND : OUTPUT :**

* First off all create one new project using 🡪 ng new project name
* Than create component using 🡪 ng g c component name
* I create display-data component…

🡪app.component.html

<app-display-data></app-display-data>

<router-outlet></router-outlet>

🡪display-data.component.ts

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-display-data',

templateUrl: './display-data.component.html',

styleUrls: ['./display-data.component.css']

})

export class DisplayDataComponent implements OnInit {

public Students = [

{ name: "Aryan", branch: 'CE', rollno: "07" },

{ name: "Bhargav", branch: 'IT', rollno: "37" },

{ name: "Dhrumil", branch: 'BBA', rollno: "22" },

{ name: "Maulik", branch: 'CE', rollno: "10" },

{ name: "Nayan", branch: 'CV', rollno: "45" }

]

constructor() { }

ngOnInit(): void {

}

printStudentName(student:any) { return student.name; }

printStudentBranch(student:any) { return student.branch; }

printStudentRollno(student:any) { return student.rollno; }

}

🡪display-data.component.html

<h1>Student Details : </h1>

<div \*ngFor="let student of Students">

<h4>Name : {{ printStudentName(student) }}</h4>

<h4>Branch : {{ printStudentBranch(student) }}</h4>

<h4>Roll.No. : {{ printStudentRollno(student) }}</h4>

<hr>

</div>

A picture containing chart

Description automatically generated

1. Demonstrate the usage of following events with appropriate example in the existing project:

* Click MouseOver
* MouseLeave
* Keyup
* Keydown
* Keyup + enter
* Also, Provide the text box to enter username and display it in alert box when user Clicks on "Show" button.

**CODE : AND : OUTPUT :**

* I create event component…

🡪 app.component.html

<app-event></app-event>

<router-outlet></router-outlet>

🡪 event.component.html

<h1>Events</h1>

<b><p>Click Event</p></b>

<button (click)="onClick('Click Event')">submit</button>

<hr />

<b><p>Mouse Over Event</p></b>

<button (mouseover)="onClick('Mouse Over')">Mouse Over Event</button>

<hr />

<b><p>Mouse Leave Event</p></b>

<button (mouseleave)="onClick('Mouse Leave')">Mouse Leave Event</button>

<hr />

<b><p>Key Up Event</p></b>

<input type="text" placeholder="Key Up" #kup (keyup)="onClick(kup.value)" />

<hr />

<b><p>Key Down Event</p></b>

<input type="text" placeholder="Key Down" #kdown (keydown)="onClick(kdown.value)" />

<hr />

<b><p>Key Up + Enter Event</p></b>

<input type="text" placeholder="Key Up Enter" #kupenter (keyup.enter)="onClick(kupenter.value)" />

<hr />

<b><p>Enter Username</p></b>

<input type="text" placeholder="Enter Username" #box />

<button (click)="onClick(box.value)">Show</button>

<div class="alert alert-warning" role="alert">

<h3>{{uname}}</h3>

</div>

🡪 event.component.ts

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-event',

templateUrl: './event.component.html',

styleUrls: ['./event.component.css']

})

export class EventComponent implements OnInit {

uname = "";

onClick(val: any) {

console.log(val);

this.uname = val;

}

constructor() { }

ngOnInit(): void {

}

}

🡪Click event, Mouse Over event, Mouse Leave event

Graphical user interface, text, application

Description automatically generated

🡪 Key Up Event



🡪 Key Down Event

Graphical user interface, application

Description automatically generated

🡪key Up + Enter Event



🡪 and display it in alert box when the click show button.

Waterfall chart

Description automatically generated with medium confidence