F – 55041

Register No. | | | | | | | | |

BTech Degree Examination December 2022

Fourth Semester

Information Technology

20ITT44 –WEB TECHNOLOGY

(Regulations 2020)

Time: Three hours | Maximum: 100 marks

Answer all Questions

Part – A (10 × 2 = 20 marks)

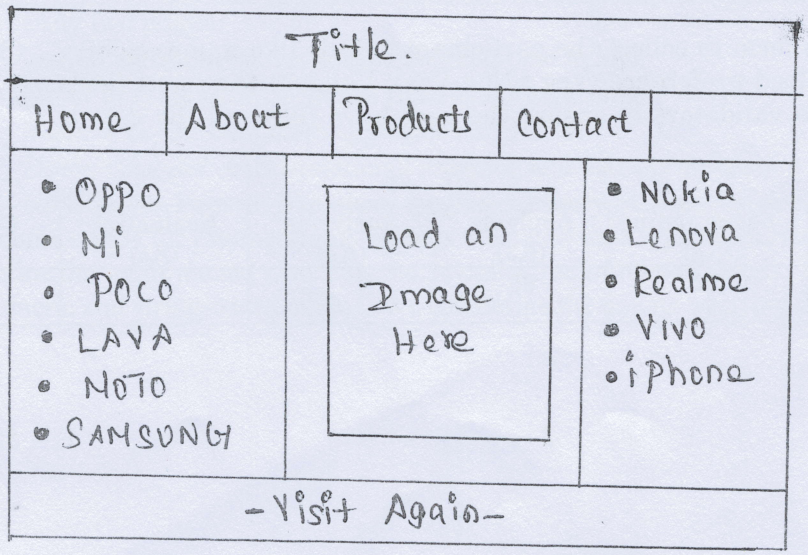
1. Write the html code to display five district names in unordered list. [CO1,K3]
2. Write syntax to create an button using bootstrap. [CO1,K3]
3. Differentiate between let and var keywords. [CO2,K2]
4. What is collections in JavaScript? [CO2,K1]
5. Differentiate between get and post method. [CO3,K2]
6. Specify the features of NOSQL database. [CO3,K1]
7. Mention the use of Enum data type. Give an example. [CO4,K1]
8. Identify the appropriate pipes to format the given text in display. [CO4,K2]
9. What is the role of dependency injection in angular? [CO5,K1]
10. Differentiate between template driven forms and reactive forms in angular. [CO5,K2]

Part – B (5 × 16 = 80 marks)

11. a. Create a website for the technical event "ERO-EXPO". The event will held on 10.07.2022. Use the different types of CSS to design the website. Also use images, list, tables and hyper link to enhance the presentation. (16) [CO1,K3]

(OR)

b. Write the code to design the sample structure given below using bootstrap. (16) [CO1,K1]



12. a. i) Explain the looping statements supported by JavaScript. (8) [CO2,K1]

ii) Create an form with following fields name, email, phone and gender. Use the JavaScript code to validlate the data enter by the user. Ensure the user need to fill all the details. (8) [CO2,K3]

(OR)

b. i) Explain the following JavaScript event: (8) [CO2,K1]

1) onmouse over
2) onmouse out
3) onfocus
4) onblur

ii) Create an menu driven calculator to perform basic arithmetic operations like addition, subtraction, multiplication and division. (8) [CO2,K3]

13. a. Write the code to create an webserver using Node JS. Ensure the server will respond to the request based on the URL. (16) [CO3,K3]

(OR)

b. A car service provider would like to collect the feedback from the customer after the service via website. Can you develop an web application to collect the information like name, city, mobile, email, feedback and service rating from the customer and store it in the mongodb database. Provide the report option to view the feedback in webpage to the provider. (16) [CO1,K1]

14. a. i) Outline the different types of functions supported by typescript with example. (8) [CO4,K2]

ii) Write the typescript program to manage the employee information using classes. (8) [CO4,K3]

(OR)

b. i) Elaborate the different data binding methods supported by angular with example. (8) [CO4,K2]

ii) Write the step by step procedure to create nested components in Angular. (8) [CO4,K3]

15. a. i) What is a service in Angular? Explain the procedure to create an service in the Angular project. (8) [CO5,K2]

ii) Create an template driven form to collect the student details like name roll no, branch, year of study, department and college. Ensure the student has filled all the details. (8) [CO5,K3]

(OR)

b. i) Outline the procedure to switch between the pages using the routing in Angular. (8) [CO5,K2]

ii) Create a reactive form to collect the participant details like name, mobile, city, gender and food preference (Veg / Non Veg). Validate the input fields using appropriate validators to ensure the user have filled all the details properly. (8) [CO5,K3]

| Bloom's Taxonomy Level | Remembering (K1) | Understanding (K2) | Applying (K3) | Analysing (K4) | Evaluating (K5) | Creating (K6) |
|---|---|---|---|---|---|---|
| Percentage | 13 | 22 | 65 | - | - | - |

**B.Tech Degree Exam Dec - 2022**

Fourth Semester

Information Technology

**20ITT44 WEB TECHNOLOGY**

(R-2020)

Part – A (10 x 2 =20 Marks)

1. (2)

```
<ul>
  <li>Erode</li>
  <li>Chennai</li>
  <li>Covai</li>
  <li>Tiruppur</li>
  <li>Salem</li>
</ul>
```

2. (2)

```
<a href="#" class="btn btn-info" role="button">Link Button</a>
(or)
<button type="button" class="btn btn-info">Button</button>
(or)
<input type="button" class="btn btn-info" value="Input Button">
```

3. **Let vs Var (Any two points)** (2)

| let | var |
|---|---|
| let is block-scoped. | var is function scoped. |
| let does not allow to redeclare variables. | var allows to redeclare variables. |
| Hoisting does not occur in let. | Hoisting occurs in var. |

4. `HTMLCollection` object is an array-like list (collection) of HTML elements. (2)

```
const myCollection = document.getElementsByTagName("p");
```

The elements in the collection can be accessed by an index number.

To access the second <p> element you can write:

```
myCollection[1]
```

5. **GET vs POST (any two points)** (2)

| GET | POST |
|---|---|
| • GET requests can be cached<br>• GET requests remain in the browser history<br>• GET requests can be bookmarked<br>• GET requests should never be used when dealing with sensitive data<br>• GET requests have length restrictions<br>• GET requests are only used to request data (not modify) | • POST requests are never cached<br>• POST requests do not remain in the browser history<br>• POST requests cannot be bookmarked<br>• POST requests have no restrictions on data length |

6. (Any two points) (2)

- NoSQL databases never follow the relational model
- Never provide tables with flat fixed-column records
- Work with self-contained aggregates or BLOBs
- Doesn't require object-relational mapping and data normalization
- No complex features like query languages, query planners,referential integrity joins, ACID

7. Enums stands for Enumerations. Enums are a new data type supported in TypeScript. It is used to define the set of named constants, i.e., a collection of related values. TypeScript supports both numeric and string-based enums. We can define the enums by using the enum keyword. (2)

```
enum Direction {
   Up = 1,
   Down,
   Left,
   Right,
}
```

8. (2)

It takes integers, strings, arrays, and date as input separated with **|** to be converted in the format as required and display the same in the browser.

```
{{ Welcome to Angular 4 | lowercase}}
```

9. Dependency injection, or DI, is one of the fundamental concepts in Angular. DI is wired into the Angular framework and allows classes with Angular decorators, such as Components, Directives, Pipes, and Injectables, to configure dependencies that they need. (2)

10. **Any two points** (2)

| Reactive Forms | Template-driven Forms |
|---|---|
| More robust. | Add a simple form to your application |
| More scalable, reusable, and testable | Not as scalable as reactive forms |
| Preferred to use if forms are a key part of your application, or your application is already built using reactive patterns | Mainly used if your application's requires a very basic form and logic. It can easily be managed in a template. |
| Reactive forms are more explicit. They are created in component class. | Template-driven forms are less explicit. They are created by directives. |
| Structured | Unstructured |
| Synchronous | Asynchronous |
| Functions | Directives |
| Immutable | Mutable |
| Low-level API access | Abstraction on top of APIs |

PART - B

11.a. **Website Using CSS** (16)

```
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
<link rel="stylesheet" href="styles.css">
</head>
<body>
<h1 style="color:blue;">A Blue Heading</h1>

<p style="color:red;">A red paragraph.</p>
<a href="url">link text</a>
<img src="pic_trulli.jpg" alt="Italian Trulli">
<table>
```

```
  <tr>
    <th>Company</th>
    <th>Contact</th>
    <th>Country</th>
  </tr>
  <tr>
    <td>Alfreds Futterkiste</td>
    <td>Maria Anders</td>
    <td>Germany</td>
  </tr>
 <ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>

</table>

</body>
</html>
```

styles.css

```
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

11.b **Website design using** (16)

```
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css">
  <script src="https://cdn.jsdelivr.net/npm/jquery@3.6.1/dist/jquery.slim.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>

<div class="container">
  <div class="jumbotron">
    <h1>Title of the Website</h1>
     </div>
 </div>

<nav class="navbar navbar-expand-sm bg-light">
  <ul class="navbar-nav">
    <li class="nav-item">
      <a class="nav-link" href="#">Link 1</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Link 2</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Link 3</a>
```

```
    </li>
  </ul>
</nav>

<div class="container-fluid">
  <div class="row">
    <div class="col-md-3" style="background-color:lavender;"> <ul>
  <li>Oppo</li>
  <li>Mi</li>
  <li>POCO</li>
</ul>
</div>
 <div class="col-md-6" style="background-color:orange;"><img src="pic_trulli.jpg" alt="Italian Trulli"></div>
 <div class="col-md-3" style="background-color:lavender;">
<ul>
  <li>Nokia</li>
  <li>Lenova</li>
  <li>Realme</li>
</ul>
</div>
</div>
</div>

<div class="row">
  <div class="col-md-12">Visit Again</div>
</div>
</body>
</html>
```

12.a.i **Looping in JS (Consider any four) 2 x 4 = 8 Marks** (8)

**for - loops through a block of code a number of times**

```
for (let step = 0; step < 5; step++) {
  // Runs 5 times, with values of step 0 through 4.
  console.log("Walking east one step");
}
```

**while loop loops through a block of code as long as a specified condition is true.**

```
while (condition) {
  // code block to be executed
}
```

**for in statement loops through the properties of an Object:**

```
const person = {fname:"John", lname:"Doe", age:25};

let text = "";
for (let x in person) {
  text += person[x];
}
```

**for of statement loops through the values of an iterable object.**

```
for (variable of iterable) {
  // code block to be executed
}
```

**do/while - also loops through a block of code while a specified condition is true**

```
let i = 0;
do {
  i += 1;
  console.log(i);
} while (i < 5);
```

12.a.ii **Form Validation:** (8)

```
<html>
<head>
<script type="text/javascript">
function validation()
{
var a = document.form.name.value;
if(a=="")
{
alert("Please Enter Your Name");
document.form.name.focus();
return false;
}

var a = document.form.email.value;
if(a=="")
{
alert("Please Enter Your Password");
document.form.email.focus();
return false;
}

var a = document.form.phone.value;
if(a=="")
{
alert("Please Enter Your Phone Number");
document.form.phone.focus();
return false;
}

if ( ( form.gender[0].checked == false ) && ( form.gender[1].checked == false ) )
{
alert ( "Please choose your Gender: Male or Female" );
return false;
}

}
</script>
</head>
<body>
<form name="form" method="post" onsubmit="return validation()">
<tr>
<td> Name:</td>
<td><input type="text" name="name""></td>
</tr>
<tr>
<td> Email:</td>
<td><input type="text" name="email""></td>
</tr>
<tr>
<td> Phone:</td>
<td><input type="text" name="phone""></td>
</tr>
Your Gender: <input type="radio" name="gender" value="Male"> Male
```

```
<input type="radio" name="gender" value="Female"> Female
<tr>
<td></td>
<td><input type="submit" name="sub" value="Submit"></td>
</tr>
</form>
</body>
</html>
```

12.b.i **Onmouseover, onmouseout, onfocus and onblur** (8)

```
<html>
  <head>
    <script>
        function sayHello() {
           alert("Mouse Over")
         }
     function sayHello1() {
           alert("Mouse Out")
        }

 function myFunction() {
  let x = document.getElementById("fname");
  x.value = x.value.toUpperCase();
}

function myFunction2(x) {
  x.style.background = "yellow";
}
    </script>
  </head>
<body>
<p onmouseover = "sayHello()">mouseover event.</p>

<p onmouseout = "sayHello1()">mouseout event.</p>

<input type="text" id="fname" onblur="myFunction()">

<input type="text" onfocus="myFunction2(this)">
  </body>
</html>
```

12.b.ii **Calculator:** (8)

```
<!DOCTYPE html>
<html>
<body>

 <script>

   let result;

   const operator = prompt('Enter operator ( either +, -, * or / ): ');
   const number1 = parseFloat(prompt('Enter first number: '));
   const number2 = parseFloat(prompt('Enter second number: '));

   switch(operator) {
     case '+':
     result = number1 + number2;
     console.log(`${number1} + ${number2} = ${result}`);
     break;

     case '-':
```

```
    result = number1 - number2;
    console.log(`${number1} - ${number2} = ${result}`);
    break;

    case '*':
    result = number1 * number2;
    console.log(`${number1} * ${number2} = ${result}`);
    break;

    case '/':
    result = number1 / number2;
    console.log(`${number1} / ${number2} = ${result}`);
    break;

    default:
    console.log('Invalid operator');
    break;
  }
 </script>

</body>
</html>
```

13.a **NODE JS SERVER** (16)

Node JS URL Handling

```
var http=require('http');
http.createServer(function(req,res){
 if(req.url=="/admin")
 {
  res.writeHead(200,{'Content-type':'text/plain'});
  res.end('Admin Page');
 }
 else if (req.url=="/student")
  {
   res.writeHead(200,{'Content-type':'text/html'});
            res.write('<html><body>');
            res.write('<h1>Student Page</h1>');
            res.write('</body></html>');
   res.end();
  }
 else {
  res.end('This is wrong url');
 }

}).listen(3000);
```

13.b. **HTML Form Code:** (16)

```
<html>
<body>
<form method="post">
        Customer  name<input type="text" name="customer_name" value=""/><br/>
        City<input type="text" name="city" value=""/><br/>

        Mobile<input type="text" name="mno" /><br/>

        Email<input type="text" name="email" /><br/>

       Feedback<input type="text" name="feedback" value=""/><br/>
      Rating<input type="text" name="rating" value=""/><br/>
```

```
        <input type="submit" name="show" value="Show"
                                formaction="http://localhost:3000/show"/>
        <input type="submit" name="login" value="Save"
                                formaction="http://localhost:3000/save"/>

</form>
</body>
</html>
```

**Mongo DB Processing:**

```
var MongoClient=require('mongodb').MongoClient;
var url='mongodb://127.0.0.1:27017/';
console.log("MongoDB");

exports.saveData= function (name,city,mno,email,feedback,rating, response) {

        MongoClient.connect(url,function(err,db){ //Connection to server
        if(err) throw err;
        var dbcon=db.db('WTdemo');  //opening the db
        var msg="";
        var myobj = {"customer_name":name ,"city":city,"email":email,
"mno":mno,"rating":rating,"feedback":feedback};
        dbcon.collection("feedback").insertOne(myobj,function(err,res){
                if (err)
                {
                        console.log(err);
                        msg="Feedback Not inserted";
                }
                else
                {
                        msg="  *** Feedback Inserted***";
                        console.log("Document inserted");
                }

                response.write(msg);
                response.end();
                db.close();
        });

        });

 };

 exports.showData= function ( response) {

        MongoClient.connect(url,function(err,db){ //Connection to server
        if(err) throw err;
        var dbcon=db.db('WTdemo');  //opening the db
        var msg="";


        dbcon.collection("books").find().toArray(function(err, result) {
                if (err)
                {
                        console.log(err);
                        msg="Error!!!";
                }
                else
                {
```

```
            var Length = result.length;
            msg="<table><tr><td>S.No</td><td>
Name</td><td>City</td>><td>Mobile</td>><td>Feedback</td><td>Rating</td></tr>";
          for(var i=0; i<Length; i++)
            {
            msg+="<tr><td>"+(i+1)+"</td><td>"+result[i].customer_name+"</td><td>"+result[i].city+
     "</td><td>"+result[i].mno+"</td><td>"+result[i].feedback+"</td><td>"+result[i].rating +"</td></tr>";

            }
            msg+="</table>";
                  }
                  response.write(msg);
                  response.end();
                  db.close();

             });


            });

 };
```

**Server Program:**

```
var module = require('./db_module');
var url = require('url');
var querystring = require('querystring');
var http = require('http');

http.createServer(function(request, response) {
var data1 = '';

request.on('data', function(chunk) {
          data1 += chunk;
       });

request.on('end', function() {
var customer_name = querystring.parse(data1)["customer_name"];
var city = querystring.parse(data1)["city"];
var mno = querystring.parse(data1)["mno"];
var feedback = querystring.parse(data1)["feedback"];
var email = querystring.parse(data1)["email"];
var rating = querystring.parse(data1)["rating"];


if (request.url === '/show') {
module.showData( response);
          }
 else if (request.url === '/save') {
module.saveData(customer_name, city,mno,email,feedback,rating, response);
          }
     });

}).listen(3000);
console.log("Server started");
```

14.a.i **Typescript Functions: (Any 4 types)** (8)

```
function greetings() {
   console.log("Hello World");
```

```
}

function Greet(greeting: string, name: string ) : string {
   return greeting + ' ' + name + '!';
}

Optional Parameter:
function Greet(greeting: string, name?: string ) : string {
   return greeting + ' ' + name + '!';
}

Arrow functions:
let sum1 = (x: number, y: number): number => {
   return x + y;
}

console.log(sum1(100, 20)); //returns 120

function addition(x, y) {
   return x + y;
}
// calling
console.log("Sum: 1 + 2 = " + addition(1, 2));
greetings();

Anonymous:
var msg = function() {
  return "hello world";
}
console.log(msg())

Lambda Functions:
var foo = (x:number)=>10 + x
console.log(foo(100))      //outputs 110
```

14.a.ii

```
class Employee {
   //defining fields
   id: number;
   name:string;
   desig:string;
   branch:string;

   //creating method or function
   display():void {
      console.log("Employee ID is: "+this.id)
      console.log("Name is: "+this.name)
      console.log("Designation: "+this.desig)
      console.log("Branch: "+this.branch)
   }
}

//Creating an object or instance
let obj = new Employee();
obj.id = 301;
obj.name = "ABCD";
obj.desig="Sales";
obj.branch="Erode";
obj.display();
```

(8)

14.b.i Data binding is a very useful and powerful feature used in software development technologies. It acts as a bridge (8)

between the view and business logic of the application.

AngularJS follows Two-Way data binding model.

One-Way Data Binding

The one-way data binding is an approach where a value is taken from the data model and inserted into an HTML element. There is no way to update model from view. It is used in classical template systems. These systems bind data in only one direction.

```
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
   <p>First name: {{firstname}}</p>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
   $scope.firstname = "John";
   $scope.lastname = "Doe";
});
</script>
</body>
</html>
```

Two-Way Data Binding

Data-binding in Angular apps is the automatic synchronization of data between the model and view components.

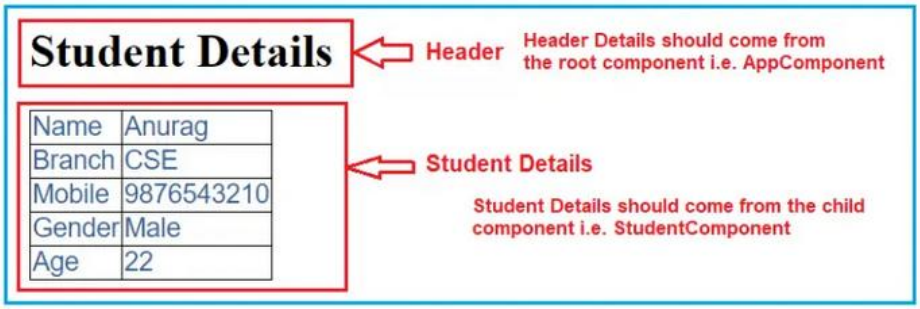
Data binding lets you treat the model as the single-source-of-truth in your application. The view is a projection of the model at all times. If the model is changed, the view reflects the change and vice versa.

```
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="" ng-init="firstName='Ajeet'">
<p>Input something in the input box:</p>
<p>Name: <input type="text" ng-model="firstName"></p>
<p>You wrote: {{ firstName }}</p>
</div>
</body>
</html>
```

14.b.ii The Angular framework allows us to use a component within another component and when we do so then it is called Angular Nested Components. The outside component is called the parent component and the inner component is called the child component. (8)

**Steps to create two angular components**

Create a new project using the following command
ng new MyAngularApp
**AppComponent**: The AppComponent is used to display the page header. This is our parent component.
**StudentComponent**: The StudentComponent is used to display the Student details. This is our child component. As this is the child component, it should be nested inside the AppComponent i.e. parent component.
The following diagram gives you a clear idea about the above two points.




Step1: Creating student component
Open Visual Studio Code terminal and type ng g c student and press the enter button as shown in the below image.

Step2: Modifying Student.component.html file:

This file is going to contain the HTML for the student component.

```
<table>
<tr>
<td>Name</td>
<td>{{Name}}</td>
</tr>
<tr>
<td>Branch</td>
<td>{{Branch}}</td>
</tr>
<tr>
<td>Mobile</td>
<td>{{Mobile}}</td>
</tr>
<tr>
<td>Gender</td>
<td>{{Gender}}</td>
</tr>
<tr>
<td>Age</td>
<td>{{Age}}</td>
</tr>
</table>
```

Step3: Modifying the student.component.ts file:
Within the student folder, you can find a file with the name student.component.ts within which the StudentComponent is created.

```
import { Component } from '@angular/core';
@Component({
selector: 'app-student',
templateUrl: './student.component.html',
styleUrls: ['./student.component.css']
})
export class StudentComponent {
Name: string = 'Anurag';
Branch: string = 'CSE';
Mobile: number = 9876543210;
Gender: string = 'Male';
Age: number = 22;
}
```

Step3: Modifying the App.Component.html file
Now, open app.component.html file

```
 div>
```

```
<h1>{{pageHeader}}</h1>
<app-student></app-student>
</div>`
```

Running Angular Application:
In order to compile and run the angular application with your default browser,
type **ng serve -o** and press the enter key as shown below.

15.a.i Service is a broad category encompassing any value, function, or feature that an application needs. A service is typically a class with a narrow, well-defined purpose. It should do something specific and do it well. (8)

Steps to Create Service:

Step 1: Create a new project using the following command
ng new demoservice
This will create a new project called demoservice

Step 2: Run the project using the following command
ng serve –open

Step 3: Create a new service class using the following command
ng g service myservice
The above command create two files in src/app/ folder
myservice.service.spec.ts
myservice.service.ts

Step 4: import the service file into the app.module.ts file.
import { MyserviceService } from './myservice.service';
Update the providers array with service name
providers: [MyserviceService]

Ste 5: Add business logic inside myservice.service.ts file

```
export class MyserviceService {
  dept= “Kongu Engg”;
  constructor() { }
  getdepartment()
  {
    return this.dept;
  }
}
```

Step 6: Use the service in component file
In the app.component.ts import the service
import { MyserviceService } from './myservice.service';

To access the service

```
mydept:string;
  constructor(private myservice:MyserviceService)
  {
    this.mydept=myservice.getdepartment();

  }
```

Step 7: Add the code in app.component.html

```
{{mydept}}
```

15.a.ii Template Driven-Form(app.component.html-normal ngmodel html) (8)

```
<form (ngSubmit)="login(form1)" #form1="ngForm">
   <div class="form-row">

     <div class="form-group col-md-6">
       <label for="inputEmail4">Name</label>
```

```
      <input type="text" class="form-control" id="inputName" placeholder="Name"  required ngModel>
    </div>

    <div class="form-group col-md-6">
      <label for="inputEmail4">Roll No</label>
      <input type="text" class="form-control" minlength="10"   id="inputRollno" placeholder="RollNo" required
ngModel>
    </div>

    <div class="form-group col-md-6">
      <label for="inputEmail4">Branch</label>
      <input type="text" class="form-control" id="inputBranch" placeholder="Branch" required ngModel>
    </div>

    <div class="form-group col-md-6">
      <label for="inputPassword4">Year of study</label>
      <input type="yos" class="form-control"  id="inputYos" ngModel required placeholder="Yos">
    </div>
  </div>

    <div class="form-group col-md-6">
      <label for="inputPassword4">Department </label>
      <input type="department" class="form-control" id="inputDept" ngModel  required placeholder="Yos">
    </div>
  </div>

    <div class="form-group col-md-6">
      <label for="inputCollege">College</label>
      <input type="college" class="form-control" id="inputCollege" ngModel required placeholder="Yos">
    </div>
  </div>

      <button type="submit" class="btn btn-primary">Sign in</button>
      </form>
```

App.component.ts

```
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

    login(myform:NgForm)
    {
      alert("success")
    }
}
```

Appmodule.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';

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

@NgModule({
```

```
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app.component.css

```
.ng-invalid .ng-touched
{
   border-color:red;
}
.error-block
{
   color:red;
}
```

15.b.i Angular Router Tutorial (8)

Step 1:

Create a new Project

```
ng new Routerdemo
```

Step 2:

Create three components namely home, about and dashboard by using the following command.

```
ng g c home
ng g c about
ng g c dashboard
```

Step 3:

Now app.module.ts file will look like as below

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { DashboardComponent } from './dashboard/dashboard.component';

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    AboutComponent,
    DashboardComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 4:

Add the following in app-routing.module.ts file

```
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { DashboardComponent } from './dashboard/dashboard.component';
const routes: Routes = [
  {
    path:'home',
    component:HomeComponent
  },
  {
    path:'aboutus',
    component: AboutComponent
  },
  {
    path:'dashboard',
    component: DashboardComponent
  }
];
```

Step5 :
Add the following line in app.component.html

```
<div class="toolbar" role="banner">
  <nav>
    <a routerLink="home">Home</a>
    <a routerLink="aboutus">About</a>
    <a routerLink="dashboard">Dashboard</a>
  </nav>
</div>
<router-outlet></router-outlet>
```

Step 6:
Run the Project using the following command

```
ng serve --open
```

15.b.ii **Reactive Form:** (8)

```
<form [formGroup]="registrationForm" (ngSubmit)="onSubmit()">
  <div class="form-row">
    <div class="form-group col-md-6">
      <label for="inputEmail4">Name</label>
      <input type="email" formControlName="uname" class="form-control" id="inputEmail4" >
      <span class="error-block"
      *ngIf="!registrationForm.get('uname').valid && registrationForm.get('uname').touched" >Please valid
email</span>
    </div>

    <div class="form-group col-md-6">
      <label for="inputPassword4">City</label>
      <input type="password"formControlName="city" class="form-control" id="inputCity" >
    </div>

    <div class="form-group col-md-2">
      <label for="inputZip">Mobile</label>
      <input type="text" formControlName="mobile"class="form-control" id="inputMobile">
    </div>
  </div>

  <div class="form-group">
      <label for="gender">Gender:</label>
      <div>
          <input id="male" type="radio" value="male" name="gender" formControlName="gender">
           <label for="male">Male</label>
```

```
      </div>
      <div>
          <input id="female" type="radio" value="female" name="gender" formControlName="gender">
          <label for="female">Female</label>
      </div>
      </div>
    </div>

  <div class="form-group">
      <label for="gender">Food Preference:</label>
      <div>
           <input id="veg" type="radio" value="veg" name="food" formControlName="food">
            <label for="veg">veg</label>
      </div>
      <div>
          <input id="nonveg" type="radio" value="nonveg" name="food" formControlName="food">
          <label for="nonveg">Non Veg</label>
      </div>
        </div>
    </div>
     <button type="submit" class="btn btn-primary" >Sign in</button>
     <br>
     FORMNAME:{{registrationForm.value|json}}<br>
  </form>

 appcomponent.css
.ng-invalid .ng-touched
{
   border-color:red;
}
.error-block
{
   color:red;
}

appcomponent.ts
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'reactiveform';
  registrationForm:FormGroup;
 constructor(){}
  ngOnInit()
  {
  this.registrationForm=new FormGroup({
  uname:new FormControl('',[Validators.required,Validators.email]),
  mobile:new FormControl('',Validators.required),
  gender:new FormControl('',Validators.required),
  food:new FormControl('',Validators.required),
  city:new FormControl('',Validators.required)
  },this.passwordmatch);
  }

  onSubmit()
  {
```

```
    console.log(this.registrationForm)
   }

}
```

Appmoduel.ts

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```