**Practical 3**

1. Create component ‘prac-3’

>>> ng generate component prac-3

1. app/prac-3/prac-3.component.html:

<div>

<label>Enter Text:</label>

<input [(ngModel)]="textValue" placeholder="Type something" />

<button (click)="displayText()">Display Text</button>

</div>

<h1 \*ngIf="displayTextFlag">{{ textValue }}</h1>

1. app/prac-3/prac-3.component.ts:

import { CommonModule } from '@angular/common';

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

import { FormsModule, NgModel } from '@angular/forms';

@Component({

selector: 'app-prac-3',

standalone: true,

imports: [CommonModule, FormsModule],

templateUrl: './prac-3.component.html',

styleUrl: './prac-3.component.css',

})

export class Prac3Component {

textValue = '';

displayTextFlag = false;

displayText(): void {

this.displayTextFlag = true;

}

}

1. app/app.component.html:

<app-prac-3></app-prac-3>

1. app/app.component.ts:

…

@Component({

selector: 'app-root',

standalone: true,

imports: [CommonModule, RouterOutlet, Prac3Component],

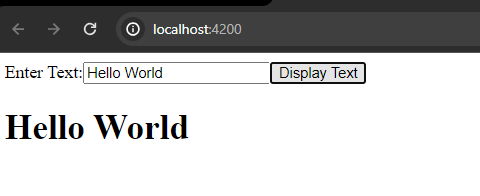
templateUrl: './app.component.html',

styleUrl: './app.component.css',

})

…

1. OUTPUT:



Practical Related exercise:

1. Interpolation and Property Binding:

HTML:

<div>

<p>Name: {{name}}</p>

<p>Enrollment Number: {{enrollmentNumber}}</p>

<p>College Name: {{collegeName}}</p>

<img [src]="photoURL">

</div>

TS:

export class YourComponent {

name = 'Your Name';

enrollmentNumber = 'Your Enrollment Number';

collegeName = 'Your College Name';

photoURL = 'path/to/your/photo.jpg';

}

1. Event Binding:

HTML:

<div>

<p>Name: {{name}}</p>

<p>Enrollment Number: {{enrollmentNumber}}</p>

<p>College Name: {{collegeName}}</p>

</div>

TS:

export class YourComponent {

name = '';

enrollmentNumber = '';

collegeName = '';

updateValues(event: Event) {

const target = event.target as HTMLInputElement;

switch (target.name) {

case 'name':

this.name = target.value;

break;

case 'enrollmentNumber':

this.enrollmentNumber = target.value;

break;

case 'collegeName':

this.collegeName = target.value;

break;

}

}

}

1. Login Page with Validations using ngModel Directive:

HTML:

<form #loginForm="ngForm" (ngSubmit)="submitForm(loginForm)">

<input type="email" name="email" ngModel required email>

<input type="password" name="password" ngModel required minlength="6">

<button type="submit">Login</button>

</form>

TS:

export class LoginComponent {

submitForm(form: NgForm) {

if (form.valid) {

console.log('Form Submitted');

} else {

console.log('Form Invalid');

}

}

}

Remember to import FormsModule in your module file to use ngModel.

**Practical 4**

1. Generate Component ‘prac-4’:

>>> ng g c prac-4

1. app/prac-4/prac-4.component.html:

<div>

<h1

[ngClass]="{ 'red-text': !displayTextFlag, 'green-text': displayTextFlag }"

(click)="toggleText()"

>

NG CLASS

</h1>

<h1

[ngStyle]="{

color: displayTextFlag ? 'green' : 'red',

'font-style': displayTextFlag ? 'italic' : 'normal'

}"

>

NG STYLE

</h1>

</div>

1. app/prac-4/prac-4.component.ts:

import { CommonModule } from '@angular/common';

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

@Component({

selector: 'app-prac-4',

standalone: true,

imports: [CommonModule],

templateUrl: './prac-4.component.html',

styleUrl: './prac-4.component.css',

})

export class Prac4Component {

displayTextFlag = false;

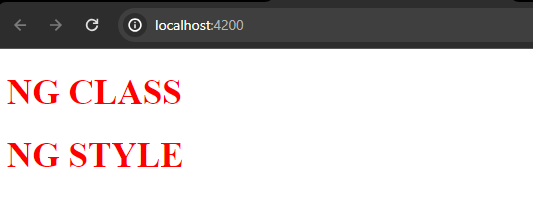
toggleText = () => {

this.displayTextFlag = !this.displayTextFlag;

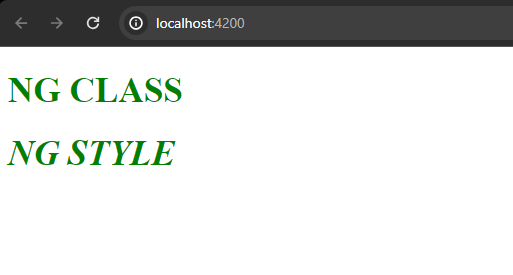
};

}

1. Import and call the component in your app (app html an ts file) component.
2. OUTPUT:



After clicking NG CLASS:



Practical Related Exercise:  
2. Header and footer,

1. Create a header component,
   1. >>> ng g c prac-4/header
   2. >>> ng g c prac-4/footer
2. Header HTML:

<div

[ngStyle]="{

'text-align': 'center',

backgroundColor: 'black',

color: 'white',

padding: '10px',

display: 'flex',

flexDirection: 'column',

}"

>

<h1>VPMP Polytechnic</h1>

<div>Affiliated to Gujarat Technological University</div>

<br />

</div>

1. Footer HTML:

<div [ngClass]="{ 'footer-style': true }">Computer Department | Sem 4</div>

1. Footer CSS:

.footer-style {

text-align: center;

background-color: black;

border-radius: 10px;

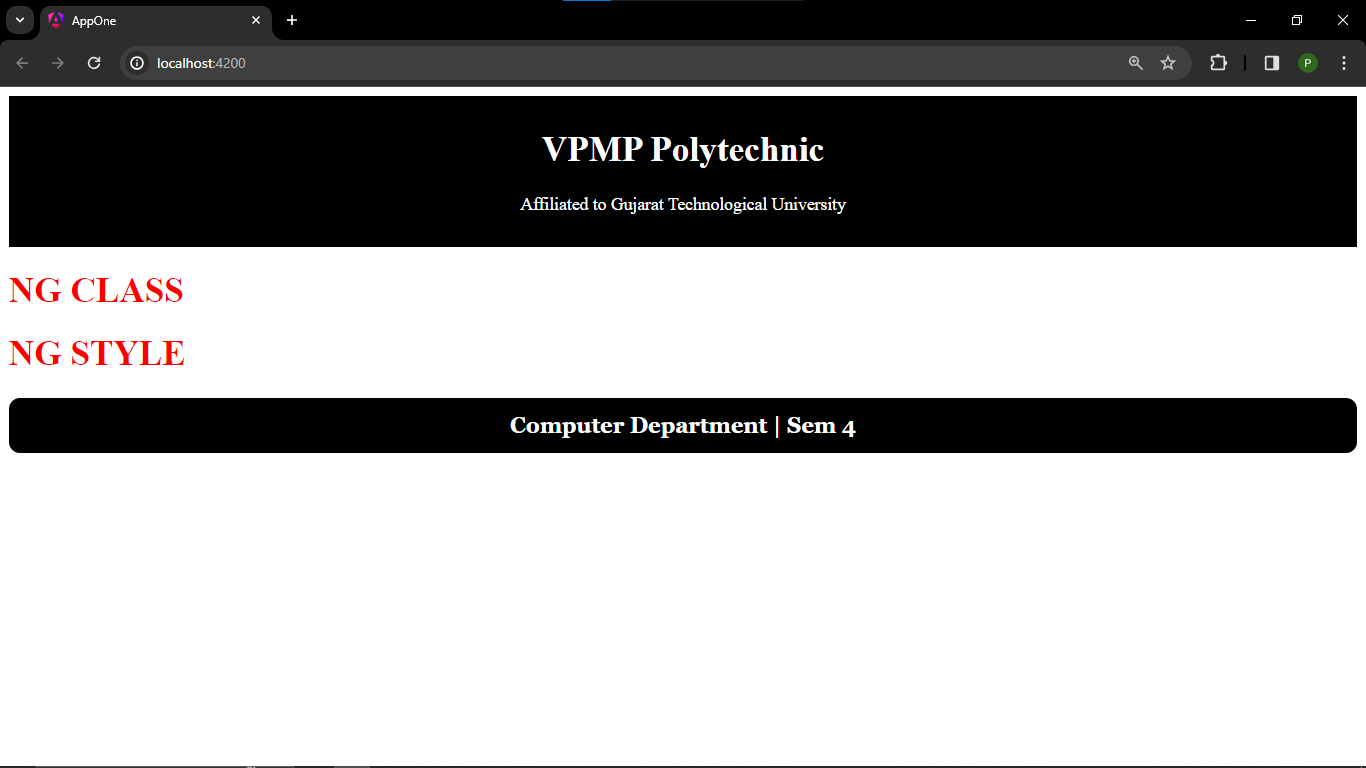
color: white;

padding: 10px;

font: bold 20px/30px Georgia, serif;

}

1. OUTPUT:



**Practical 4**

1. Generate Component ‘prac-5’

>>> ng g c prac-5

1. HTML:

<div>

<table>

<thead>

<th>Name</th>

<th>Percentage</th>

<th>Grades</th>

<th>Remarks</th>

</thead>

<tbody>

<tr

\*ngFor="let student of students; let i = index"

[ngClass]="i % 2 === 0 ? 'even' : 'odd'"

>

<td>{{ student.name }}</td>

<td>{{ student.percentage }}</td>

<td>{{ calculateGrade(student.percentage) }}</td>

<td [ngSwitch]="calculateGrade(student.percentage)">

<div \*ngSwitchCase="'A'">Excellent</div>

<div \*ngSwitchCase="'B'">Very Good</div>

<div \*ngSwitchCase="'C'">Good</div>

<div \*ngSwitchCase="'D'">Average</div>

<div \*ngSwitchDefault>Fail</div>

</td>

</tr>

</tbody>

</table>

</div>

1. TS:

…

export class Prac5Component {

students = [

{ name: 'Student 1', percentage: 98 },

{ name: 'Student 2', percentage: 85 },

{ name: 'Student 3', percentage: 78 },

];

calculateGrade(percentage: number): string {

if (percentage >= 90) return 'A';

else if (percentage >= 80) return 'B';

else if (percentage >= 70) return 'C';

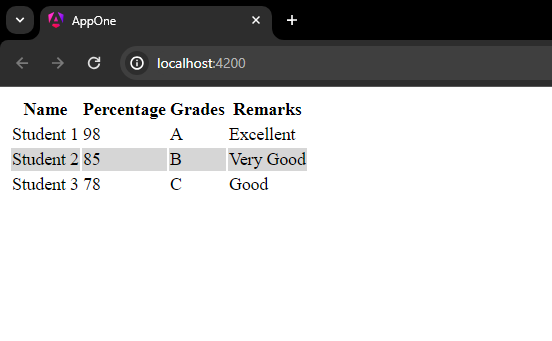
else if (percentage >= 60) return 'D';

else return 'F';

}

}

1. OUTPUT:



Practical Related Questions:

1. For image selection:
   1. HTML:

<select (change)="onChange($event)">

<option \*ngFor="let option of options" [value]="option">{{ option }}</option>

</select>

<div [ngSwitch]="selectedOption">

<img

\*ngSwitchCase="'Fire'"

src="https://images.unsplash.com/photo-1618325508550-951512a1e82d?q=80&w=1000&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8N3x8ZmlyZXxlbnwwfHwwfHx8MA%3D%3D"

alt="Image 1"

/>

<img

\*ngSwitchCase="'Ice'"

src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSZvxtDjSe6u4aDqNwwCPfaztHSIHYrcHJrBfeOUcvxiA&s"

alt="Image 2"

/>

<img

\*ngSwitchCase="'Water'"

src="https://static6.depositphotos.com/1043073/665/i/450/depositphotos\_6651515-stock-photo-water-splash.jpg"

alt="Image 3"

/>

<div \*ngSwitchDefault>Select an option</div>

</div>

* 1. TS:

import { CommonModule } from '@angular/common';

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

@Component({

selector: 'app-q-1',

standalone: true,

imports: [CommonModule],

templateUrl: './q-1.component.html',

styleUrl: './q-1.component.css',

})

export class Q1Component {

options = ['Fire', 'Ice', 'Water'];

selectedOption = '';

onChange(event: Event) {

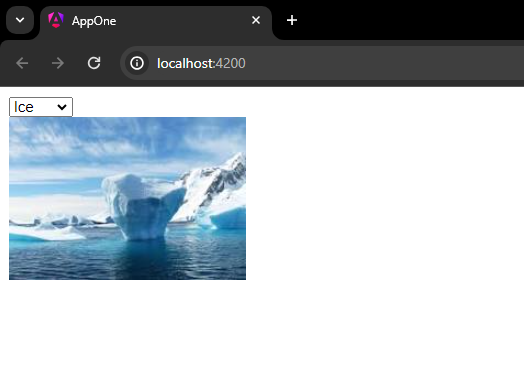
const target = event.target as HTMLSelectElement;

this.selectedOption = target.value;

}

}

* 1. OUTPUT:



1. (same as practical)

HTML:

<div>

<table>

<thead>

<th>Name</th>

<th>Percentage</th>

<th>Grades</th>

<th>Remarks</th>

</thead>

<tbody>

<tr

\*ngFor="let student of students; let i = index"

[ngClass]="i % 2 === 0 ? 'even' : 'odd'"

>

<td>{{ student.name }}</td>

<td>{{ student.percentage }}</td>

<td>{{ calculateGrade(student.percentage) }}</td>

<td [ngSwitch]="calculateGrade(student.percentage)">

<div \*ngSwitchCase="'A'">Excellent</div>

<div \*ngSwitchCase="'B'">Very Good</div>

<div \*ngSwitchCase="'C'">Good</div>

<div \*ngSwitchCase="'D'">Average</div>

<div \*ngSwitchDefault>Fail</div>

</td>

</tr>

</tbody>

</table>

</div>

TS:

…

export class Prac5Component {

students = [

{ name: 'Student 1', percentage: 98 },

{ name: 'Student 2', percentage: 85 },

{ name: 'Student 3', percentage: 78 },

];

calculateGrade(percentage: number): string {

if (percentage >= 90) return 'A';

else if (percentage >= 80) return 'B';

else if (percentage >= 70) return 'C';

else if (percentage >= 60) return 'D';

else return 'F';

}

}

**Practical 6**

1. Create Component ‘prac-6’

>>> ng g c prac-6

1. HTML:

<form (submit)="addStudent()">

<input

[(ngModel)]="newStudentNum"

[ngModelOptions]="{ standalone: true }"

placeholder="Enter student phone number"

/>

<button type="submit">Add Student</button>

</form>

<table>

<tr>

<th>ID</th>

<th>Name</th>

<th>Action</th>

</tr>

<tr \*ngFor="let student of students">

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

<td>{{ student.num }}</td>

<td><button (click)="removeStudent(student.id)">Remove</button></td>

</tr>

</table>

1. TS:

import { CommonModule } from '@angular/common';

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

import { FormsModule } from '@angular/forms';

class Student {

id: number = 0;

num: number = 0;

}

@Component({

selector: 'app-prac-6',

standalone: true,

imports: [CommonModule, FormsModule],

templateUrl: './prac-6.component.html',

styleUrl: './prac-6.component.css',

})

export class Prac6Component {

students: Student[] = [];

newStudentNum: number = 0;

addStudent() {

const newStudent = new Student();

newStudent.id = this.students.length + 1;

newStudent.num = this.newStudentNum;

this.students.push(newStudent);

this.newStudentNum = 0;

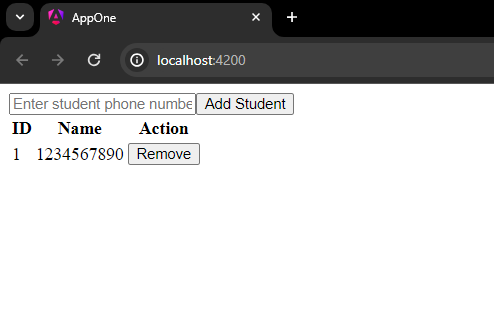
}

removeStudent(studentId: number) {

this.students = this.students.filter((student) => student.id !== studentId);

}

}

1. OUTPUT:  
   

Practical Related Questions:

1. Code:
   1. HTML:

<form (submit)="addStudent()">

<input

[(ngModel)]="newStudentName"

[ngModelOptions]="{ standalone: true }"

placeholder="Enter student name"

/>

<button type="submit">Add Student</button>

</form>

<table>

<tr>

<th>ID</th>

<th>Name</th>

<th>Action</th>

</tr>

<tr \*ngFor="let student of students">

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

<td>{{ student.name }}</td>

<td><button (click)="removeStudent(student.id)">Remove</button></td>

</tr>

</table>

* 1. TS:

import { CommonModule } from '@angular/common';

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

import { FormsModule } from '@angular/forms';

class Student {

id: number = 0;

name: string = '';

}

@Component({

selector: 'app-p6q1',

standalone: true,

imports: [CommonModule, FormsModule],

templateUrl: './p6q1.component.html',

styleUrl: './p6q1.component.css',

})

export class P6q1Component {

students: Student[] = [];

newStudentName: string = '';

maxStudents: number = 10;

addStudent() {

if (this.students.length >= this.maxStudents) {

alert('Maximum limit Reached');

return;

}

const newStudent = new Student();

newStudent.id = this.students.length + 1;

newStudent.name = this.newStudentName;

this.students.push(newStudent);

this.newStudentName = '';

}

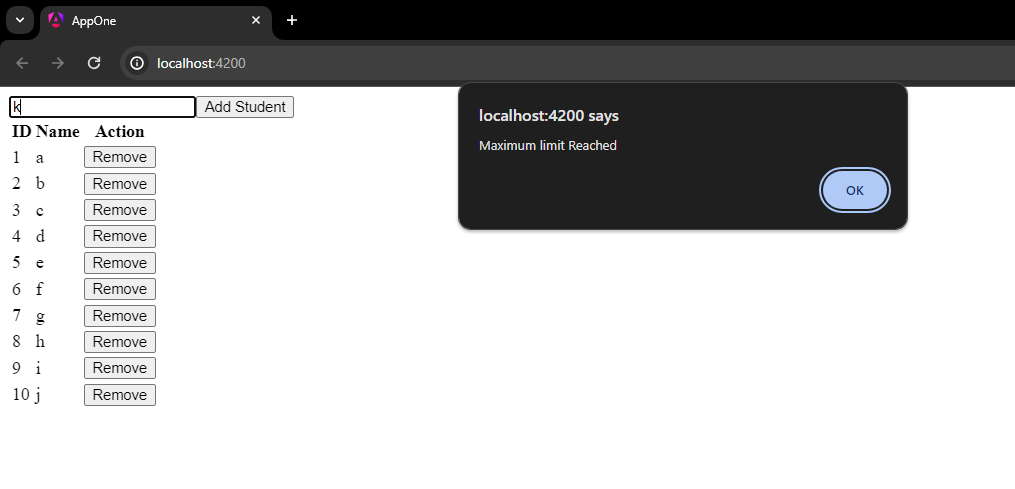
removeStudent(studentId: number) {

this.students = this.students.filter((student) => student.id !== studentId);

}

}

* 1. OUTPUT:



1. Code:

HTML:

<table>

<tr>

<th>ID</th>

<th>Name</th>

<th>Action</th>

</tr>

<tr \*ngFor="let product of products">

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

<td>{{ product.name }}</td>

<td><button (click)="removeProduct(product.id)">Remove</button></td>

</tr>

</table>

TS:  
import { CommonModule } from '@angular/common';

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

import { FormsModule } from '@angular/forms';

class Product {

id: number = 0;

name: string = '';

}

@Component({

selector: 'app-p6q2',

standalone: true,

imports: [CommonModule, FormsModule],

templateUrl: './p6q2.component.html',

styleUrl: './p6q2.component.css',

})

export class P6q2Component {

products: Product[] = [

{ id: 1, name: 'Apple' },

{ id: 2, name: 'Banana' },

{ id: 3, name: 'Cherry' },

{ id: 4, name: 'Dates' },

];

removeProduct(productId: number) {

const productIndex = this.products.findIndex(

(product) => product.id === productId

);

if (productIndex > -1 && productIndex < this.products.length - 1) {

this.products.splice(productIndex, 2);

} else {

this.products = this.products.filter(

(product) => product.id !== productId

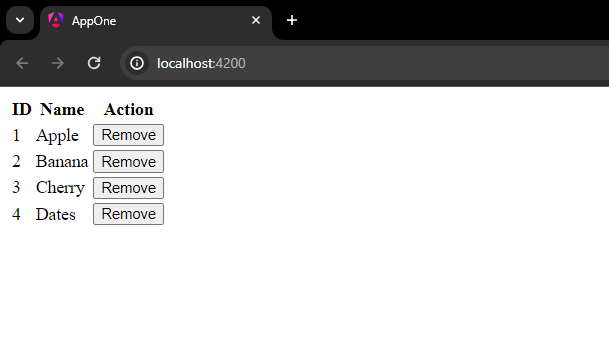
);

}

}

}

OUTPUT:



**Practical 7**

(Requires app with –no-standalone)

1. Create Component prac-7

>>> ng g c prac-7

1. Prac-7 HTML:

<input [(ngModel)]="searchInput" placeholder="Search products">

<button (click)="performSearch()">Search</button>

<div \*ngFor="let product of products | filter:searchTerm">

<!-- Display product details -->

<h2>{{product.name}}</h2>

<p>ID: {{product.id}}</p>

<p>Purchase Date: {{product.purchaseDate | date}}</p>

<p>Price: {{product.price }}</p>

<img [src]="product.image" alt="{{product.name}}">

</div>

1. TS:

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

export class Product {

id: number;

name: string;

purchaseDate: Date;

price: number;

image: string;

constructor() {

this.id = 0;

this.name = '';

this.purchaseDate = new Date();

this.price = 0;

this.image = '';

}

}

@Component({

selector: 'app-prac-7',

templateUrl: './prac-7.component.html',

styleUrl: './prac-7.component.css',

})

export class Prac7Component {

searchInput: string = '';

searchTerm: string = '';

products: Product[] = [

{

id: 1,

name: 'Football',

purchaseDate: new Date('2020-01-01'),

price: 799,

image: 'https://via.placeholder.com/150',

},

{

id: 2,

name: 'Chess Set',

purchaseDate: new Date('2020-02-01'),

price: 899,

image: 'https://via.placeholder.com/150',

},

{

id: 3,

name: 'Cricket Ball',

purchaseDate: new Date('2020-03-01'),

price: 79,

image: 'https://via.placeholder.com/150',

},

];

performSearch() {

this.searchTerm = this.searchInput;

}

}

1. Pipe (create it using ng g pipe filter):

import { Pipe, PipeTransform } from '@angular/core';

import { Product } from './prac-7.component';

@Pipe({

name: 'filter',

})

export class FilterPipe implements PipeTransform {

transform(products: Product[], searchTerm: string): Product[] {

if (!products || !searchTerm) {

return products;

}

return products.filter((product) =>

product.name.toLowerCase().includes(searchTerm.toLowerCase())

);

}

}

1. 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 { Prac7Component } from './prac-7/prac-7.component';

import { FilterPipe } from './prac-7/filter.pipe';

import { FormsModule } from '@angular/forms';

@NgModule({

declarations: [AppComponent, Prac7Component , FilterPipe],

imports: [BrowserModule, AppRoutingModule, FormsModule],

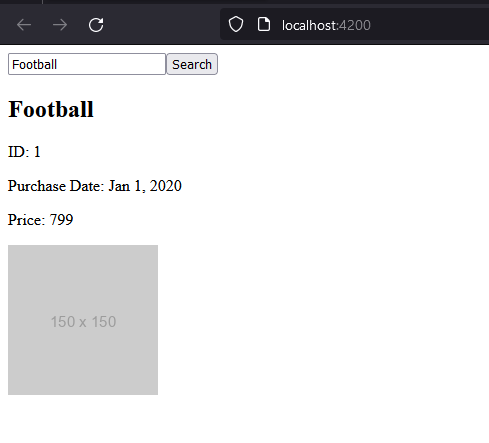
providers: [],

bootstrap: [AppComponent],

})

export class AppModule {}

1. OUTPUT:



Practical Related Questions:

1. CODE:
   1. HTML:

<h1>Date Pipe</h1>

<p>{{ date | date }}</p>

<h1>Number Pipe</h1>

<p>{{ number | number }}</p>

<h1>Uppercase Pipe</h1>

<p>{{ text | uppercase }}</p>

<h1>Lowercase Pipe</h1>

<p>{{ text | lowercase }}</p>

<h1>Currency Pipe</h1>

<p>{{ number | currency }}</p>

<h1>Percent Pipe</h1>

<p>{{ number | percent }}</p>

* 1. TS:

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

@Component({

selector: 'app-p7q1',

templateUrl: './p7q1.component.html',

})

export class P7Q1Component {

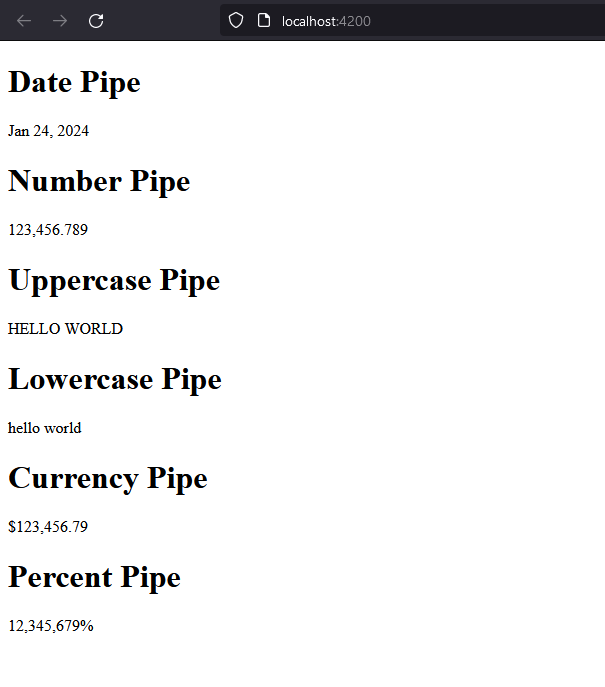
date = new Date();

number = 123456.789;

text = 'hello world';

}

OUTPUT:



1. CODE:

HTML:

<ul>

<li \*ngFor="let student of students">

<h2>{{ student.name }}</h2>

<p>ID: {{ student.id }}</p>

<p>Address: {{ student.address | slice:0:20 }}...</p>

<p>Semester: {{ student.semester }}</p>

</li>

</ul>

TS:  
import { Component } from '@angular/core';

@Component({

selector: 'app-p7q2',

templateUrl: './p7q2.component.html',

})

export class P7Q2Component {

students = [

{

id: 1,

name: 'ABC',

address: '123 Sector 5 Gandhinagar Gujarat India',

semester: '4',

},

// Add more students as needed

];

}

OUTPUT:  


**Practical 8**

(Requires app with –no-standalone)

1. Create Component prac-8

>>> ng g c prac-8

1. HTML:

<form #studentForm="ngForm" (ngSubmit)="register(studentForm)">

<label>

Name:

<input type="text" name="name" ngModel required>

<div \*ngIf="!studentForm.controls['name'].valid && studentForm.controls['name'].touched">

Name is required.

</div>

</label>

<label>

Email:

<input type="email" name="email" ngModel required>

<div \*ngIf="!studentForm.controls['email'].valid && studentForm.controls['email'].touched">

Please enter a valid email.

</div>

</label>

<label>

Password:

<input type="password" name="password" ngModel required minlength="8">

<div \*ngIf="!studentForm.controls['password'].valid && studentForm.controls['password'].touched">

Password is required and must be at least 8 characters long.

</div>

</label>

<button type="submit" [disabled]="!studentForm.form.valid">Register</button>

</form>

1. TS:

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

import { NgForm } from '@angular/forms';

@Component({

selector: 'app-prac-8',

templateUrl: './prac-8.component.html',

styleUrls: ['./prac-8.component.css'],

})

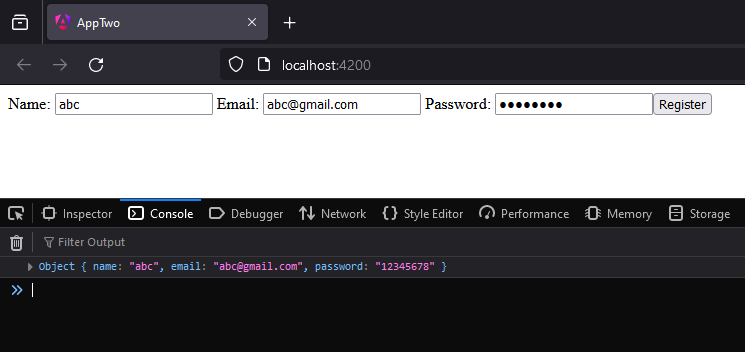
export class Prac8Component {

register(form: NgForm) {

console.log(form.value);

}

}

1. OUTPUT:  
   

Practical Related Questions:

1. CODE:

HTML:

<form #studentForm="ngForm" (ngSubmit)="calculateGrade(studentForm)">

<label>

Angular:

<input type="number" name="subject1" ngModel required>

</label>

<br>

<label>

JAVA:

<input type="number" name="subject2" ngModel required>

</label>

<br>

<label>

Software Engineering:

<input type="number" name="subject3" ngModel required>

</label>

<br>

<button type="submit" [disabled]="!studentForm.form.valid">Calculate Grade</button>

</form>

<div \*ngIf="grade">

Your grade is: {{ grade }}

</div>

<div \*ngIf="!grade">

Sorry, you did not pass.

</div>

TS:

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

import { NgForm } from '@angular/forms';

@Component({

selector: 'app-p8q1',

templateUrl: './p8q1.component.html',

styleUrls: ['./p8q1.component.css'],

})

export class P8q1Component {

grade: string | null = null;

calculateGrade(form: NgForm) {

const subject1 = form.value.subject1;

const subject2 = form.value.subject2;

const subject3 = form.value.subject3;

const averageMark = (subject1 + subject2 + subject3) / 3;

if (averageMark > 50) {

this.grade = 'Passed';

} else {

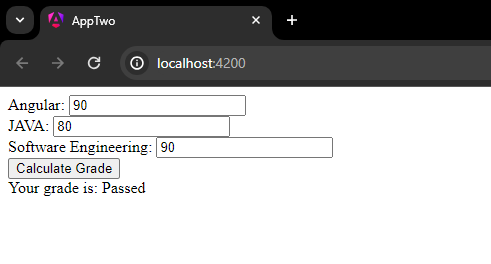
this.grade = null;

}

}

}

OUTPUT:



1. CODE:

HTML:

<form #feedbackForm="ngForm" (ngSubmit)="submitFeedback(feedbackForm)">

<label>

Teacher's Name:

<input type="text" name="teacherName" ngModel required>

</label>

<br>

<label>

Application Name:

<input type="text" name="applicationName" ngModel required>

</label>

<br>

<label>

Feedback:

<textarea name="feedback" ngModel required></textarea>

</label>

<br>

<button type="submit" [disabled]="!feedbackForm.form.valid">Submit Feedback</button>

</form>

TS:

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

import { NgForm } from '@angular/forms';

@Component({

selector: 'app-p8q2',

templateUrl: './p8q2.component.html',

styleUrls: ['./p8q2.component.css']

})

export class P8q2Component {

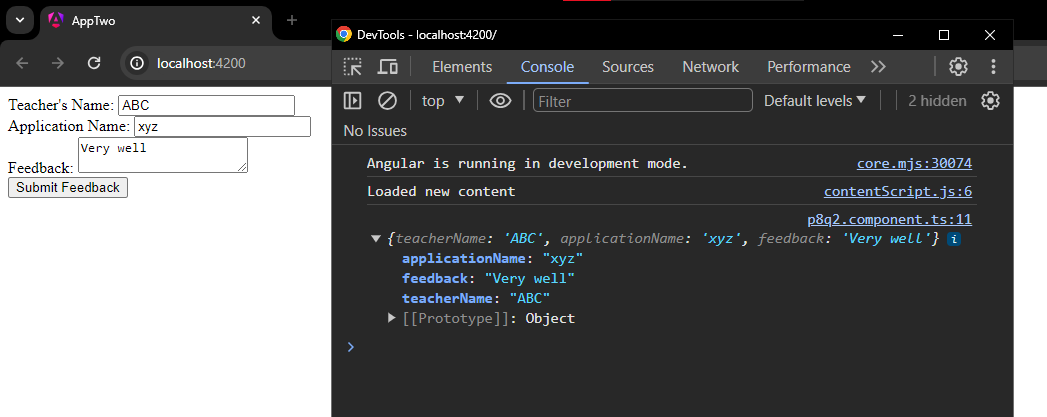
submitFeedback(form: NgForm) {

console.log(form.value);

}

}

OUTPUT:



**Practical 9**

(Requires app with –no-standalone)

1. Generate Component prac-9

>>> ng g c prac-9

1. HTML:

<form #facultyForm="ngForm" (ngSubmit)="onSubmit(facultyForm)">

<label>

Code:

<input type="text" name="code" ngModel required>

</label>

<br>

<label>

Name:

<input type="text" name="name" ngModel required>

</label>

<br>

<label>

Email:

<input type="email" name="email" ngModel required>

</label>

<br>

<label>

Type:

<select name="type" ngModel required>

<option value="">Select Type</option>

<option value="fullTime">Full Time</option>

<option value="partTime">Part Time</option>

</select>

</label>

<br>

<label>

Status:

<select name="status" ngModel required>

<option value="">Select Status</option>

<option value="active">Active</option>

<option value="inactive">Inactive</option>

</select>

</label>

<br>

<label>

Subjects Teaching:

<div \*ngFor="let subject of subjects; let i=index">

<input [(ngModel)]="subjects[i]" name="subject{{i}}" required>

</div>

<button (click)="addSubject()">Add Subject</button>

</label>

<br>

<button type="submit" [disabled]="!facultyForm.form.valid">Submit</button>

</form>

1. TS:

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

import { NgForm } from '@angular/forms';

@Component({

selector: 'app-prac-9',

templateUrl: './prac-9.component.html',

styleUrls: ['./prac-9.component.css'],

})

export class Prac9Component {

subjects: string[] = [];

addSubject(): void {

this.subjects.push('');

}

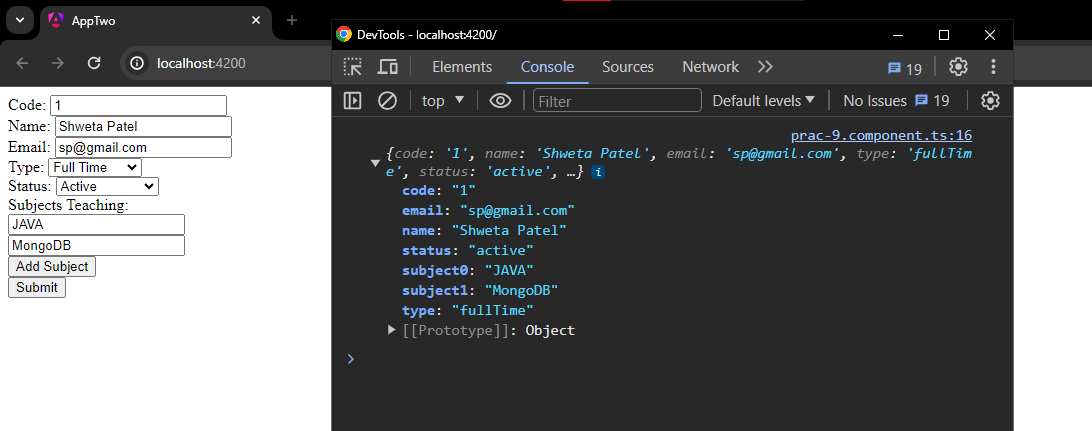
onSubmit(form: NgForm): void {

console.log(form.value);

}

}

1. OUTPUT:



Practical Related Questions:

|  |  |  |
| --- | --- | --- |
| Aspect | Template-Driven Forms | Reactive Forms |
| Form Model | Uses directives in the template to bind to DOM elements. | Uses a model-driven approach where form controls are defined in the component class. |
| Validation | Limited to basic validations. | Provides robust validation capabilities including custom validators. |
| Flexibility | Less flexible compared to Reactive Forms. | More flexible and powerful, suitable for complex scenarios. |
| Performance | Slower because of the overhead of change detection. | Faster because it updates only the components that are affected by changes. |
| Learning Curve | Easier to learn and use, especially for beginners. | Has a steeper learning curve due to its complexity. |
| Use Case | Ideal for simple forms and prototyping. | Ideal for complex forms and dynamic forms. |

1. CODE:

HTML:

<form #userForm="ngForm" (ngSubmit)="onSubmit(userForm)">

<label>

Name:

<input type="text" name="name" ngModel required>

</label>

<br>

<label>

Email:

<input type="email" name="email" ngModel required>

</label>

<br>

<label>

Phone Number:

<input type="tel" name="phoneNumber" ngModel pattern="^\d{10}$" required>

</label>

<br>

<label>

Password:

<input type="password" name="password" ngModel required minlength="8">

</label>

<br>

<button type="submit" [disabled]="!userForm.form.valid">Submit</button>

</form>

TS:

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

import { NgForm } from '@angular/forms';

@Component({

selector: 'app-p9q2',

templateUrl: './p9q2.component.html',

styleUrls: ['./p9q2.component.css'],

})

export class P9q2Component {

onSubmit(form: NgForm): void {

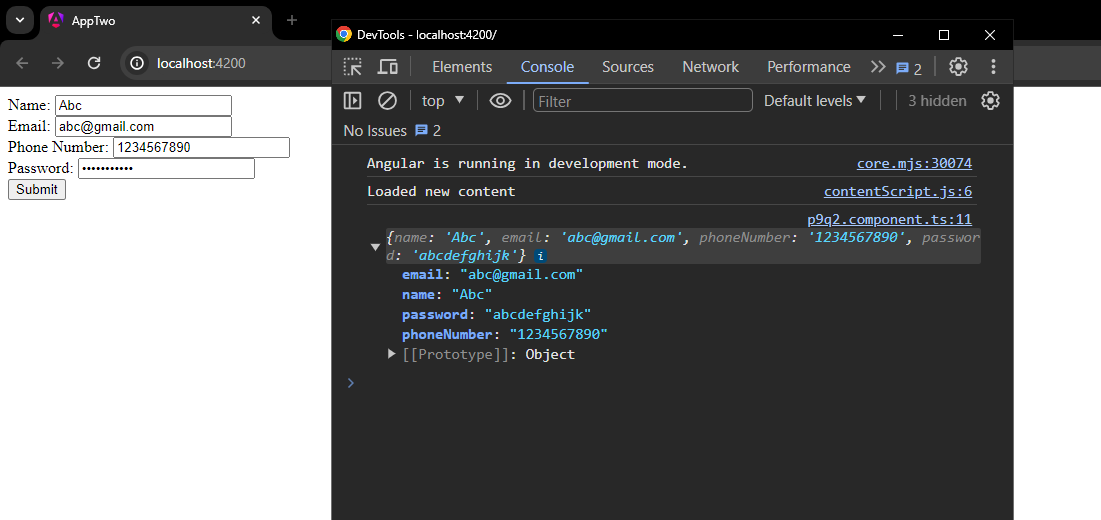
console.log(form.value);

// Handle form submission here

}

}

OUTPUT:



**Practical 10**

(Requires app with –no-standalone)

1. Generate Component prac-10

>>> ng g c prac-10

1. App html:

<app-prac-10 (productAdded)="onProductAdded($event)"></app-prac-10>

<h2>Added Items:</h2>

<ul>

<li \*ngFor="let item of addedItems">{{ item.name }} - ${{ item.price }}</li>

</ul>

1. App TS:

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

interface Product {

name: string;

price: string;

}

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css'],

})

export class AppComponent {

addedItems: Product[] = [];

onProductAdded(product: Product): void {

this.addedItems.push(product);

}

}

1. Component html:

<div \*ngFor="let product of products">

<h2>{{ product.name }}</h2>

<p>{{ product.price }}</p>

<button (click)="addToCart(product)">Add to Cart</button>

</div>

1. Component TS:

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

@Component({

selector: 'app-prac-10',

templateUrl: './prac-10.component.html',

styleUrls: ['./prac-10.component.css'],

})

export class Prac10Component {

@Output() productAdded = new EventEmitter<any>();

products = [

{ name: 'Product 1', price: '10' },

{ name: 'Product 2', price: '20' },

{ name: 'Product 3', price: '30' },

];

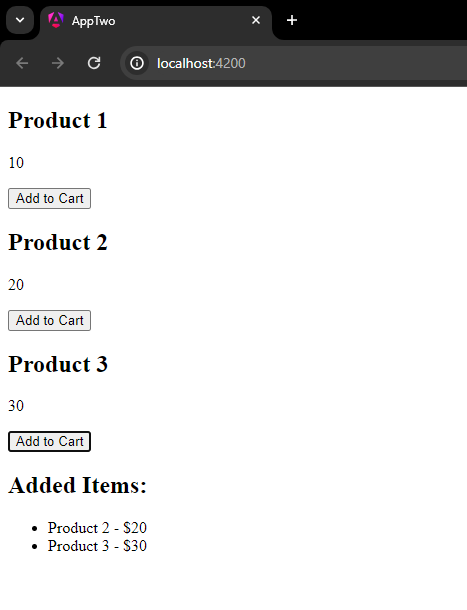
addToCart(product: any): void {

this.productAdded.emit(product);

}

}

1. OUTPUT:



**Practical 11**

(Requires app with –no-standalone)

1. Generate Component prac-11

>>> ng g c prac-11

1. TS:

import {

Component,

OnInit,

OnChanges,

DoCheck,

AfterContentInit,

AfterContentChecked,

AfterViewInit,

AfterViewChecked,

OnDestroy,

SimpleChanges

} from '@angular/core';

@Component({

selector: 'app-prac-11',

templateUrl: './prac-11.component.html',

styleUrl: './prac-11.component.css',

})

export class Prac11Component {

constructor() {}

ngOnInit(): void {

console.log('ngOnInit');

}

ngOnChanges(changes: SimpleChanges): void {

console.log('ngOnChanges');

}

ngDoCheck(): void {

console.log('ngDoCheck');

}

ngAfterContentInit(): void {

console.log('ngAfterContentInit');

}

ngAfterContentChecked(): void {

console.log('ngAfterContentChecked');

}

ngAfterViewInit(): void {

console.log('ngAfterViewInit');

}

ngAfterViewChecked(): void {

console.log('ngAfterViewChecked');

}

ngOnDestroy(): void {

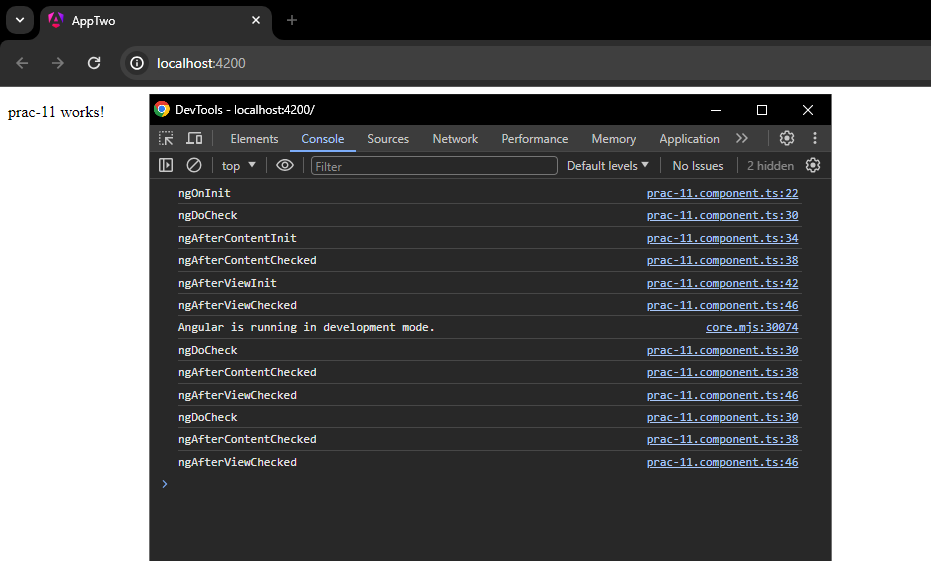
console.log('ngOnDestroy');

}

}

1. OUTPUT:

Following shows component lifecycle order of execution,



**Practical 12**

(Requires app with –no-standalone)

1. Make a folder named prac-12
2. Generate Component product

>>> ng g c prac-12/product

1. Generate Component product-details

>>> ng g c prac-12/product-details

1. Product:

HTML:

<div \*ngFor="let product of products">

<h2>{{ product.name }}</h2>

<button (click)="selectProduct(product)">View Details</button>

</div>

<app-product-details \*ngIf="selectedProduct" [product]="selectedProduct"></app-product-details>

TS:

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

@Component({

selector: 'app-product',

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

styleUrls: ['./product.component.css'],

})

export class ProductComponent {

products = [

{ id: 1, name: 'Product 1', price: 100 },

{ id: 2, name: 'Product 2', price: 200 },

// Add more products as needed

];

selectedProduct: any;

selectProduct(product: any) {

this.selectedProduct = product;

}

}

1. Product details:

HTML:

<div \*ngIf="product">

<h2>{{ product.name }}</h2>

<p>ID: {{ product.id }}</p>

<p>Price: ${{ product.price }}</p>

</div>

TS:

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

@Component({

selector: 'app-product-details',

templateUrl: './product-details.component.html',

styleUrls: ['./product-details.component.css'],

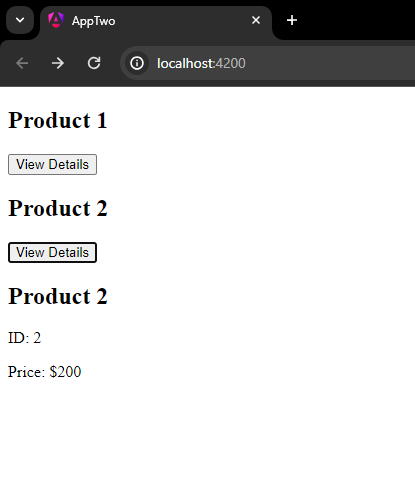
})

export class ProductDetailsComponent {

@Input() product: any;

}

1. Call both components in app
2. OUTPUT:



**Practical 13**

(Requires app with –no-standalone)

1. Generate Component prac-13

>>> ng g c prac-13

1. Generate Service services:

>>> ng g s services

1. Service TS:

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

@Injectable({

providedIn: 'root',

})

export class StudentService {

constructor() {}

getStudentData(): any {

return {

id: 1,

name: 'ABC',

age: 20,

course: 'Diploma in CE',

};

}

}

1. Prac-13 TS:

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

import { StudentService } from '../services/student.service';

@Component({

selector: 'app-prac-13',

templateUrl: './prac-13.component.html',

styleUrls: ['./prac-13.component.css'],

})

export class Prac13Component implements OnInit {

student: any;

constructor(private studentService: StudentService) {}

ngOnInit(): void {

this.student = this.studentService.getStudentData();

}

}

1. HTML:

<div \*ngIf="student">

<h2>{{ student.name }}</h2>

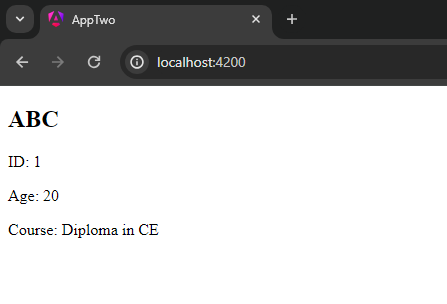
<p>ID: {{ student.id }}</p>

<p>Age: {{ student.age }}</p>

<p>Course: {{ student.course }}</p>

</div>

1. OUTPUT:



**Practical 14**

(Requires a simple http-server made using ExpressJS (As node js))

1. Install EXPRESS JS and NODEMON ( for HMR )

>>> npm install express nodemon

1. Install CORS for Cross Origin Resource Sharing

>>> npm install cors

1. Create a http-server file (httpServer.js) with following code:

const express = require('express');

const cors = require('cors');

const app = express();

app.use(cors());

const data = [

{

name: 'Football',

content: "This shows description for football"

},

{

name: 'Chess',

content: "This shows description for chess"

}

]

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

res.send(data);

})

app.listen(3000 , () => console.log('server started'))

1. Run the server

>>> nodemon httpServer.js

1. The server should be started on http://localhost:3000

(Requires app with –no-standalone)

1. Generate Service sport:

>>> ng g s services/sport

1. Service TS:

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

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

import { Observable } from 'rxjs';

import { map } from 'rxjs/operators';

@Injectable({

providedIn: 'root',

})

export class SportService {

private apiUrl = 'http://localhost:3000/sports';

constructor(private http: HttpClient) {}

getSports(): Observable<any[]> {

return this.http.get<any[]>(this.apiUrl).pipe(map((response) => response));

}

searchSports(term: string): Observable<any[]> {

return this.http

.get<any[]>(`${this.apiUrl}?search=${term}`)

.pipe(map((response) => response));

}

}

1. Generate Component prac-14:

>>> ng g c prac-14

1. Prac-14 TS:

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

import { SportService } from '../services/sport.service';

@Component({

selector: 'app-prac-14',

templateUrl: './prac-14.component.html',

styleUrls: ['./prac-14.component.css'],

})

export class Prac14Component implements OnInit {

sports: any[] = [];

searchTerm: string = '';

constructor(private sportService: SportService) {}

ngOnInit(): void {

this.fetchSports();

}

fetchSports(): void {

this.sportService.getSports().subscribe((data: any[]) => {

this.sports = data;

});

}

searchSports(): void {

this.sportService.searchSports(this.searchTerm).subscribe((data: any[]) => {

this.sports = data;

});

}

}

1. HTML:

<input [(ngModel)]="searchTerm" placeholder="Search sports" (input)="searchSports()" />

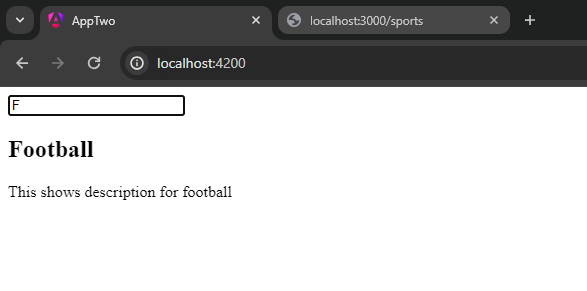
<div \*ngFor="let sport of sports">

<h2>{{ sport.name }}</h2>

<p>{{ sport.content }}</p>

</div>

1. OUTPUT:



**Practical 15**

(Requires a simple http-server made using ExpressJS (As node js))

1. The server and the services remain same as practical 14. Make sure the server is running.
2. Generate Component prac-15:

>>> ng g c prac-15

1. TS:

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

import { SportService } from '../services/sport.service';

@Component({

selector: 'app-prac-15',

templateUrl: './prac-15.component.html',

styleUrls: ['./prac-15.component.css'],

})

export class Prac15Component implements OnInit {

sports: any[] = [];

constructor(private sportService: SportService) {}

ngOnInit(): void {

this.fetchSports();

}

fetchSports(): void {

this.sportService.getSports().subscribe((data: any[]) => {

this.sports = data;

});

}

}

1. HTML:

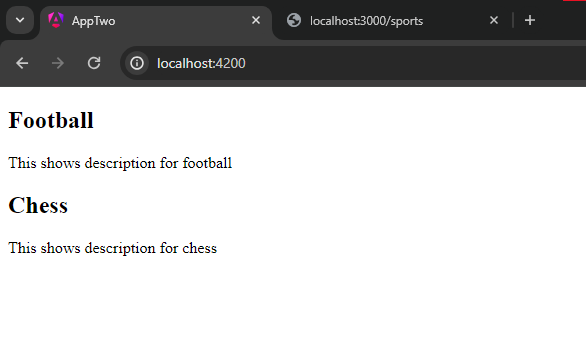
<div \*ngFor="let sport of sports">

<h2>{{ sport.name }}</h2>

<p>{{ sport.content }}</p>

</div>

1. OUTPUT:



**Practical 16**

(Requires a simple http-server made using ExpressJS (As node js))

1. Install dependencies:

>>> npm i body-parser mysql

1. Create Component prac-16:

>>> ng g c prac-16

1. Requires a database, here, phpMyAdmin (provided in XAMPP) is used.
2. Server:

const express = require("express");

const bodyParser = require("body-parser");

const mysql = require("mysql");

const cors = require("cors");

const app = express();

app.use(bodyParser.json());

app.use(cors());

const connection = mysql.createConnection({

host: "localhost",

user: "root",

password: "",

database: "angular-data",

});

connection.connect((err) => {

if (err) throw err;

console.log("Connected to the database!");

});

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

const { username, password } = req.body;

const sql = "INSERT INTO userdata (username, password) VALUES (?, ?)";

connection.query(sql, [username, password], (err, result) => {

if (err) throw err;

res.send("User data saved successfully!");

});

});

app.listen(3000, () => {

console.log("Server is running on port 3000");

});

1. HTML:

<form (submit)="onSubmit()">

<label>

Username:

<input [(ngModel)]="username" name="username" required>

</label>

<label>

Password:

<input [(ngModel)]="password" name="password" type="password" required>

</label>

<button type="submit">Submit</button>

</form>

1. TS:

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

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

@Component({

selector: 'app-prac-16',

templateUrl: './prac-16.component.html',

styleUrls: ['./prac-16.component.css'],

})

export class Prac16Component {

username: string = '';

password: string = '';

constructor(private http: HttpClient) {}

onSubmit(): void {

const url = 'http://localhost:3000/saveUserData'; // Replace with your Express server URL

this.http

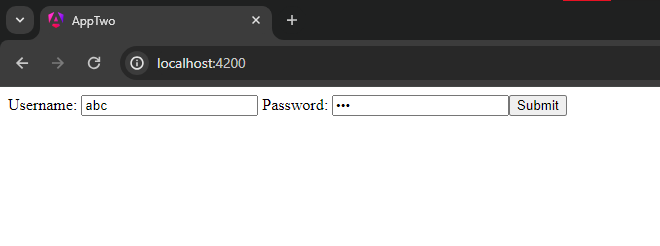
.post(url, { username: this.username, password: this.password })

.subscribe();

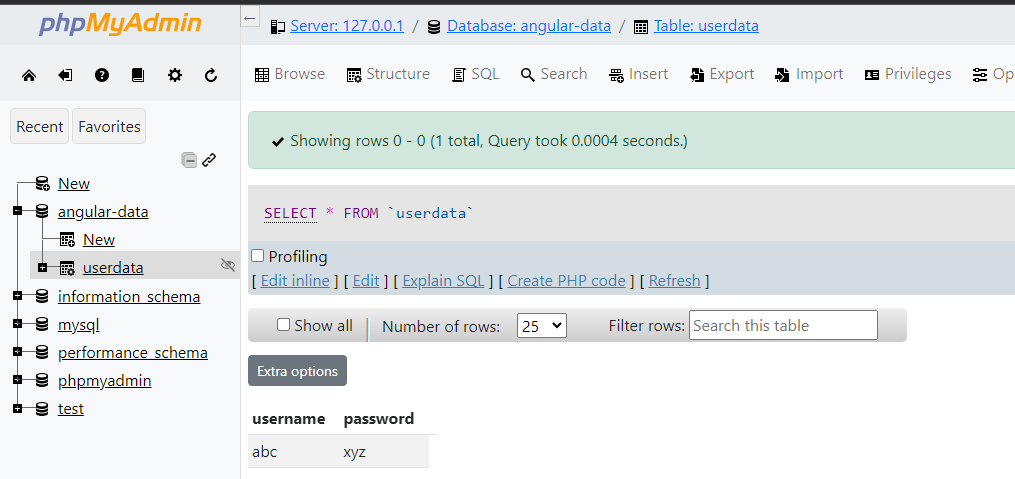
}

}

1. OUTPUT:
   1. Frontend:



* 1. DATABASE:



**Practical 18**

(Requires app with –no-standalone)

1. Generate Component inside prac-18 folder:

>>> ng g c prac-18/product-list

1. Product list TS:

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

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

@Component({

selector: 'app-product-list',

templateUrl: './product-list.component.html',

styleUrl: './product-list.component.css',

})

export class ProductListComponent implements OnInit {

products = [

{ id: 1, name: 'Product 1', price: '10' },

{ id: 2, name: 'Product 2', price: '20' },

{ id: 3, name: 'Product 3', price: '30' },

{ id: 4, name: 'Product 4', price: '40' },

{ id: 5, name: 'Product 5', price: '50' },

{ id: 6, name: 'Product 6', price: '60' },

];

currentPage = 1;

itemsPerPage = 5;

constructor() {}

ngOnInit(): void {}

get displayedProducts() {

const start = (this.currentPage - 1) \* this.itemsPerPage;

const end = start + this.itemsPerPage;

return this.products.slice(start, end);

}

nextPage() {

if (this.currentPage \* this.itemsPerPage < this.products.length) {

this.currentPage++;

}

}

prevPage() {

if (this.currentPage > 1) {

this.currentPage--;

}

}

}

1. HTML (app):

<h1>Routes</h1>

<nav>

<a routerLink="/">

<h1>Home</h1>

</a>

<a routerLink="/products"><h1>Products</h1></a>

</nav>

<router-outlet></router-outlet>

1. HTML (product list):

<div \*ngFor="let product of displayedProducts">

<h2>{{ product.name }}</h2>

<p>Price: {{ product.price }}</p>

</div>

<button (click)="prevPage()" [disabled]="currentPage === 1">Previous Page</button>

<button (click)="nextPage()" [disabled]="currentPage \* itemsPerPage >= products.length">Next Page</button>

1. app-rounting.module.ts:

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

import { RouterModule, Routes } from '@angular/router';

import { ProductListComponent } from './prac-18/product-list/product-list.component';

const routes: Routes = [

{ path: 'products', component: ProductListComponent },

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule],

})

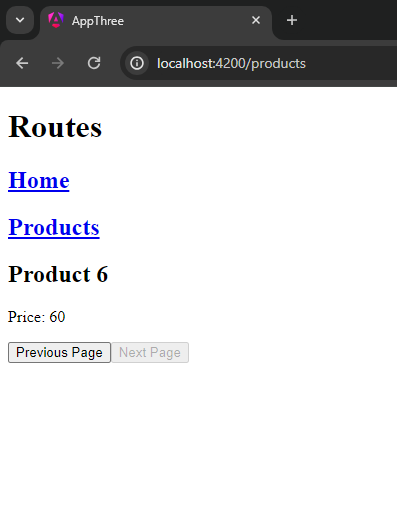
export class AppRoutingModule {}

1. Output:

Page 1:



Page 2:



**Practical 19**

(Requires app with –no-standalone)

1. Generate Component inside prac-19 folder:

>>> ng g c prac-18/customer-orders

1. TS (customer-orders):

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

@Component({

selector: 'app-customer-orders',

templateUrl: './customer-orders.component.html',

styleUrl: './customer-orders.component.css',

})

export class CustomerOrdersComponent {

orders = [

{ id: 1, customerId: 1, amount: '$100' },

{ id: 2, customerId: 1, amount: '$200' },

];

constructor() {}

ngOnInit(): void {}

}

1. HTML (component):

<div \*ngFor="let order of orders">

<h2>Order ID: {{ order.id }}</h2>

<p>Amount: {{ order.amount }}</p>

</div>

1. HTML (app):

<h1>Routes</h1>

<nav>

<a routerLink="/">

<h1>Home</h1>

</a>

<a routerLink="/customer-orders"><h1>customer orders</h1></a>

</nav>

<router-outlet></router-outlet>

1. TS (routing):

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

import { RouterModule, Routes } from '@angular/router';

const routes: Routes = [

{

path: 'customer-orders',

loadChildren: () =>

import('./prac-19/customer-orders/customer-orders.component').then(

(m) => m.CustomerOrdersComponent

),

},

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

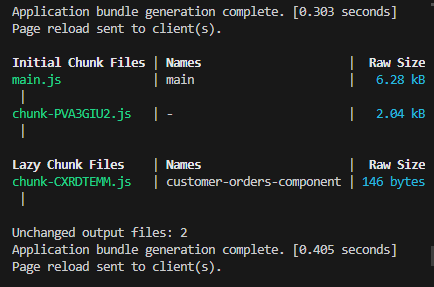
exports: [RouterModule],

})

export class AppRoutingModule {}

1. OUTPUT:

Result of lazy loading:



Leading to an optimized app

**Practical 20**

(Requires app with –no-standalone and an ExpressJS server)

Same as practical 16, as CORS (Cross Origin Resource Sharing) is enabled in it. It allows us to send a payload (suppose any data to insert into a database) to a different host with a different port, allowing us to send payload from localhost:3000 to localhost:8000 (server).