**Angular Hands-on**

**1. Angular-T02\_HOL\_001:**

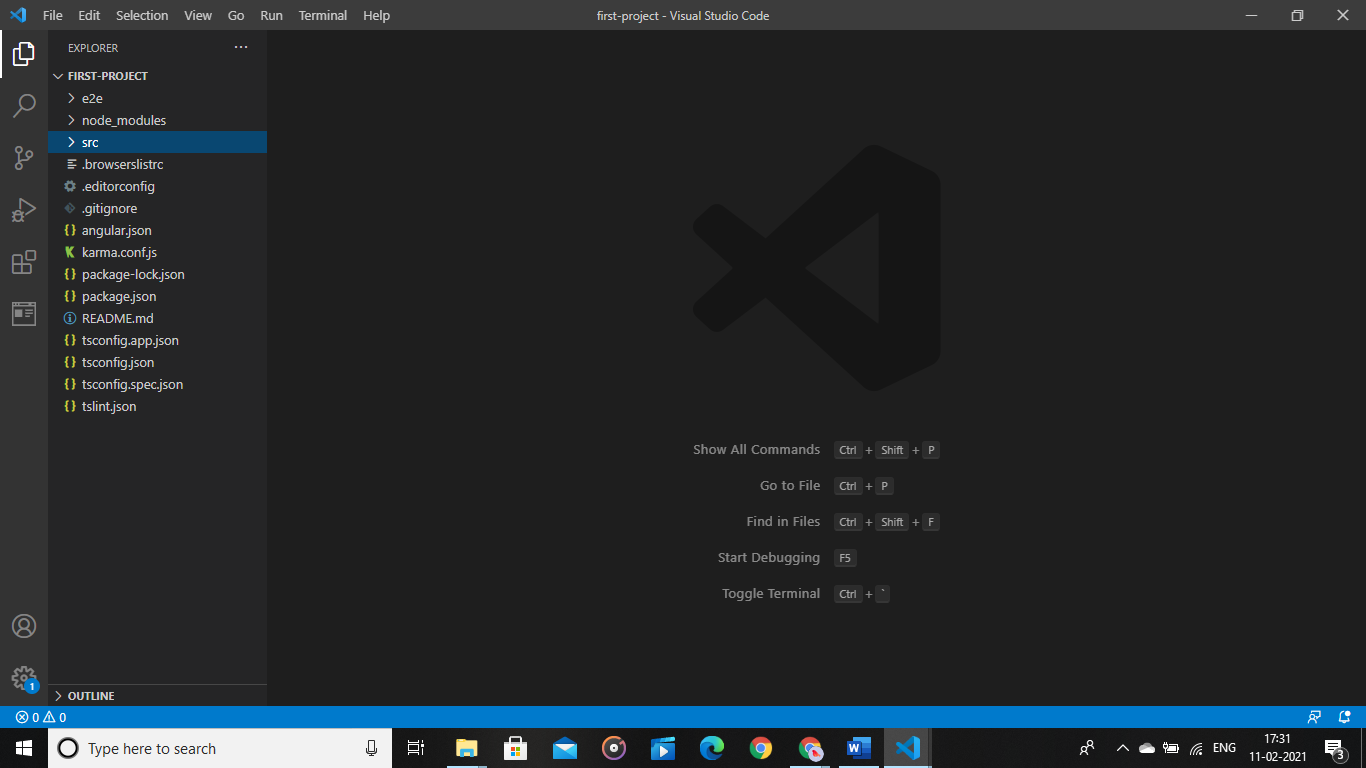
Objectives:

· Employ a very basic Angular application using Visual Studio Code editor

Creating a project:

ng new first-project

ng serve



**2.Angular-T03-HOL-001:**

Objectives:

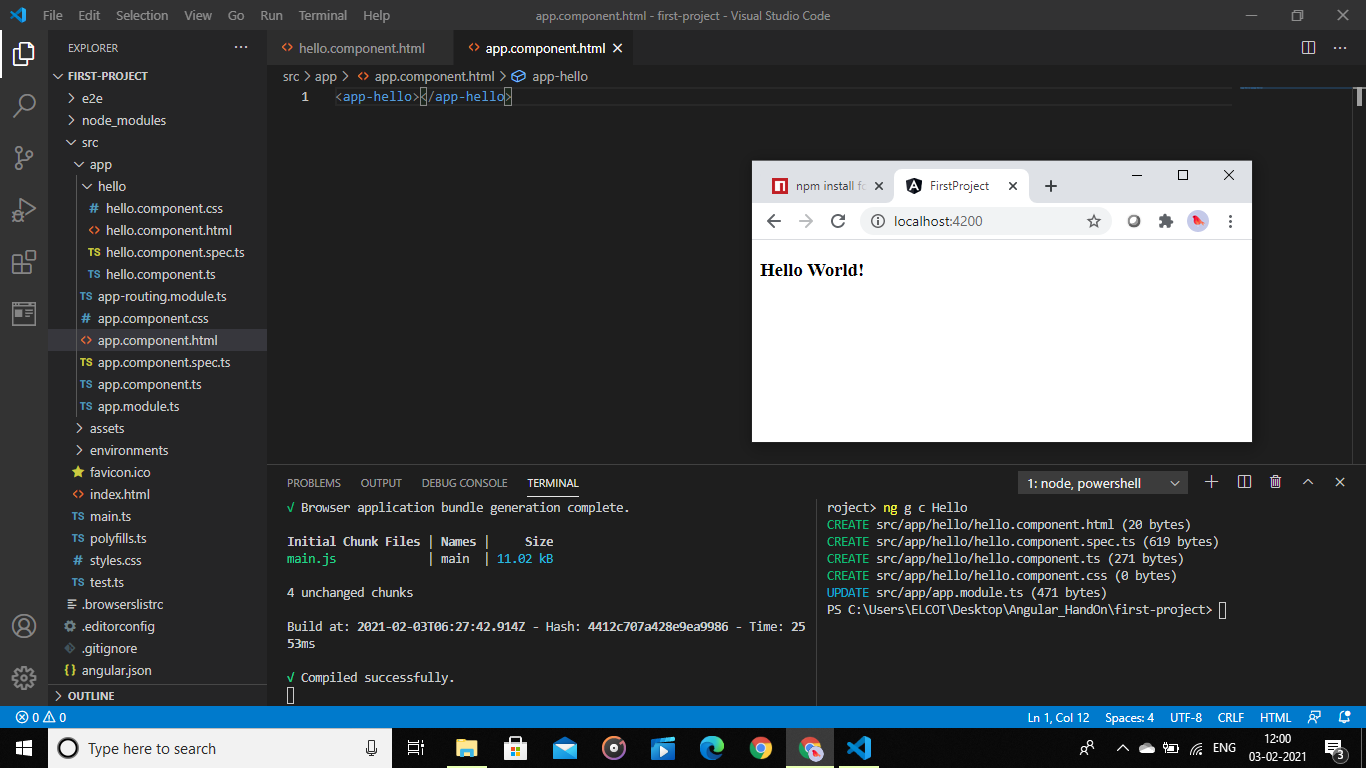
Create a hello component using the following command

ng generate component hello

hello.component.html:

<h1>Hello World!</h1>

Output Screenshot:



**3.Angular-T04-HOL-001:**

Objectives

· Implement TypeScript language features in Angular application

Department.ts:

export interface Department{

    id:number;

    name:string;

}

Skill.ts:

export interface skill{

    id:number;

    name:string;

}

Employee.ts:

export interface Employee{

    id:number;

    name:string;

    salary:number;

    permanent:boolean;

}

Employee-test.ts:  
import {Employee} from './Employee'

import {Department} from './department'

import {Skill} from './skill'

var e1:Employee={

    id:1001,

    name:"John",

    salary:10000,

    permanent:true

}

var d1:Department={

    id:1,

    name:"Payroll"

}

var s1:Skill[];

s1=[{id:1,name:"Javascript"},{id:2,name:"CSS"},{id:3,name:"Html"}];

console.log(`ID:${e1.id}`);

console.log(`Name:${e1.name}`);

console.log(`Salary:${e1.salary}`);

console.log(`Permanent:${e1.permanent}`);

console.log(`Department ID:${d1.id}`);

console.log(`Department Name:${d1.name}`);

for(let skills of s1){

    console.log(skills);

}

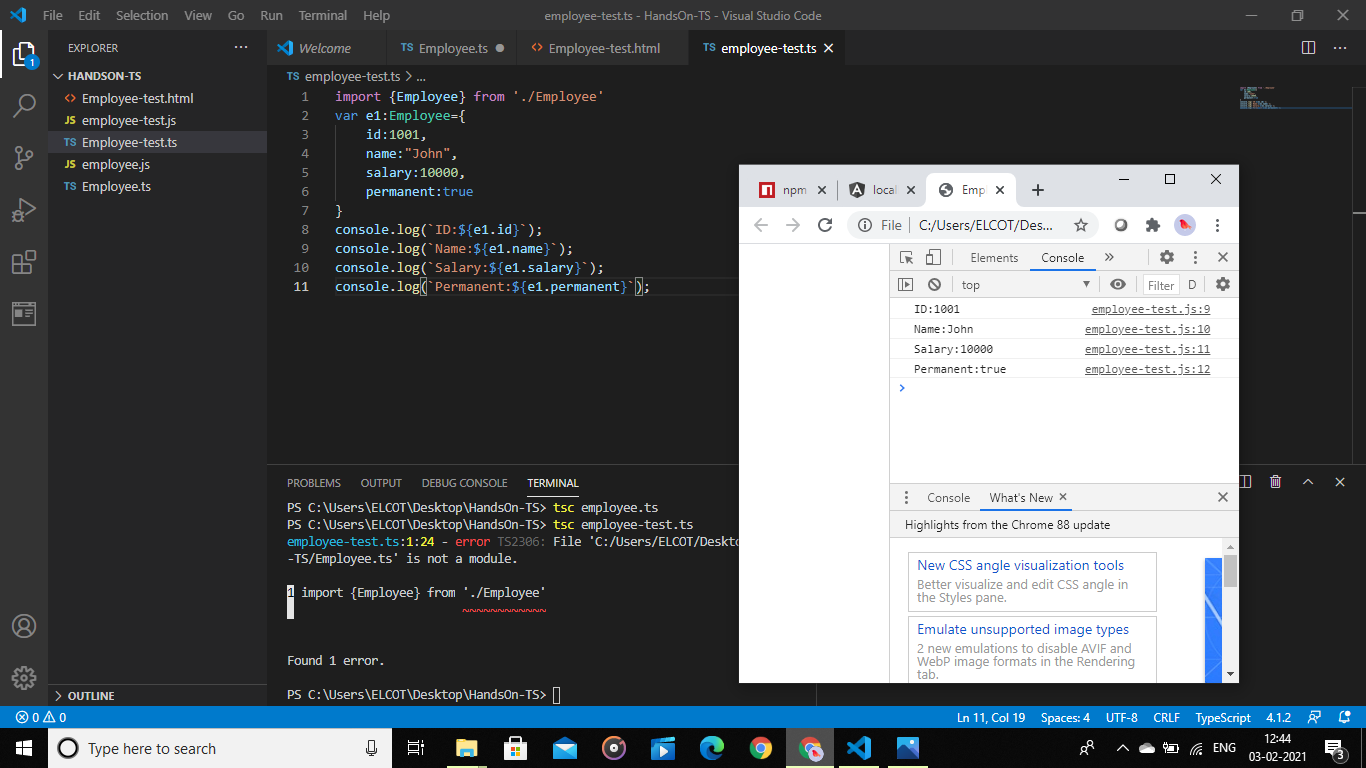
Employee-test.html:  
<script>

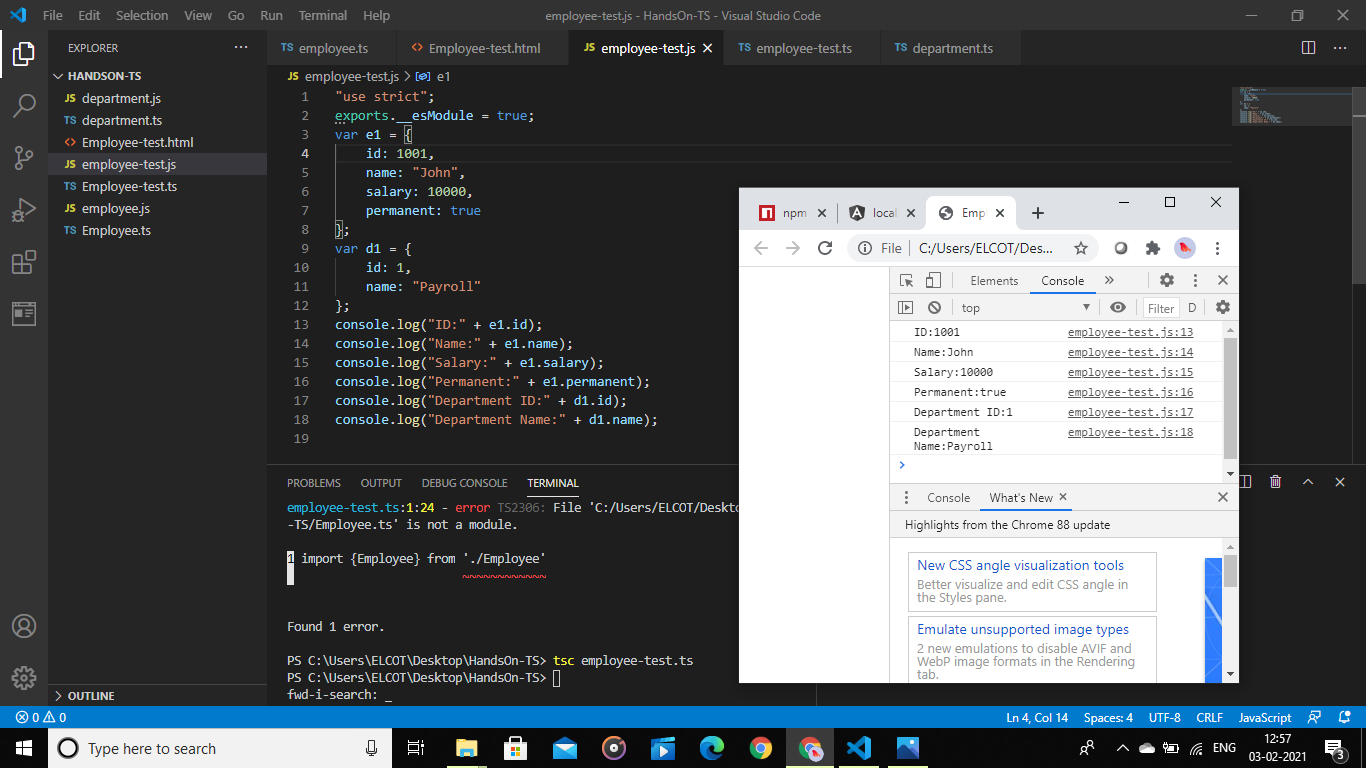
    var exports={};

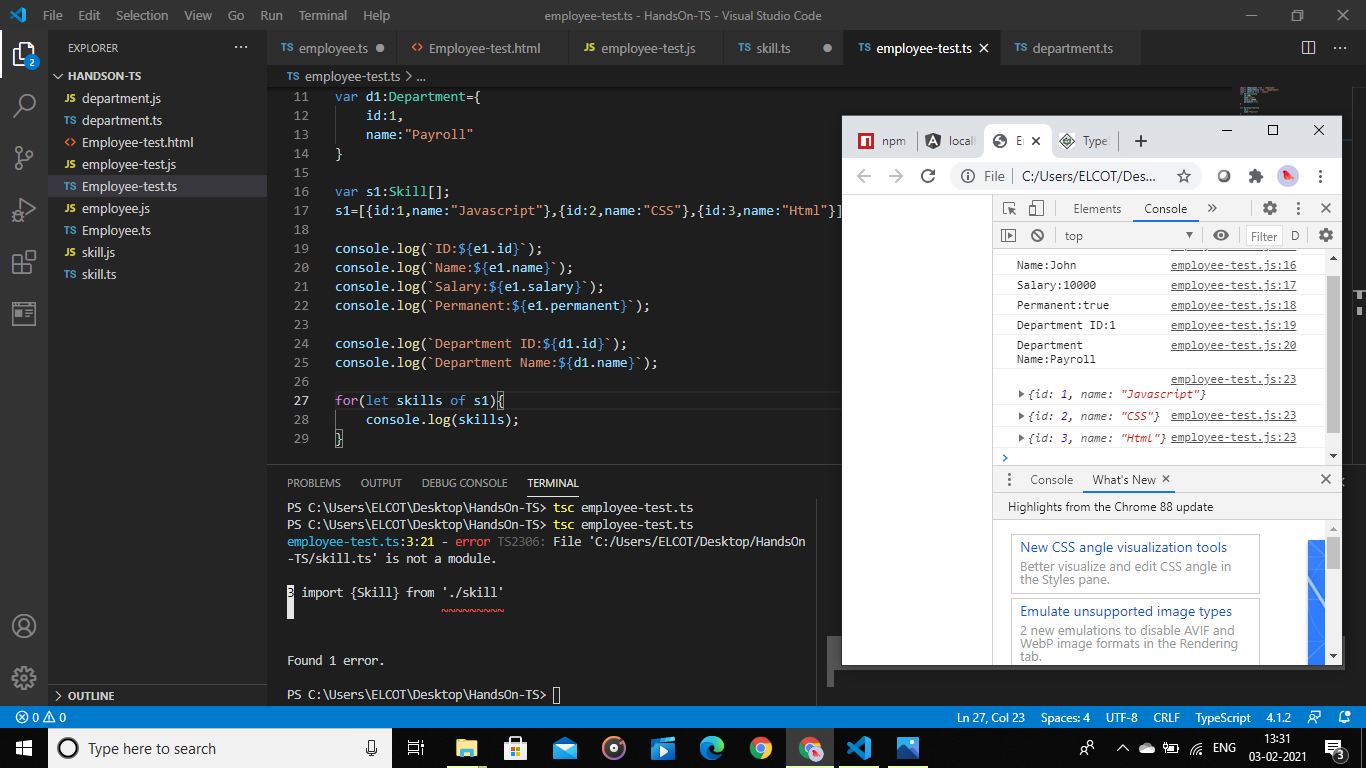
</script>

<script src="employee-test.js"></script>

Output Screenshot:







**4.Angular-T05-HOL\_002:**

Objectives

· Create a Template-Driven-Forms in Angular

Template-driven-form.component.html:

<div class="row">

    <div class="col-md-4 col-md-offset-4" style="margin-top: 50px; border: 1px solid rgb(100,98,98); padding:30px;">

    <form class="form-horizontal" role="form" #f="ngForm" (ngSubmit)="f.form.valid && onFormSubmit(f)">

        <fieldset>

            <legend>Address Details: <strong>Template Driven Form</strong></legend>

                <div class="form-group">

                    <label class="col-sm-2 control-label" for="textinput">Address</label>

                    <div class="col-sm-10">

                        <input type="text" [(ngModel)]="model.address" [ngClass]="{'invalid-data':address.invalid && f.submitted,'valid-data':address.valid && f.submitted}"

                        required name="address" placeholder="Enter Address" class="form-control" #address="ngModel">

                        <div \*ngIf="address.invalid && f.submitted">

                            <span style="color:red">Please Enter Address</span></div>

                    </div>

                </div>

                <div class="form-group">

                    <label class="col-sm-2 control-label" for="textinput">City</label>

                    <div class="col-sm-10">

                        <input type="text" name="city" placeholder="Enter City" class="form-control" ngModel>

                    </div>

                </div>

                <div class="form-group">

                    <label class="col-sm-2 control-label" for="textinput">State</label>

                    <div class="col-sm-10">

                        <input type="text" name="state" placeholder="Enter State" class="form-control" ngModel>

                    </div>

                </div>

                <div class="form-group">

                    <label class="col-sm-2 control-label" for="textinput">PostCode</label>

                    <div class="col-sm-10">

                        <input type="number" name="city" placeholder="Enter Post code" class="form-control" ngModel>

                    </div>

                </div>

                <div class="form-group">

                    <label class="col-sm-2 control-label" for="textinput">Country</label>

                    <div class="col-sm-10">

                        <select class="form-control" name="country"  [(ngModel)]="model.country" [ngClass]="{'invalid-data':country.invalid && f.submitted,'valid-data':country.valid && f.submitted}"

                        required #country="ngModel">

                            <option>---select---</option>

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

                        </select>

                    </div>

                    <div \*ngIf="address.invalid && f.submitted">

                        <span style="color:red">Please Select Country</span></div>

                </div>

                <div class="form-group">

                    <div class="col-sm-2 form-check">

                        <input type="checkbox" name="agreement"  [(ngModel)]="model.agreement" [ngClass]="{'invalid-data':agreement.invalid && f.submitted,'valid-data':agreement.valid && f.submitted}"

                        required #agreement="ngModel" class="form-check-label">

                    </div>

                    <label class="col-sm-10 form-check-label">I agree to tems and conditions</label>

                    <div \*ngIf="address.invalid && f.submitted">

                        <span style="color:red;">Please agree with terms and conditions</span>

                    </div>

                </div>

                <div class="form-group">

                    <div class="col-sm-offset-2 col-sm-10">

                        <div class="pull-right">

                            <button type="submit" class="btn-btn-primary" style="margin:4px">Save</button>

                            <button type="reset" class="btn-btn-default" style="margin:4px">Clear</button>

                        </div>

                    </div>

                </div>

        </fieldset>

    </form>

   </div>

</div>

Template-driven-form.component.ts:

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

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

import {AddressModels} from 'src/app/address-models.model';

@Component({

  selector: 'app-template-driven-form',

  templateUrl: './template-driven-form.component.html',

  styleUrls: ['./template-driven-form.component.css']

})

export class TemplateDrivenFormComponent implements OnInit {

  countryData:any[]=['India','US','UK'];

  model:AddressModels={

    address:'',

    city:'',

    state:'',

    postcode:0,

    country:[],

    agreement:false

  };

  constructor() { }

  ngOnInit(): void {

  }

  onFormSubmit(f:NgForm){

    console.log("Full Address",f.value);

  }

}

Address-model.model.ts:

export interface AddressModels {

        address:string,

        city:string,

        state:string,

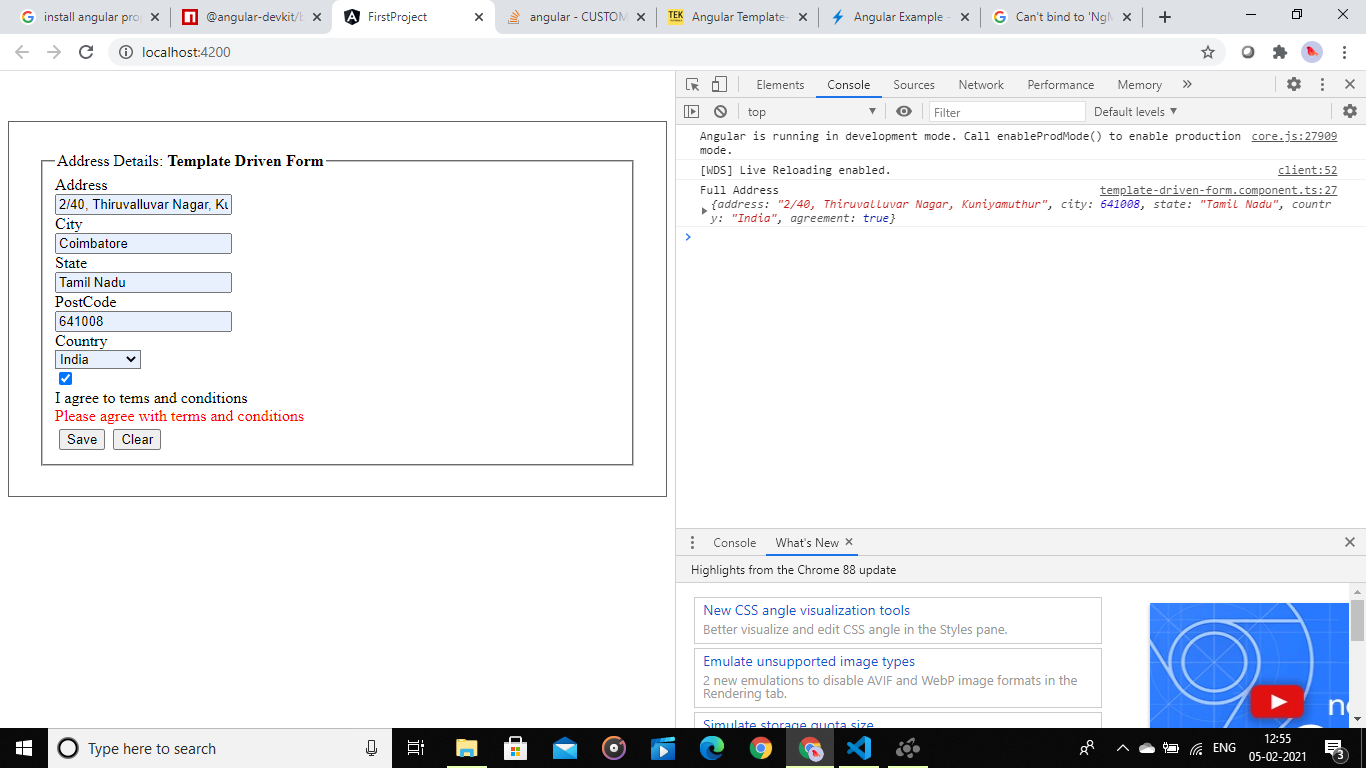
        postcode:number,

        country:any[],

        agreement:boolean

}

Output Screenshot:



**5.Angular T05-HOL\_003:**

**Objectives:**

**·** Create a Template-Driven-Forms in Angular

· Employ form validation in Angular

Template-driven-form.component.html:

<div class="row">

    <div class="col-md-4 col-md-offset-4" style="margin-top: 50px; border: 1px solid rgb(100,98,98); padding:30px;">

    <form class="form-horizontal" role="form" #f="ngForm" (ngSubmit)="f.form.valid && onFormSubmit(f)">

        <fieldset>

            <legend>Address Details: <strong>Template Driven Form</strong></legend>

                <div class="form-group">

                    <label class="col-sm-2 control-label" for="textinput">Address</label>

                    <div class="col-sm-10">

                        <input type="text" [(ngModel)]="model.address" [ngClass]="{'invalid-data':address.invalid && f.submitted,'valid-data':address.valid && f.submitted}"

                        required name="address" placeholder="Enter Address" class="form-control" #address="ngModel">

                        <div \*ngIf="address.invalid && f.submitted">

                            <span style="color:red">Please Enter Address</span></div>

                    </div>

                </div>

                <div class="form-group">

                    <label class="col-sm-2 control-label" for="textinput">City</label>

                    <div class="col-sm-10">

                        <input type="text" name="city" placeholder="Enter City" class="form-control" ngModel>

                    </div>

                </div>

                <div class="form-group">

                    <label class="col-sm-2 control-label" for="textinput">State</label>

                    <div class="col-sm-10">

                        <input type="text" name="state" placeholder="Enter State" class="form-control" ngModel>

                    </div>

                </div>

                <div class="form-group">

                    <label class="col-sm-2 control-label" for="textinput">PostCode</label>

                    <div class="col-sm-10">

                        <input type="number" name="city" placeholder="Enter Post code" class="form-control" ngModel>

                    </div>

                </div>

                <div class="form-group">

                    <label class="col-sm-2 control-label" for="textinput">Country</label>

                    <div class="col-sm-10">

                        <select class="form-control" name="country"  [(ngModel)]="model.country" [ngClass]="{'invalid-data':country.invalid && f.submitted,'valid-data':country.valid && f.submitted}"

                        required #country="ngModel">

                            <option>---select---</option>

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

                        </select>

                    </div>

                    <div \*ngIf="address.invalid && f.submitted">

                        <span style="color:red">Please Select Country</span></div>

                </div>

                <div class="form-group">

                    <div class="col-sm-2 form-check">

                        <input type="checkbox" name="agreement"  [(ngModel)]="model.agreement" [ngClass]="{'invalid-data':agreement.invalid && f.submitted,'valid-data':agreement.valid && f.submitted}"

                        required #agreement="ngModel" class="form-check-label">

                    </div>

                    <label class="col-sm-10 form-check-label">I agree to tems and conditions</label>

                    <div \*ngIf="address.invalid && f.submitted">

                        <span style="color:red;">Please agree with terms and conditions</span>

                    </div>

                </div>

                <div class="form-group">

                    <div class="col-sm-offset-2 col-sm-10">

                        <div class="pull-right">

                            <button type="submit" class="btn-btn-primary" style="margin:4px">Save</button>

                            <button type="reset" class="btn-btn-default" style="margin:4px">Clear</button>

                        </div>

                    </div>

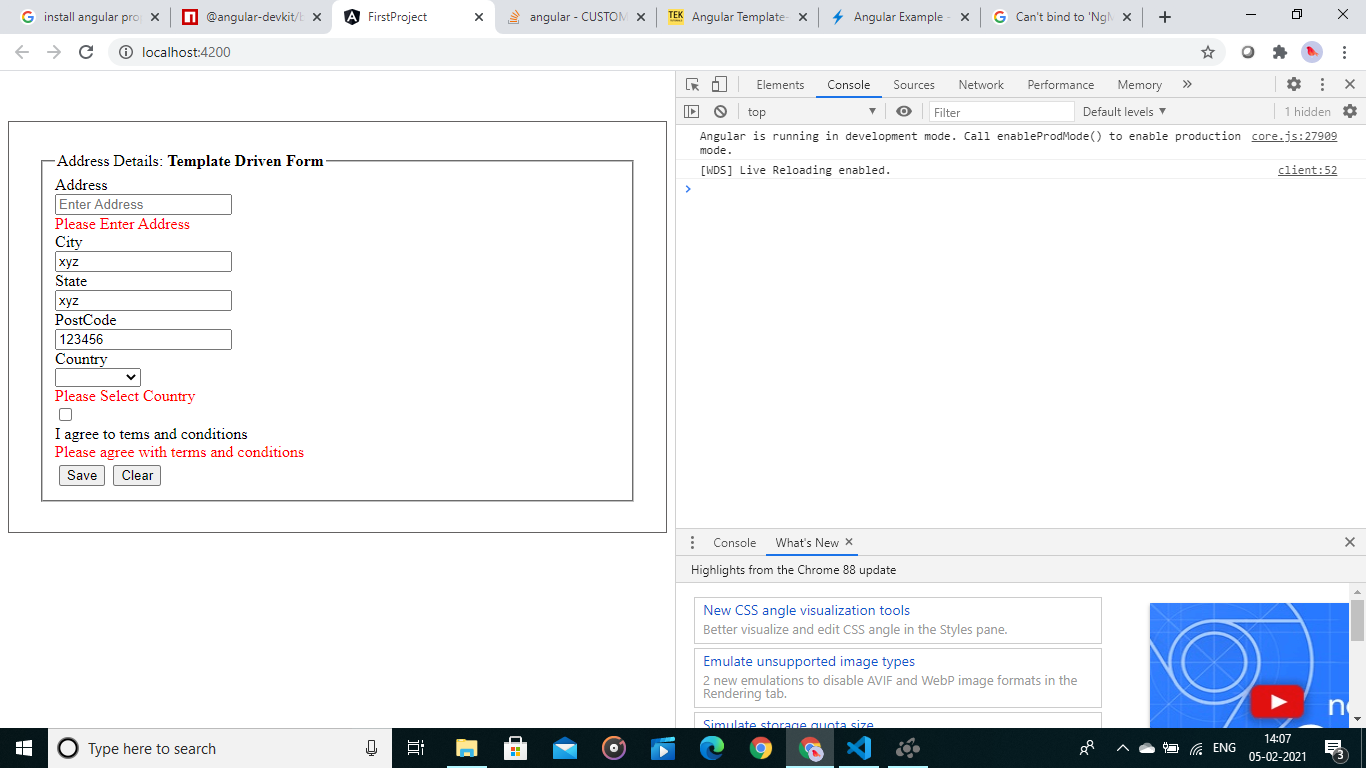
                </div>

        </fieldset>

    </form>

    </div>

</div>



**6.Angular T06-HOL\_001:**

Implement component communication in an Angular application

Create parent, child and sibling component

Parent to Child: Sharing Data via Input

Parent.component.ts:

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

@Component({

selector: 'app-parent',

template: `

<app-child [childMessage]="parentMessage"></app-child>

`,

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

})

export class ParentComponent{

parentMessage = "Hello World"

constructor() { }

child.component.ts:

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

@Component({

selector: 'app-child',

template: `

{{ message }}

`,

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

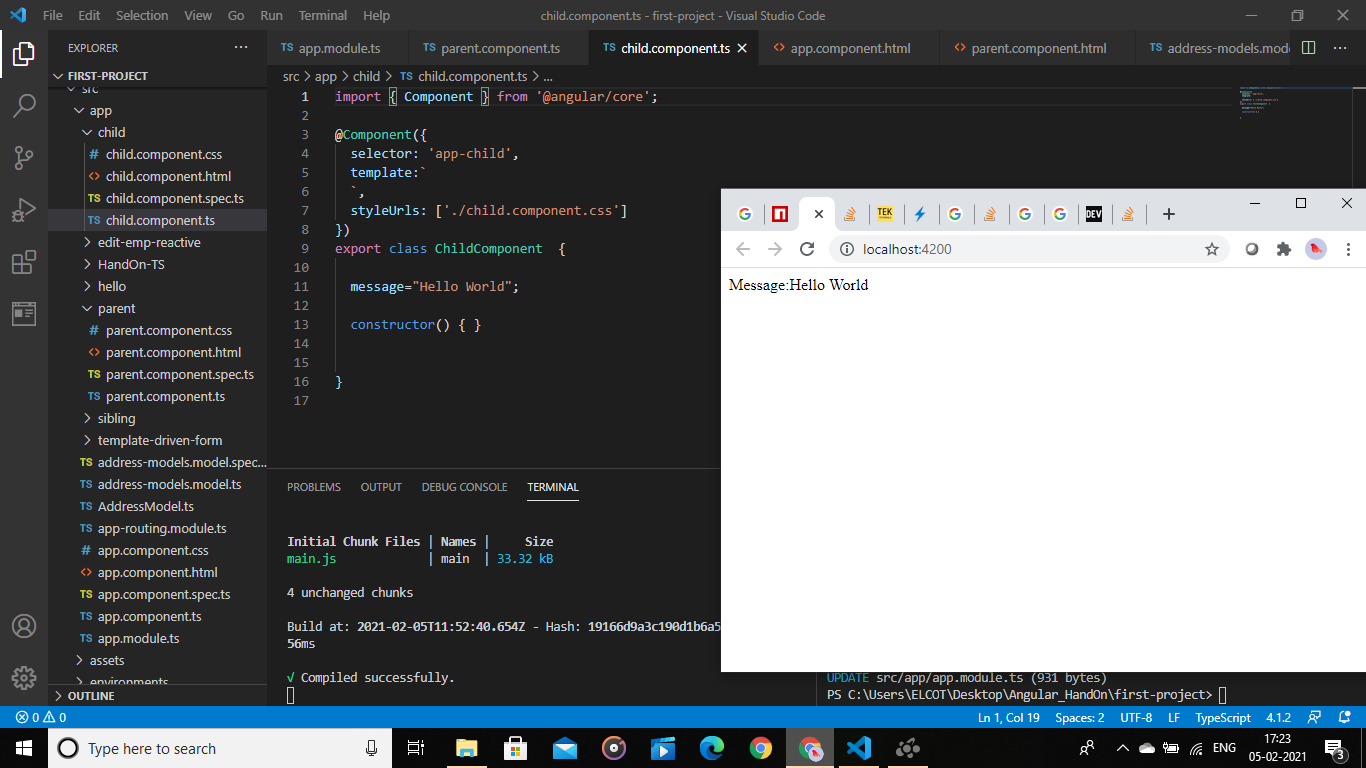
})

export class ChildComponent {

@Input() childMessage: string;

constructor() { }

}



Child to Parent: Sharing Data via ViewChild:

Paren.component.ts:

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

import { ChildComponent } from "../child/child.component";

@Component({

selector: 'app-parent',

template: `

Message: {{ message }}

<app-child></app-child>

`,

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

})

export class ParentComponent implements AfterViewInit {

@ViewChild(ChildComponent) child;

constructor() { }

message:string;

ngAfterViewInit() {

this.message = this.child.message

}

}

Child.component.html:

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

@Component({

selector: 'app-child',

template: `

`,

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

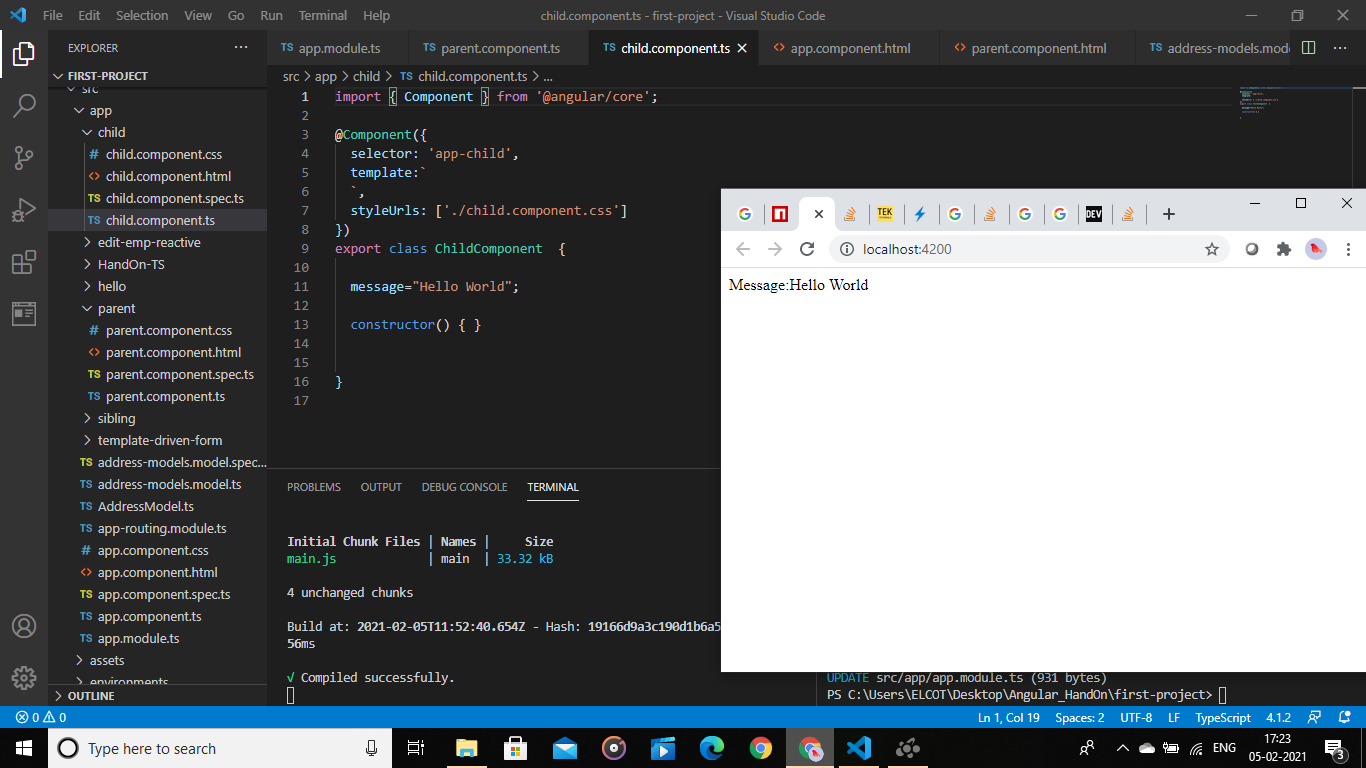
})

export class ChildComponent {

message = 'Hello World!';

constructor() { }

}



Child to Parent: Sharing Data via Output() and EventEmitter:

Parent.component.ts:

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

@Component({

selector: 'app-parent',

template: `

Message: {{message}}

<app-child (messageEvent)="receiveMessage($event)"></app-child>

`,

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

})

export class ParentComponent {

constructor() { }

message:string;

receiveMessage($event) {

this.message = $event

}

}

Child.component.ts:

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

@Component({

selector: 'app-child',

template: `

<button (click)="sendMessage()">Send Message</button>

`,

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

})

export class ChildComponent {

message: string = "Hello World!"

@Output() messageEvent = new EventEmitter<string>();

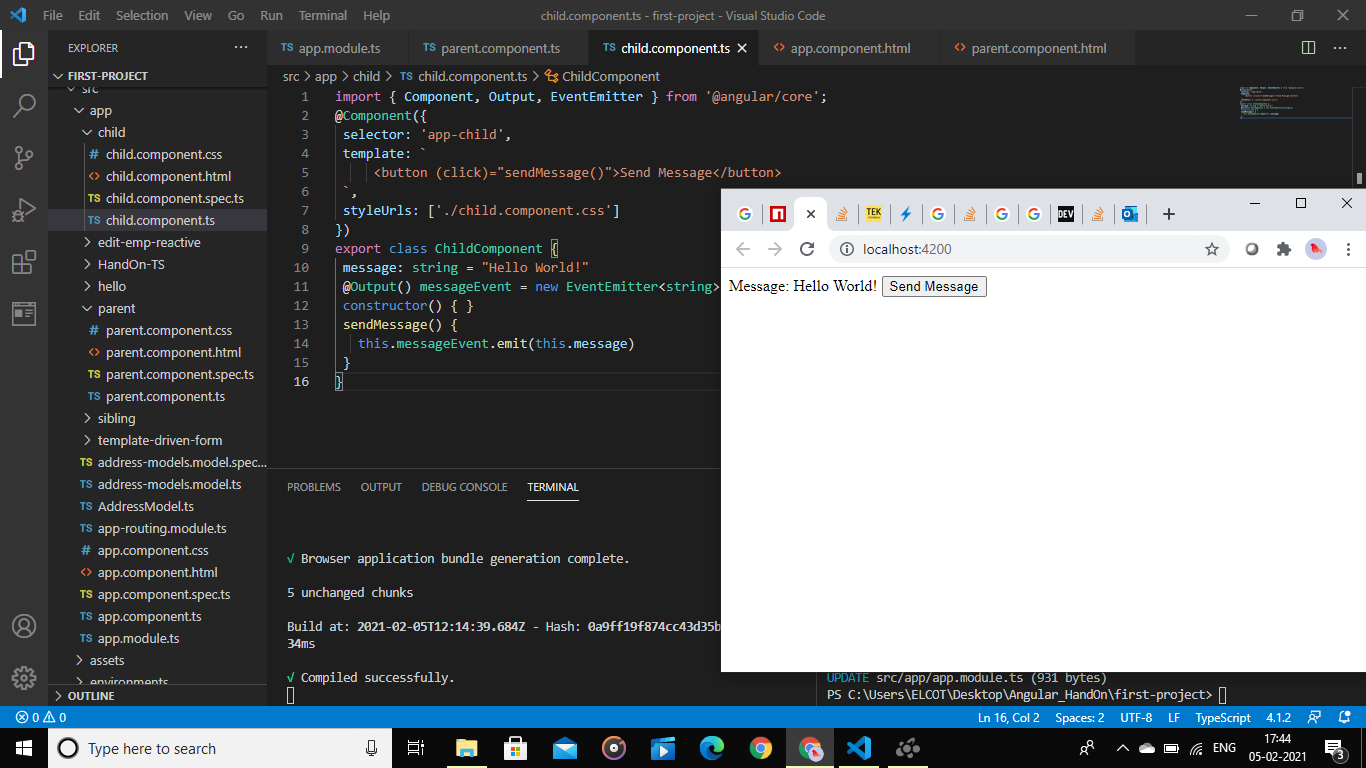
constructor() { }

sendMessage() {

this.messageEvent.emit(this.message)

}

}



**7.Angular-T06-HOL-002:**

**Objectives:**

Employ observables that provides support for passing messages between parts of an Angular application.

Unrelated Components: Sharing Data with a Service:

Sibling.component.ts:  
import { Component, OnInit,OnDestroy } from '@angular/core';

import { DataService } from "../data.service";

import { Subscription } from 'rxjs';

@Component({

 selector: 'app-sibling',

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

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

})

export class SiblingComponent implements OnInit, OnDestroy {

 message!:string;

 subscription!: Subscription ;

 constructor(private data: DataService) {

  }

 ngOnInit() {

   this.subscription = this.data.currentMessage.subscribe(message => this.message = message);

 }

 ngOnDestroy() {

   this.subscription.unsubscribe();

 }

 newMessage(){

   this.data.changeMessage("Hello from sibling");

 }

}

Parent.component.ts:  
import { Component, OnInit,OnDestroy } from '@angular/core';

import { DataService } from "../data.service";

import { Subscription } from 'rxjs';

@Component({

 selector: 'app-parent',

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

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

})

export class ParentComponent implements OnInit, OnDestroy {

 message!:string;

 subscription!:Subscription;

 constructor(private data: DataService) { }

 ngOnInit() {

   this.subscription = this.data.currentMessage.subscribe(message => this.message = message)

 }

 ngOnDestroy() {

   this.subscription.unsubscribe();

 }

}

Data-service.ts:

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

import { BehaviorSubject } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

@NgModule({

  providers:[]

})

export class DataService {

 private messageSource = new BehaviorSubject('default message');

 currentMessage = this.messageSource.asObservable();

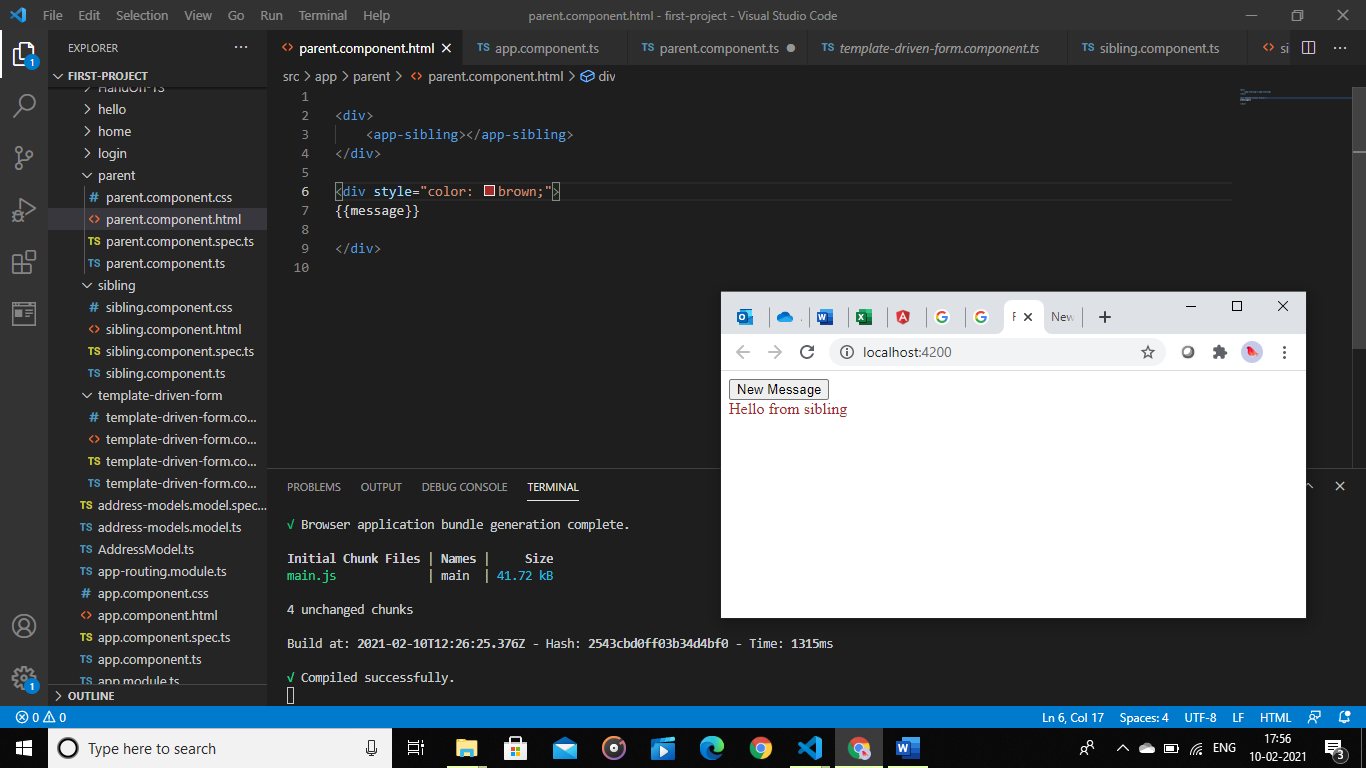
 constructor() { }

 changeMessage(message: string) {

   this.messageSource.next(message)

 }

}



**8.Angular-T07-HOL\_001:**

**Objective:**

Implement navigation in an Angular application

Menu.component.html:

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

import { BehaviorSubject } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

@NgModule({

  providers:[]

})

export class DataService {

 private messageSource = new BehaviorSubject('default message');

 currentMessage = this.messageSource.asObservable();

 constructor() { }

 changeMessage(message: string) {

   this.messageSource.next(message)

 }

}

Router-config.ts:  
import {Routes} from '@angular/router';

import {HomeComponent} from './components/home/home.component';

import {AboutComponent} from './components/about/about.component';

import {LoginComponent} from './components/login/login.component';

import {MenuComponent} from './menu/menu.component';

export const appRoutes:Routes=[

    {path:'home',component:HomeComponent},

    {path:'about',component:AboutComponent},

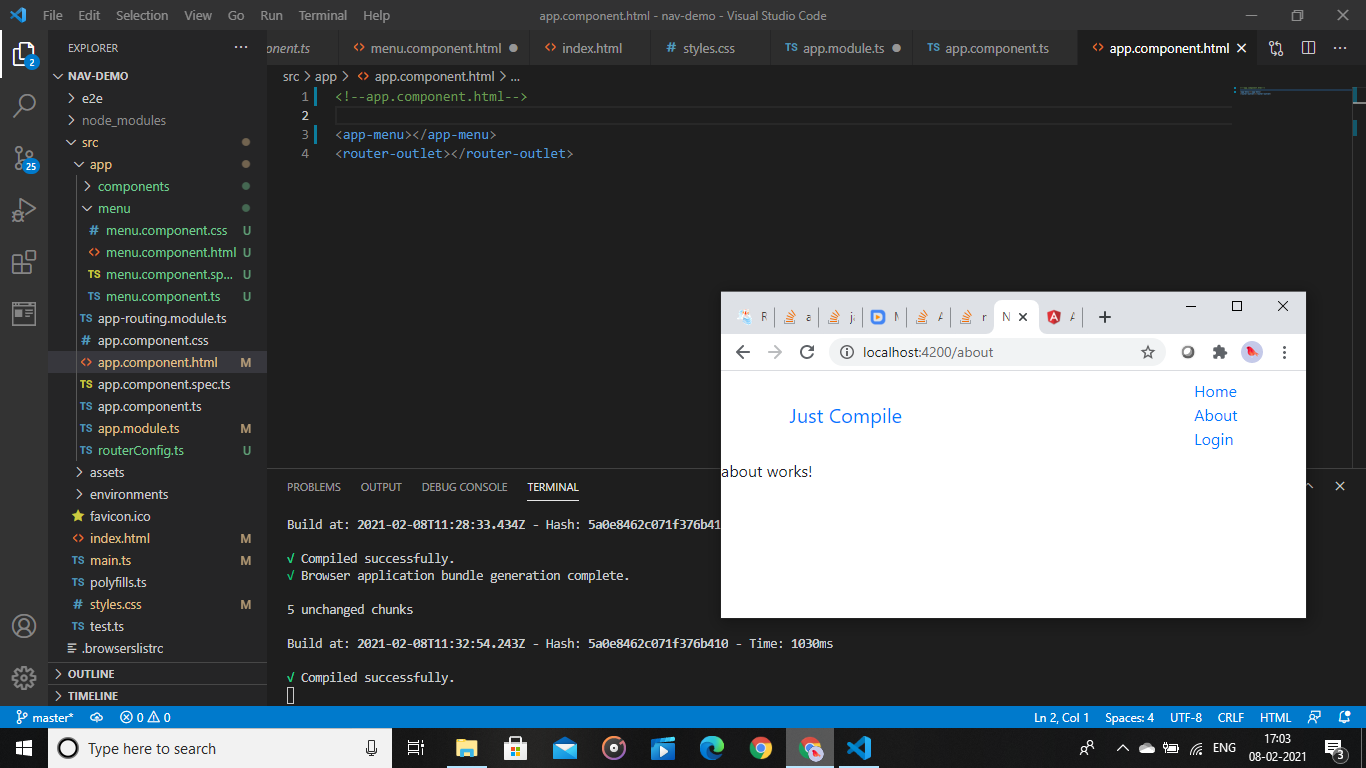
    {path:'login',component:LoginComponent},

    {path:'menu',component:MenuComponent}

]

App.component.html:  
<app-menu></app-menu>

<router-outlet></router-outlet>



**9.Angular-T08-HOL\_003:**

**Objective:**

Implement error handling in Angular application using HttpErrorResponse

Api-service.ts:

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

import { HttpClient, HttpErrorResponse } from "@angular/common/http";

@Injectable({

providedIn: 'root'

})

export class ApiService {

private SERVER = "http://server.com/api/products";

constructor(private httpClient: HttpClient) { }

handleError(error: HttpErrorResponse) {

let errorMessage = 'Unknown error!';

if (error.error instanceof ErrorEvent) {

// Client-side errors

errorMessage = `Error: ${error.error.message}`;

} else {

// Server-side errors

errorMessage = `Error Code: ${error.status}\nMessage: ${error.message}`;

}

window.alert(errorMessage);

return throwError(errorMessage);

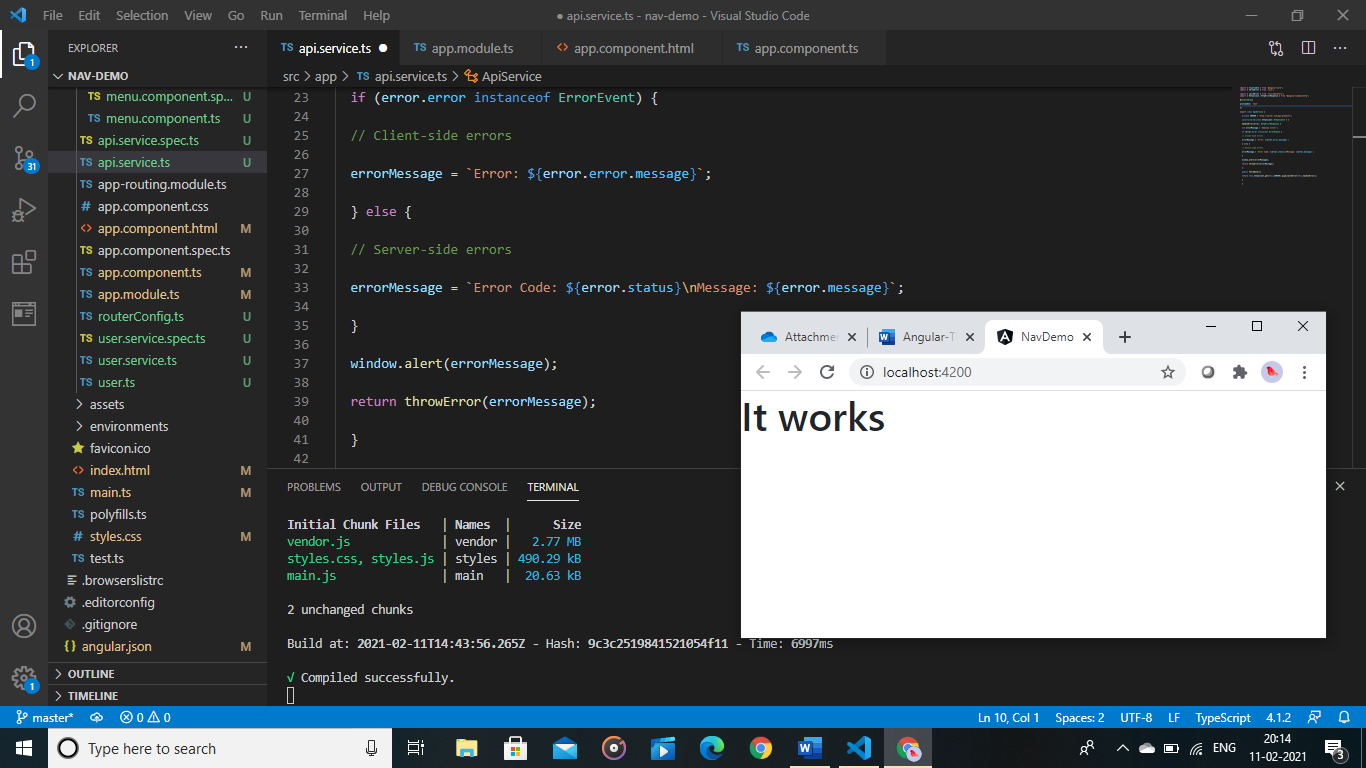
}

public fetchData(){

return this.httpClient.get(this.SERVER).pipe(catchError(this.handleError));;

}

}



**10. Angular-T08-HOL\_002:**

**Objective:**

Implement asynchronous or callback-based code in an Angular Application using RxJS.

App.component.html:

<b>Observables Demo</b>

<h6 style="margin-bottom: 0">VALUES:</h6>

<div \*ngFor="let value of numArray"> {{ value }}</div>

<div style="margin-bottom: 0">ERRORS: {{ errors }}</div>

<div style="margin-bottom: 0">finished: {{ finished }}</div>

<button style="margin-top: 2rem;" (click)="fetchData()">Fetch Data</button>

App.component.ts:

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

import { Observable } from 'rxjs';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

data!: Observable<number>;

numArray: number[] = [];

errors!: boolean;

finished!: boolean;

fetchData() {

this.data = new Observable(observer => {

setTimeout(() => { observer.next(10); }, 1000),

setTimeout(() => { observer.next(20); }, 2000),

setTimeout(() => { observer.complete(); }, 3000);

});

const sub = this.data.subscribe((value) => this. numArray.push(value),

error => this.errors = true,

() => this.finished = true);

}

}

