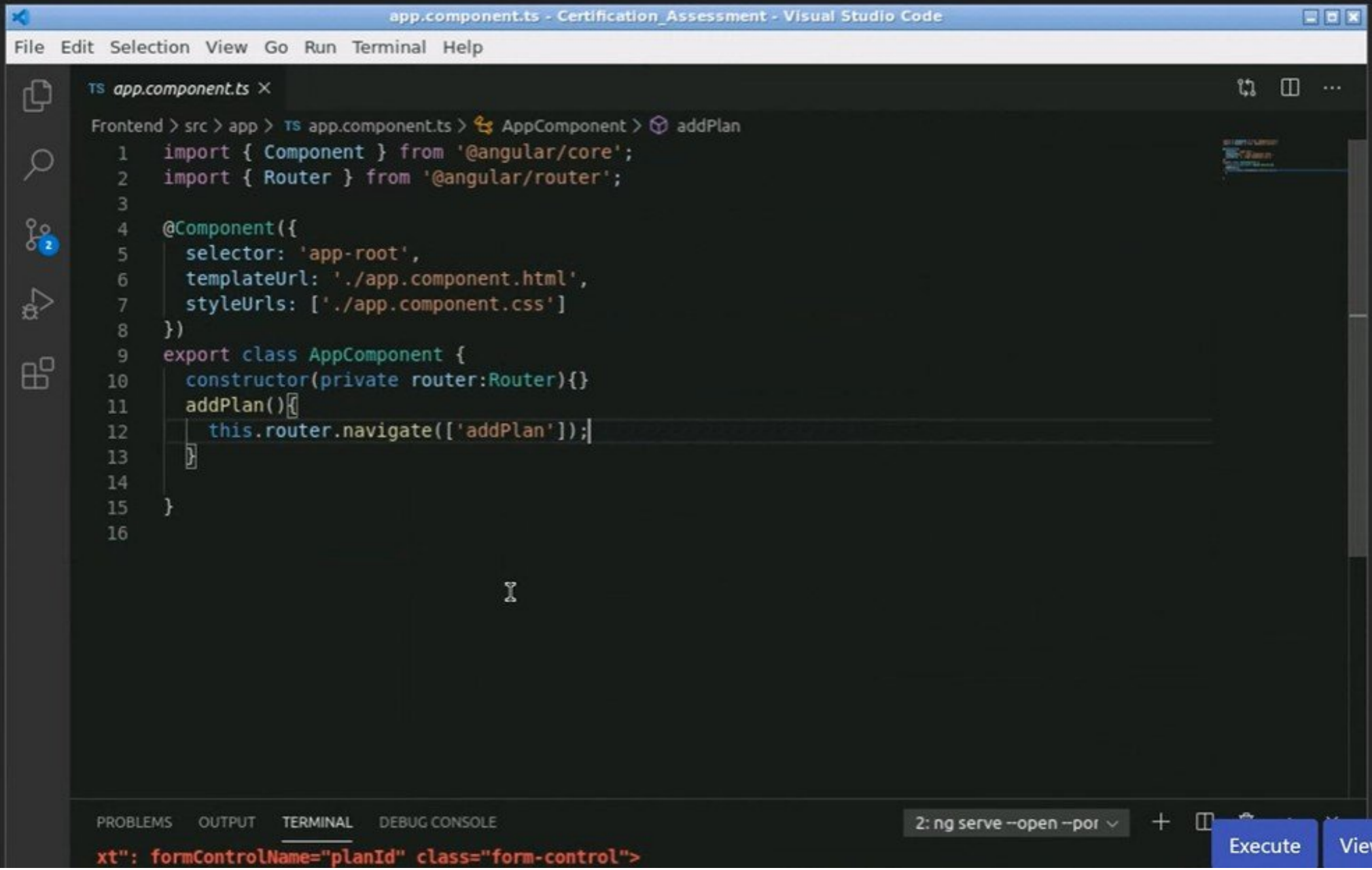
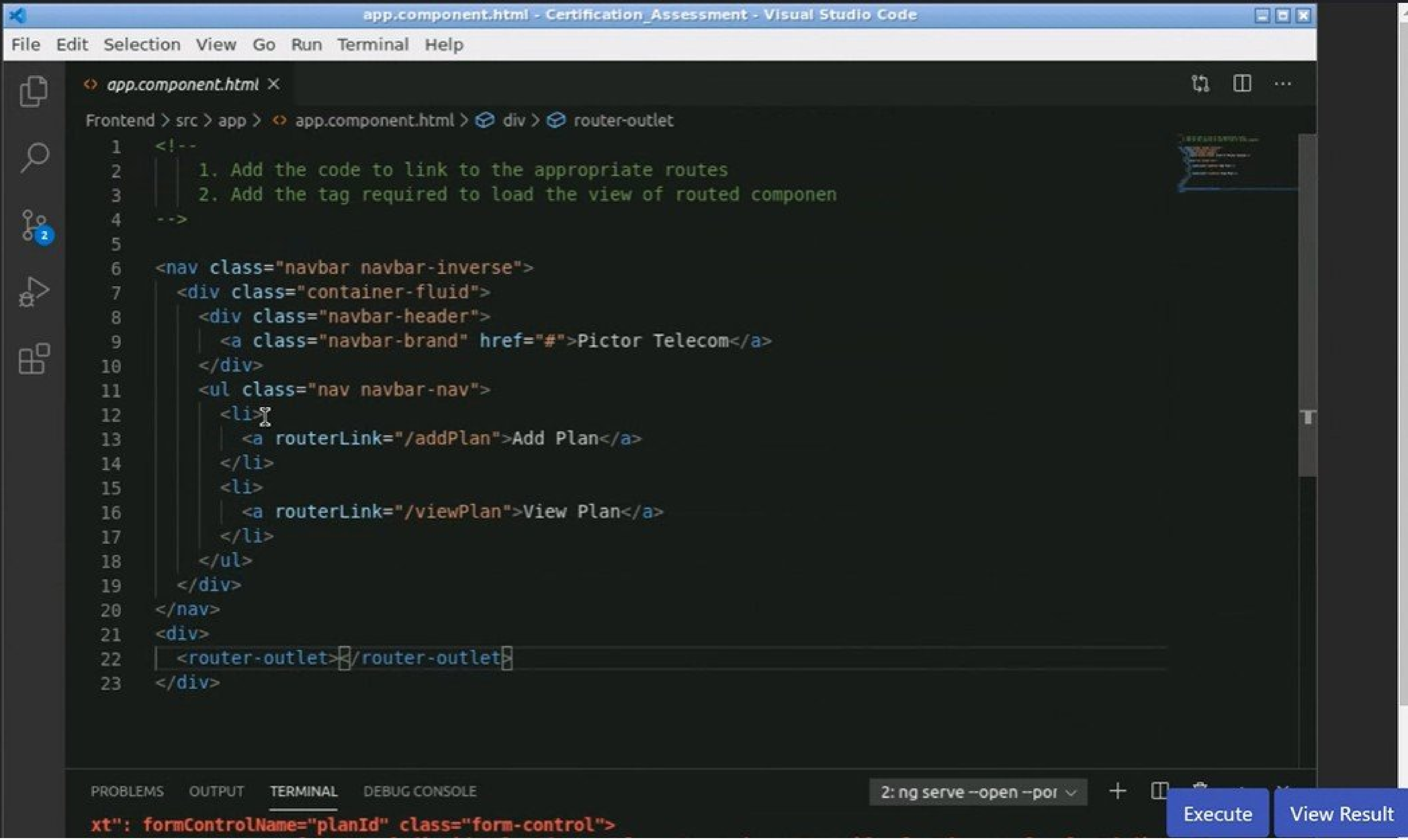
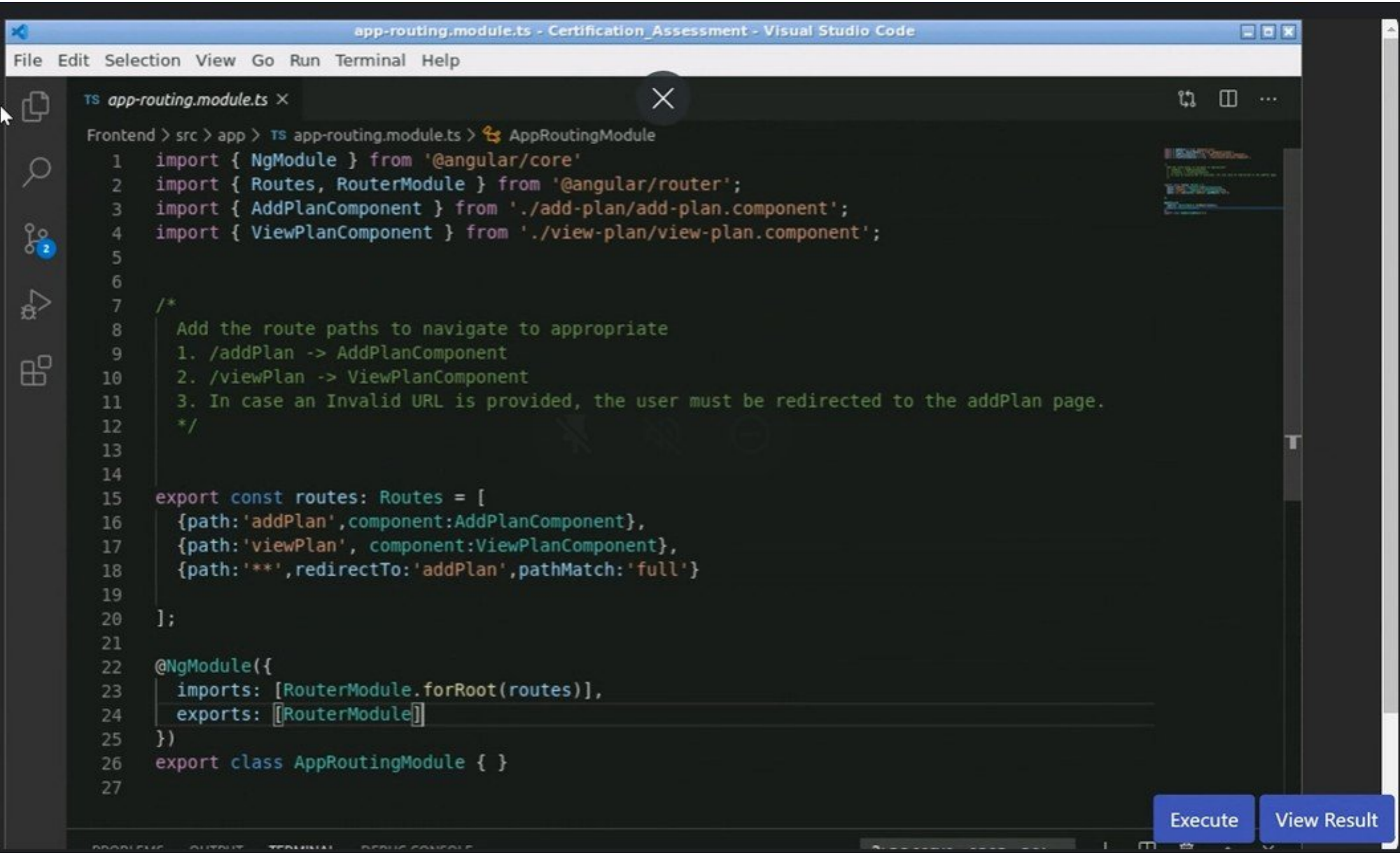

app.component.ts - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
app.component.ts
Frontend > src > app > app.component.ts > AppComponent > addPlan
import { Component } from '@angular/core';
import { Router } from '@angular/router';
@Component({
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']
})
export class AppComponent {
constructor(private router:Router){}
addPlan(){
this.router.navigate(['addPlan']);
}
}
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
2: ng serve --open --por
xt": formControlName="planId" class="form-control">
Execute
View Result

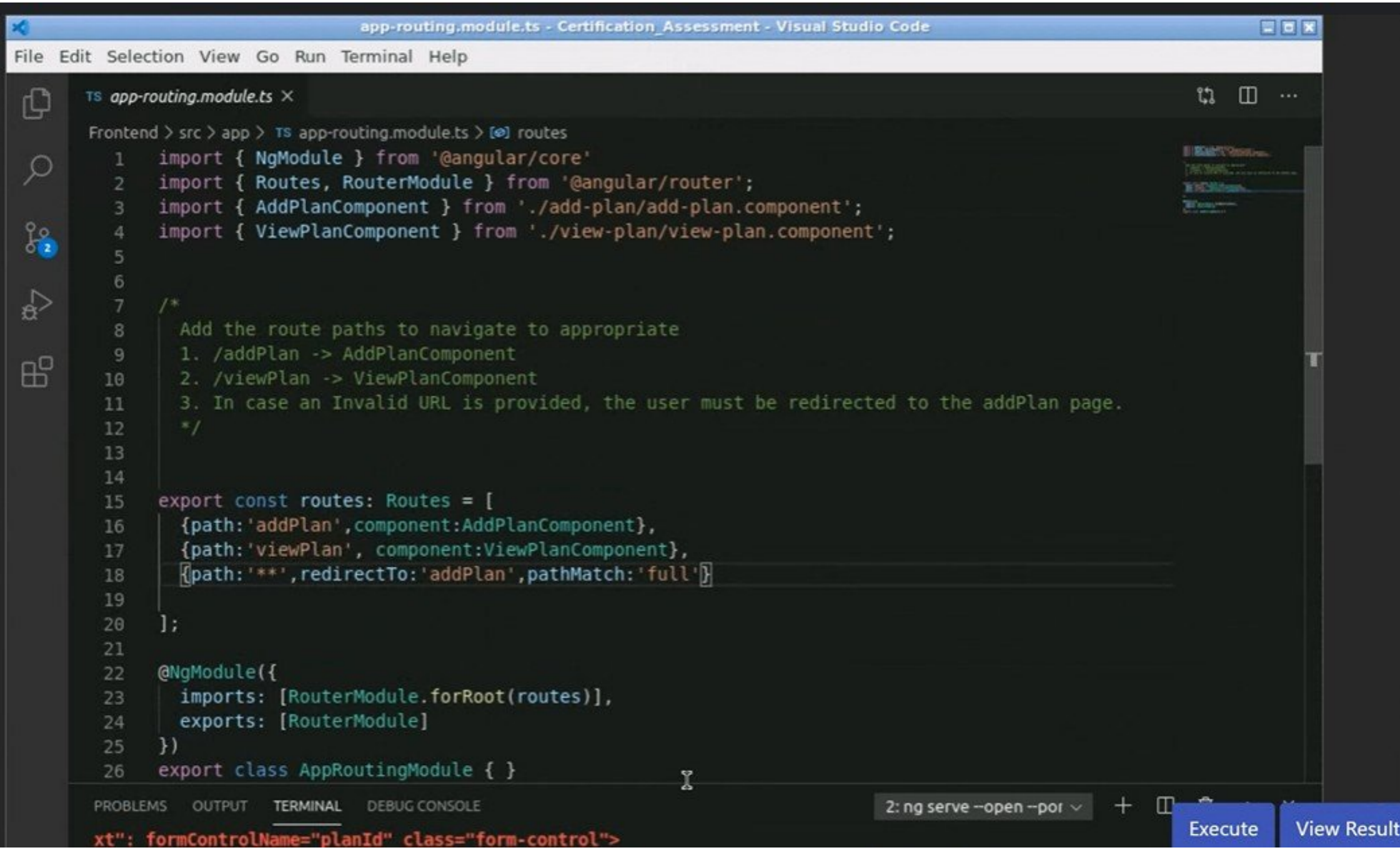

app.component.html - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
app.component.html
Frontend > src > app > app.component.html > div > router-outlet
<!--
1. Add the code to link to the appropriate routes
2. Add the tag required to load the view of routed componen
-->
<nav class="navbar navbar-inverse">
<div class="container-fluid">
<div class="navbar-header">
Pictor Telecom
</div>
<ul class="nav navbar-nav">

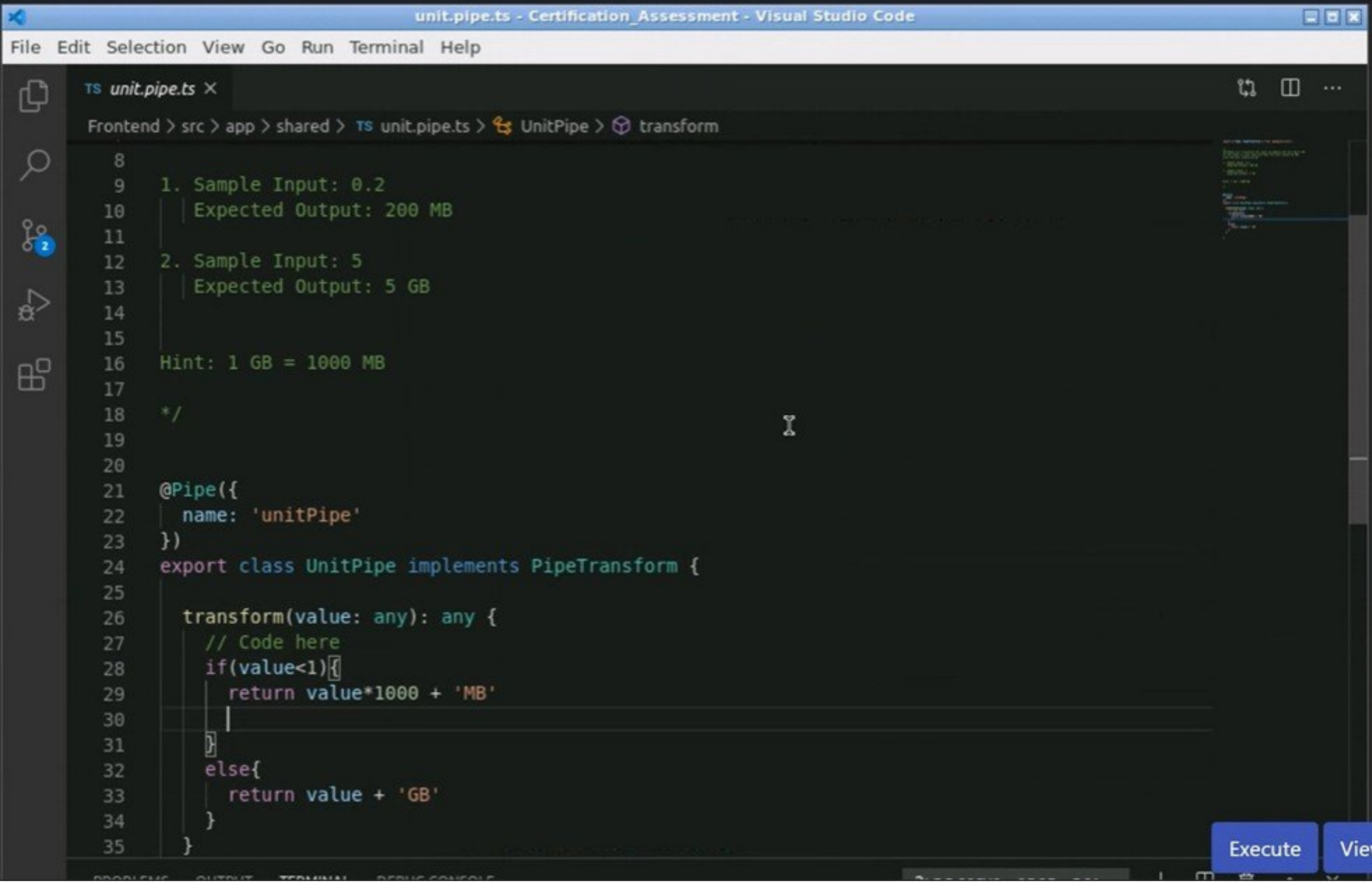
<a routerLink="/addPlan">Add Plan

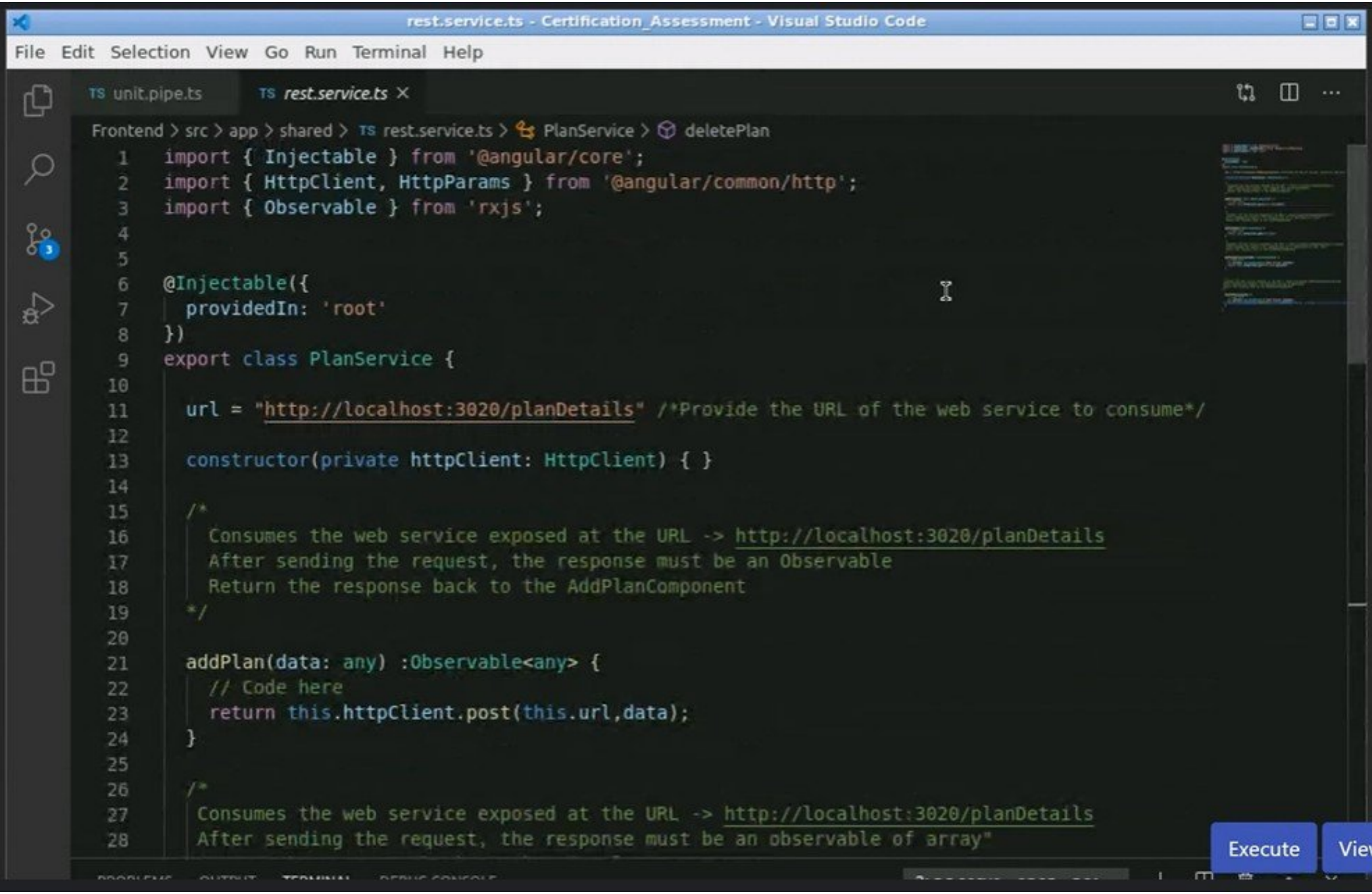
<a routerLink="/viewPlan">View Plan

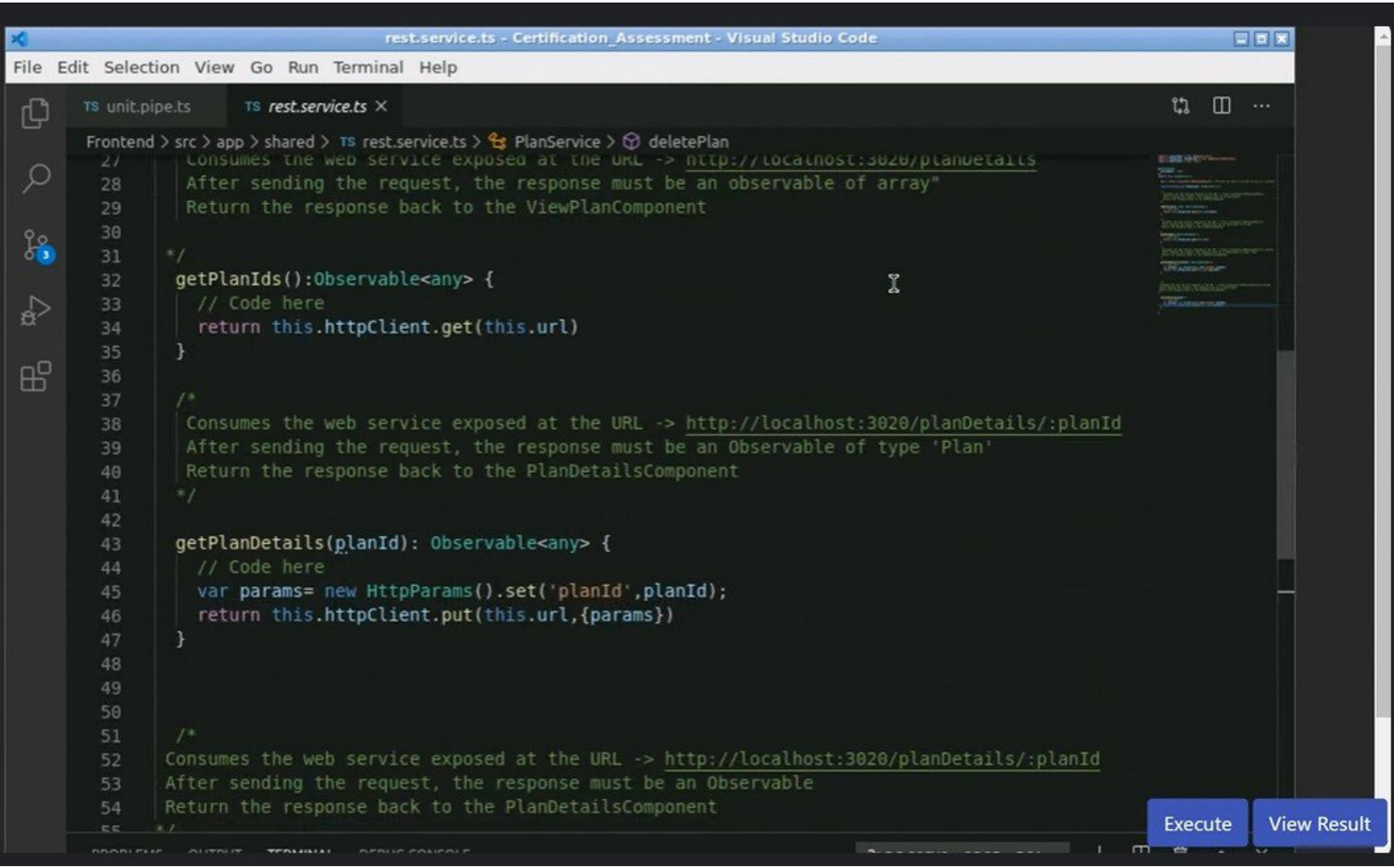
</div>
</nav>
<div>
<router-outlet></router-outlet>
</div>
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
2: ng serve --open --por
xt": formControlName="planId" class="form-control">
Execute
View Result

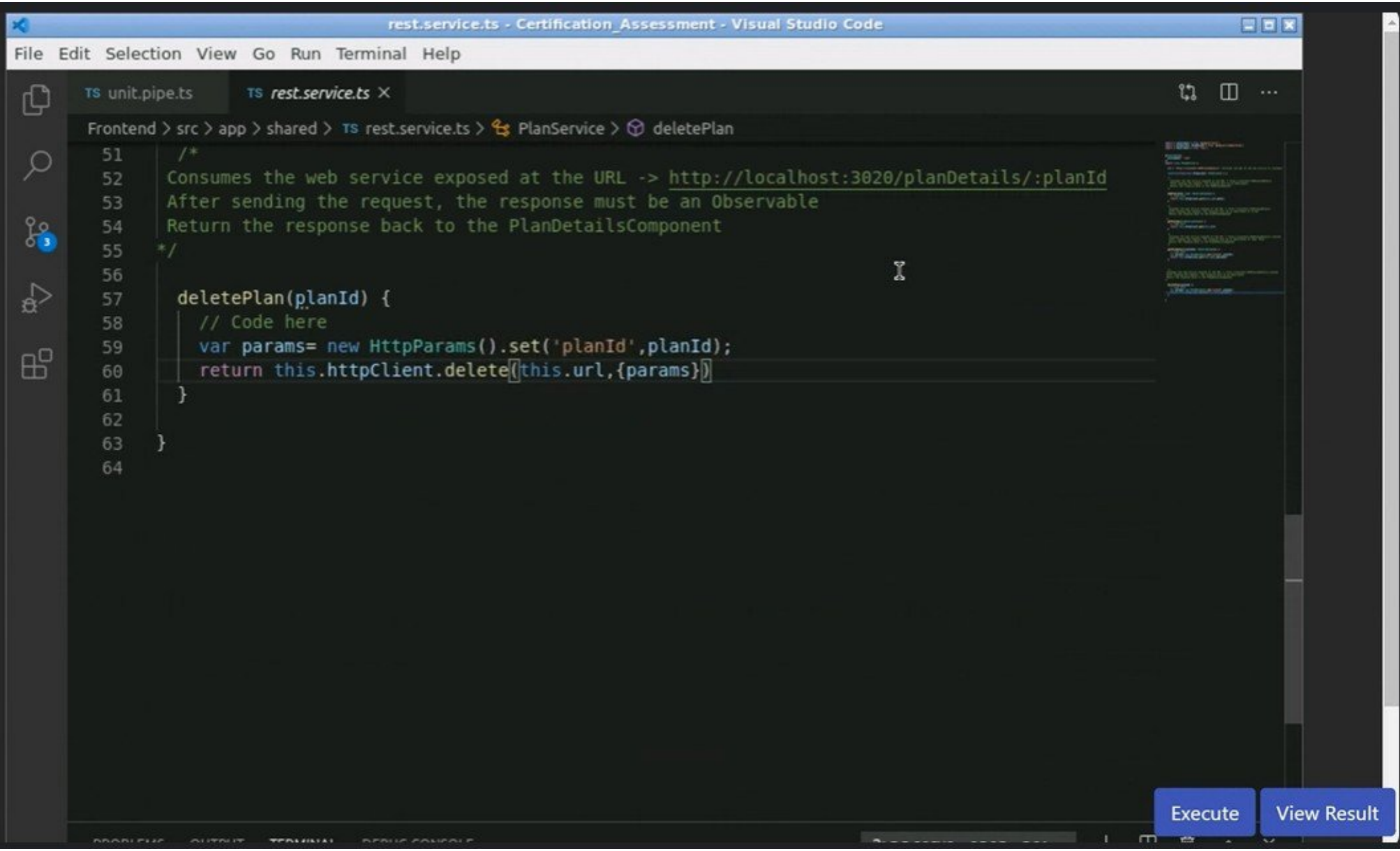

app-routing.module.ts - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
app-routing.module.ts
Frontend > src > app > app-routing.module.ts > AppRoutingModule
import { NgModule } from '@angular/core'
import { Routes, RouterModule } from '@angular/router';
import { AddPlanComponent } from './add-plan/add-plan.component';
import { ViewPlanComponent } from './view-plan/view-plan.component';
/*
Add the route paths to navigate to appropriate
1. /addPlan -> AddPlanComponent
2. /viewPlan -> ViewPlanComponent
3. In case an Invalid URL is provided, the user must be redirected to the addPlan page.
*/
export const routes: Routes = [
{path:'addPlan',component:AddPlanComponent},
{path:'viewPlan', component:ViewPlanComponent},
{path:'**',redirectTo:'addPlan',pathMatch:'full'}
];
@NgModule({
imports: [RouterModule.forRoot(routes)],
exports: [RouterModule]
})
export class AppRoutingModule { }
Execute
View Result

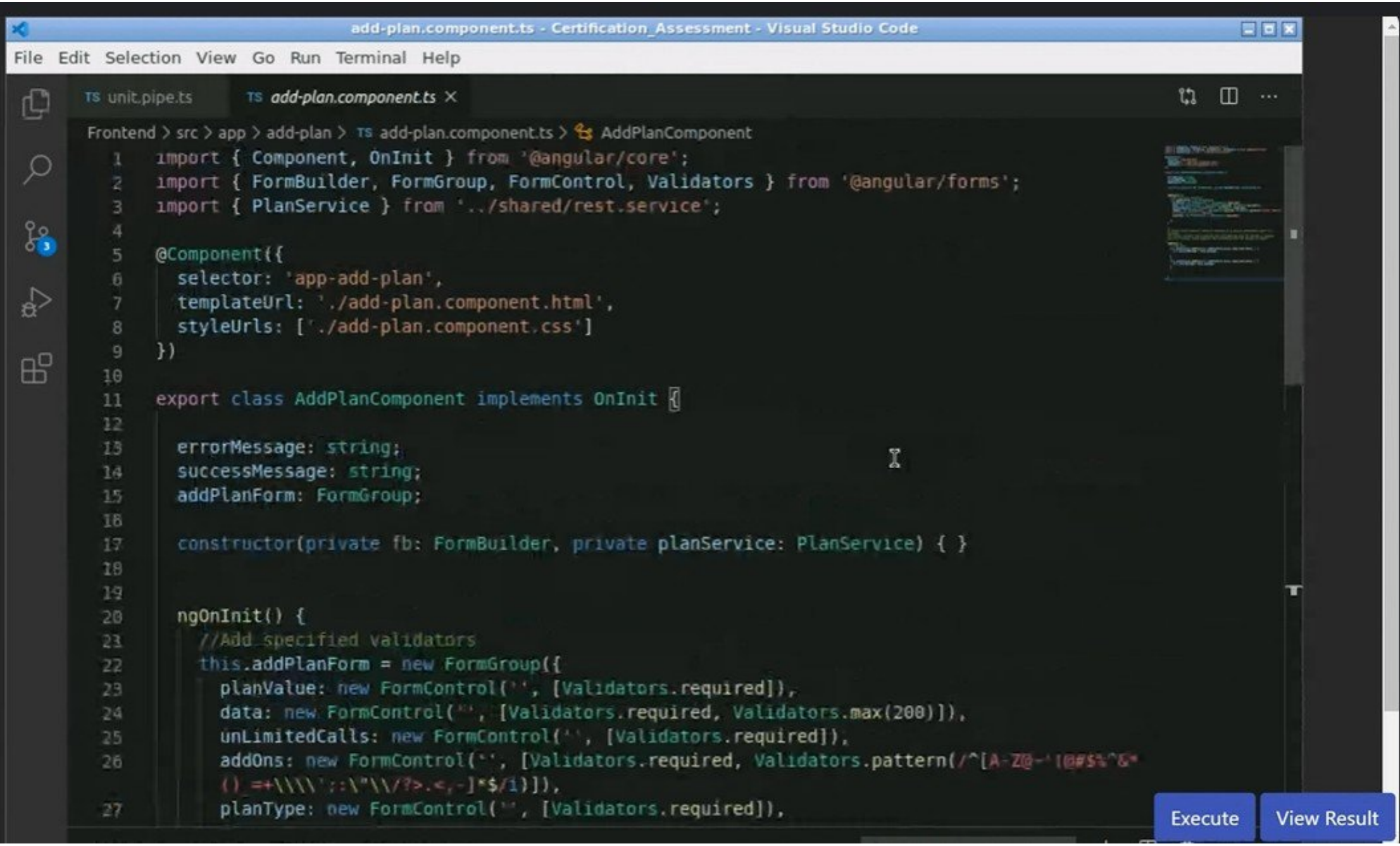

app-routing.module.ts - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
app-routing.module.ts
Frontend > src > app > app-routing.module.ts > routes
import { NgModule } from '@angular/core'
import { Routes, RouterModule } from '@angular/router';
import { AddPlanComponent } from './add-plan/add-plan.component';
import { ViewPlanComponent } from './view-plan/view-plan.component';
/*
Add the route paths to navigate to appropriate
1. /addPlan -> AddPlanComponent
2. /viewPlan -> ViewPlanComponent
3. In case an Invalid URL is provided, the user must be redirected to the addPlan page.
*/
export const routes: Routes = [
{path:'addPlan',component:AddPlanComponent},
{path:'viewPlan', component:ViewPlanComponent},
{path:'**',redirectTo:'addPlan',pathMatch:'full'}
];
@NgModule({
imports: [RouterModule.forRoot(routes)],
exports: [RouterModule]
})
export class AppRoutingModule { }
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
2: ng serve --open --por
xt": formControlName="planId" class="form-control">
Execute
View Result

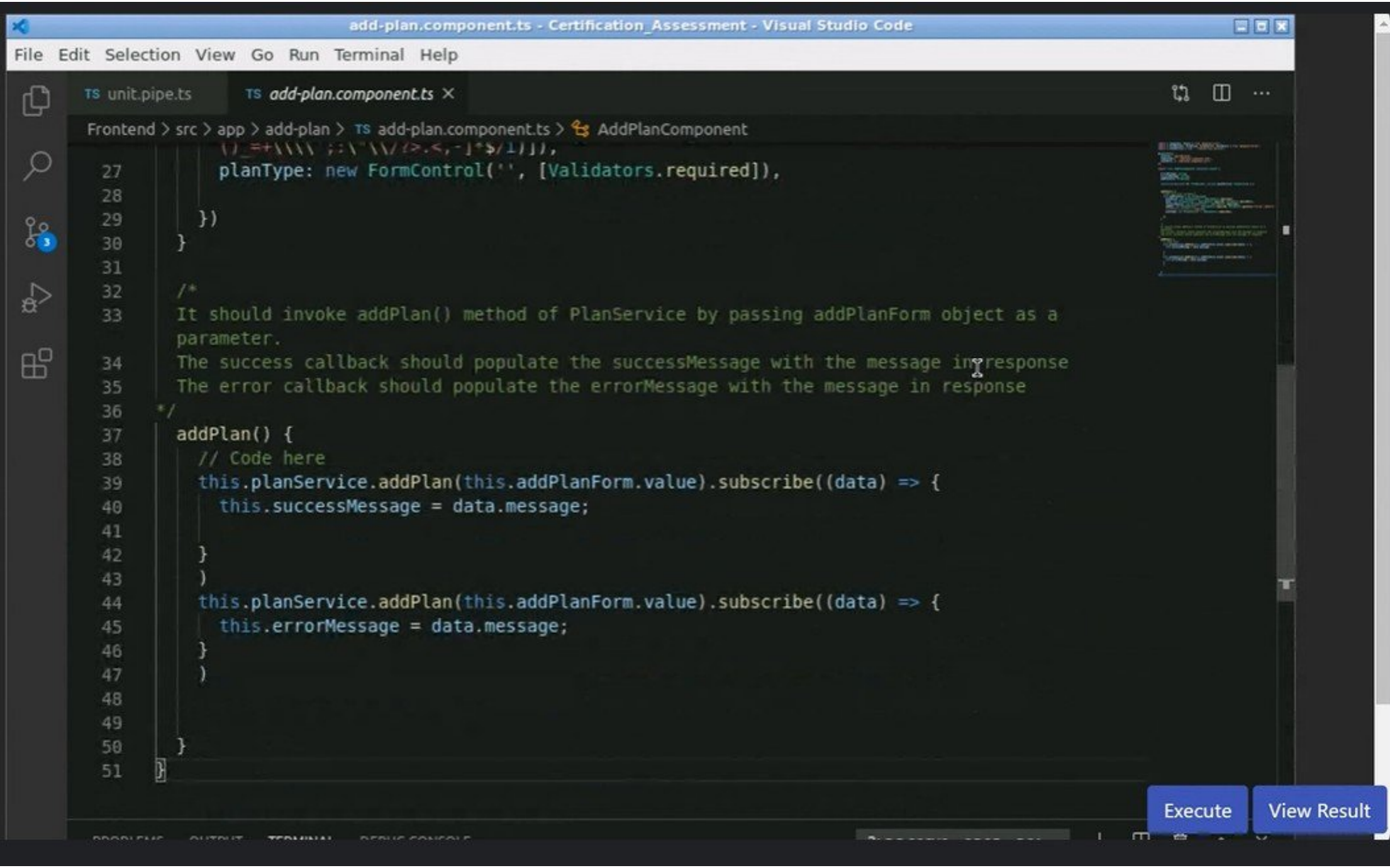

unit.pipe.ts - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
unit.pipe.ts
Frontend > src > app > shared > unit.pipe.ts > UnitPipe > transform
1. Sample Input: 0.2
Expected Output: 200 MB
2. Sample Input: 5
Expected Output: 5 GB
Hint: 1 GB = 1000 MB
*/
@Pipe({
name: 'unitPipe'
})
export class UnitPipe implements PipeTransform {
transform(value: any): any {
// Code here
if(value<1){
return value*1000 + 'MB'
}
else{
return value + 'GB'
}
}
Execute
View Result

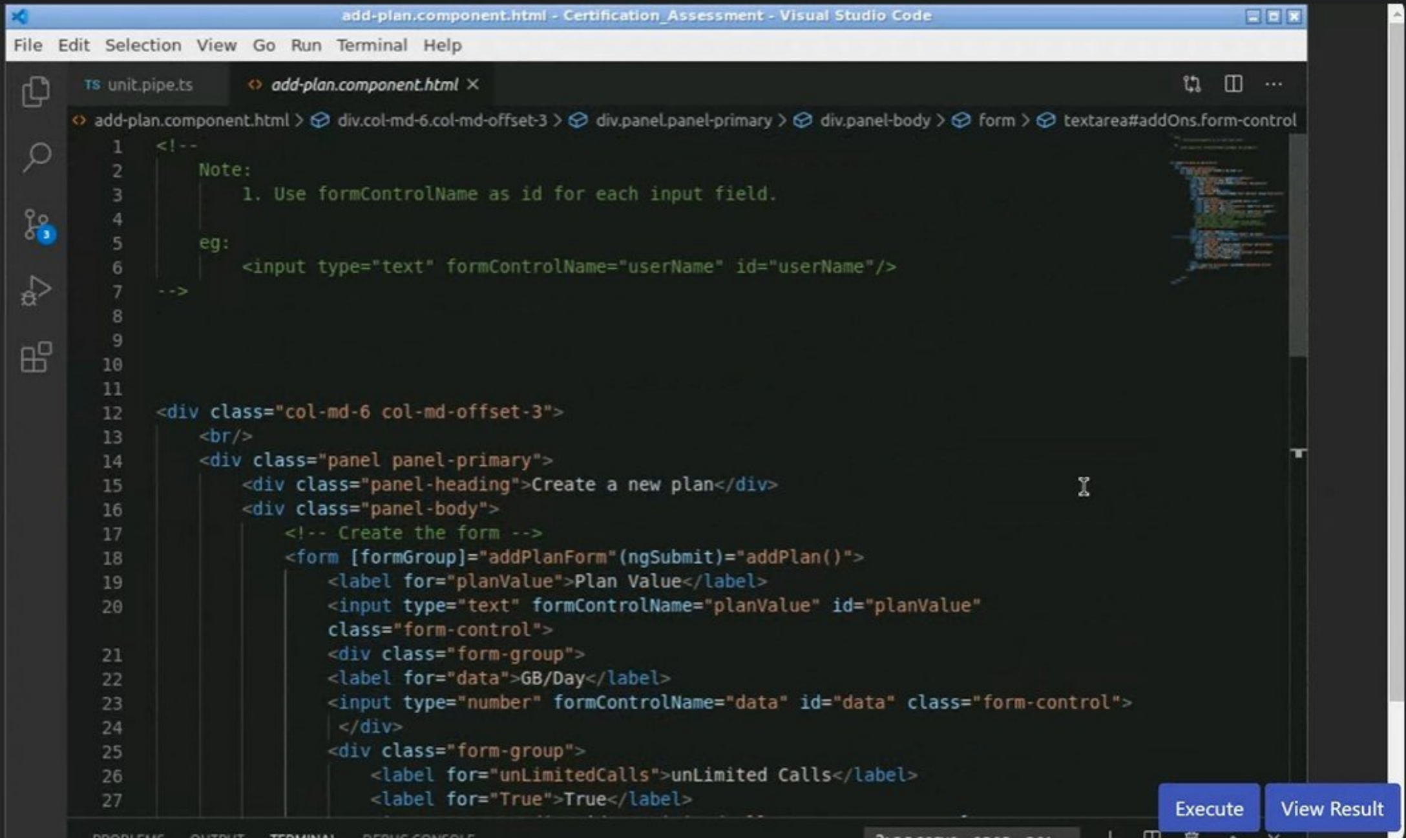

rest.service.ts - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
unit.pipe.ts
rest.service.ts
Frontend > src > app > shared > rest.service.ts > PlanService > deletePlan
import { Injectable } from '@angular/core';
import { HttpClient, HttpParams } from '@angular/common/http';
import { Observable } from 'rxjs';
@Injectable({
providedIn: 'root'
})
export class PlanService {
url = "http://localhost:3020/planDetails" /*Provide the URL of the web service to consume*/
constructor(private httpClient: HttpClient) { }
/*
Consumes the web service exposed at the URL -> http://localhost:3020/planDetails
After sending the request, the response must be an Observable
Return the response back to the AddPlanComponent
*/
addPlan(data: any) :Observable<any> {
// Code here
return this.httpClient.post(this.url,data);
}
/*
Consumes the web service exposed at the URL -> http://localhost:3020/planDetails
After sending the request, the response must be an observable of array"
Execute
View Result


rest.service.ts - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
unit.pipe.ts
rest.service.ts
Frontend > src > app > shared > rest.service.ts > PlanService > deletePlan
Consumes the web service exposed at the URL -> http://localhost:3020/planDetails
After sending the request, the response must be an observable of array"
Return the response back to the ViewPlanComponent
*/
getPlanIds():Observable<any> {
// Code here
return this.httpClient.get(this.url)
}
/*
Consumes the web service exposed at the URL -> http://localhost:3020/planDetails/:planId
After sending the request, the response must be an Observable of type 'Plan'
Return the response back to the PlanDetailsComponent
*/
getPlanDetails(planId): Observable<any> {
// Code here
var params= new HttpParams().set('planId',planId);
return this.httpClient.put(this.url,{params})
}
/*
Consumes the web service exposed at the URL -> http://localhost:3020/planDetails/:planId
After sending the request, the response must be an Observable
Return the response back to the PlanDetailsComponent
Execute
View Result


rest.service.ts - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
unit.pipe.ts
rest.service.ts
Frontend > src > app > shared > rest.service.ts > PlanService > deletePlan
51 /*
52 Consumes the web service exposed at the URL -> http://localhost:3020/planDetails/:planId
53 After sending the request, the response must be an Observable
54 Return the response back to the PlanDetailsComponent
55 */
56
57 deletePlan(planId) {
58 // Code here
59 var params= new HttpParams().set('planId',planId);
60 return this.httpClient.delete(this.url,{params})
61 }
62
63 }
64
Execute
View Result


add-plan.component.ts - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
unit.pipe.ts
add-plan.component.ts
Frontend > src > app > add-plan > add-plan.component.ts > AddPlanComponent
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, FormControl, Validators } from '@angular/forms';
import { PlanService } from '../shared/rest.service';
@Component({
selector: 'app-add-plan',
templateUrl: './add-plan.component.html',
styleUrls: ['./add-plan.component.css']
})
export class AddPlanComponent implements OnInit {
errorMessage: string;
successMessage: string;
addPlanForm: FormGroup;
constructor(private fb: FormBuilder, private planService: PlanService) { }
ngOnInit() {
//Add specified validators
this.addPlanForm = new FormGroup({
planValue: new FormControl('', [Validators.required]),
data: new FormControl('', [Validators.required, Validators.max(200)]),
unLimitedCalls: new FormControl('', [Validators.required]),
addOns: new FormControl('', [Validators.required, Validators.pattern(/^[A-Z@~`!@#$%^&*
() =+\\\\';:\"\\/?>.<,-]*$/i)]),
planType: new FormControl('', [Validators.required]),
Execute
View Result


add-plan.component.ts - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
unit.pipe.ts
add-plan.component.ts
Frontend > src > app > add-plan > add-plan.component.ts > AddPlanComponent
planType: new FormControl('', [Validators.required]),
})
}
/*
It should invoke addPlan() method of PlanService by passing addPlanForm object as a parameter.
The success callback should populate the successMessage with the message in response
The error callback should populate the errorMessage with the message in response
*/
addPlan() {
// Code here
this.planService.addPlan(this.addPlanForm.value).subscribe((data) => {
this.successMessage = data.message;
}
)
this.planService.addPlan(this.addPlanForm.value).subscribe((data) => {
this.errorMessage = data.message;
}
)
}
}
Execute
View Result

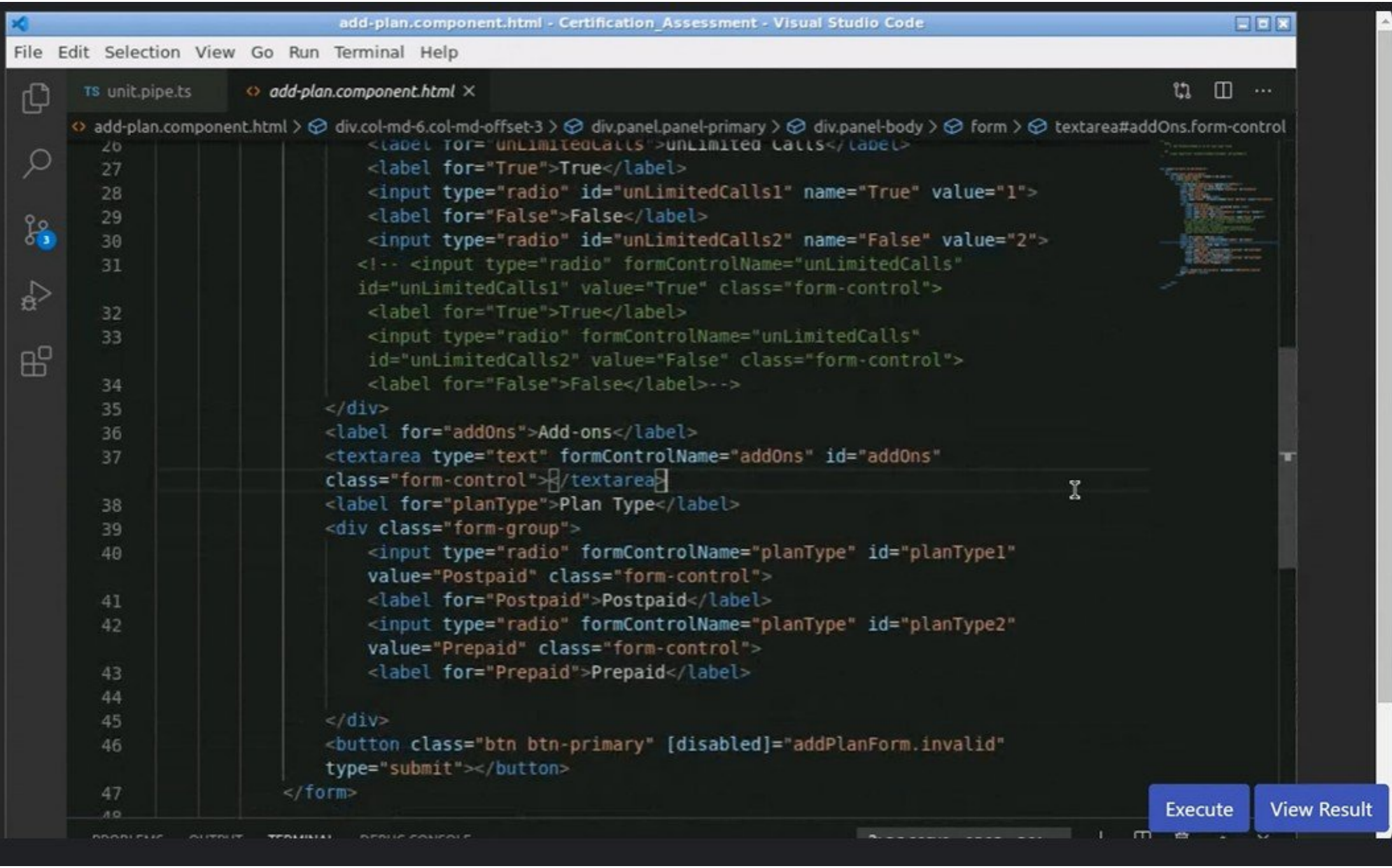

add-plan.component.html - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
unit.pipe.ts
add-plan.component.html
add-plan.component.html > div.col-md-6.col-md-offset-3 > div.panel.panel-primary > div.panel-body > form > textarea#addOns.form-control
<!--
Note:
1. Use formControlName as id for each input field.
eg:
<input type="text" formControlName="userName" id="userName"/>
-->

Create a new plan

<!-- Create the form -->
<form [formGroup]="addPlanForm"(ngSubmit)="addPlan()">
<label for="planValue">Plan Value</label>
<input type="text" formControlName="planValue" id="planValue"
class="form-control">

<label for="data">GB/Day</label>
<input type="number" formControlName="data" id="data" class="form-control">

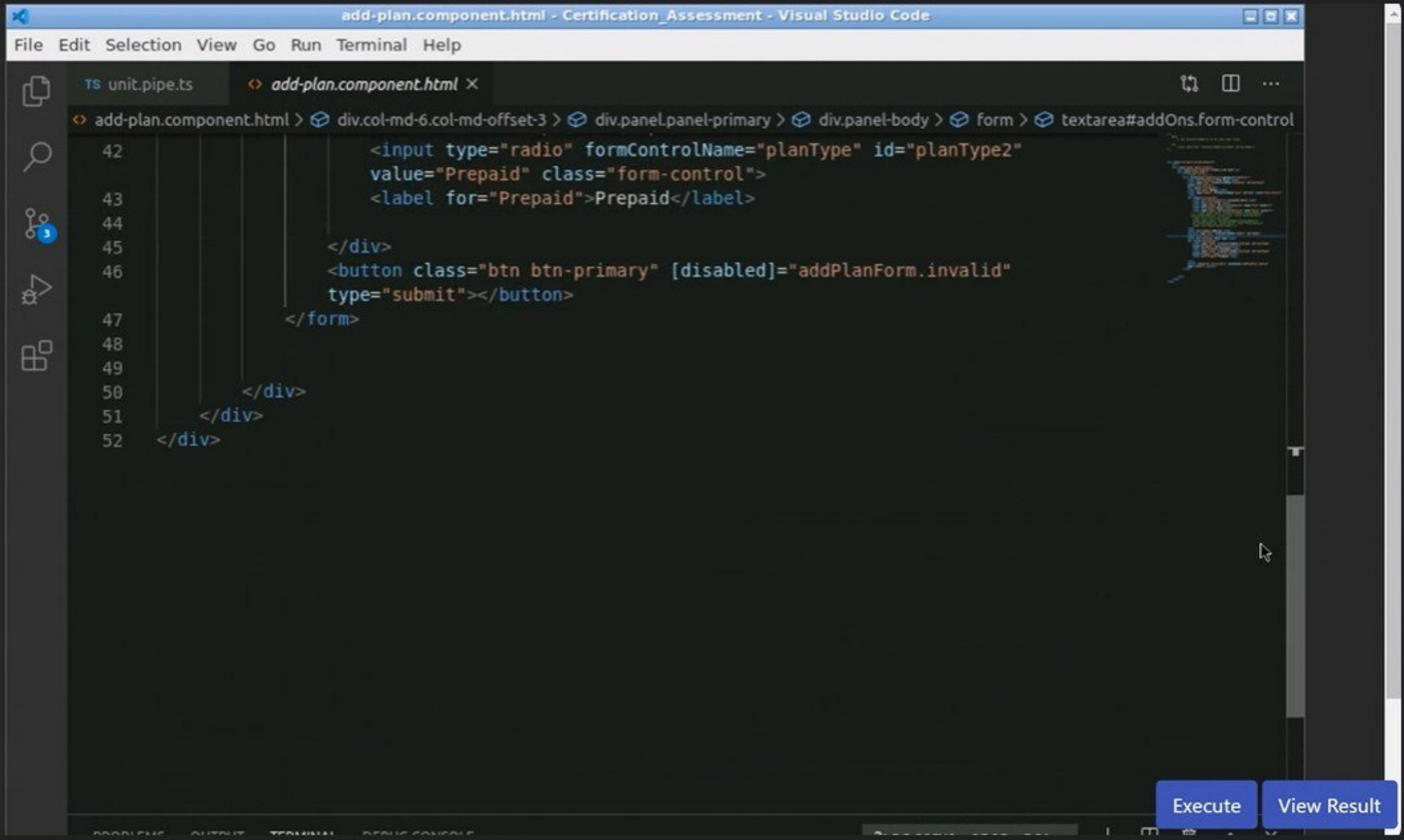
<label for="unLimitedCalls">unLimited Calls</label>
<label for="True">True</label>
Execute
View Result

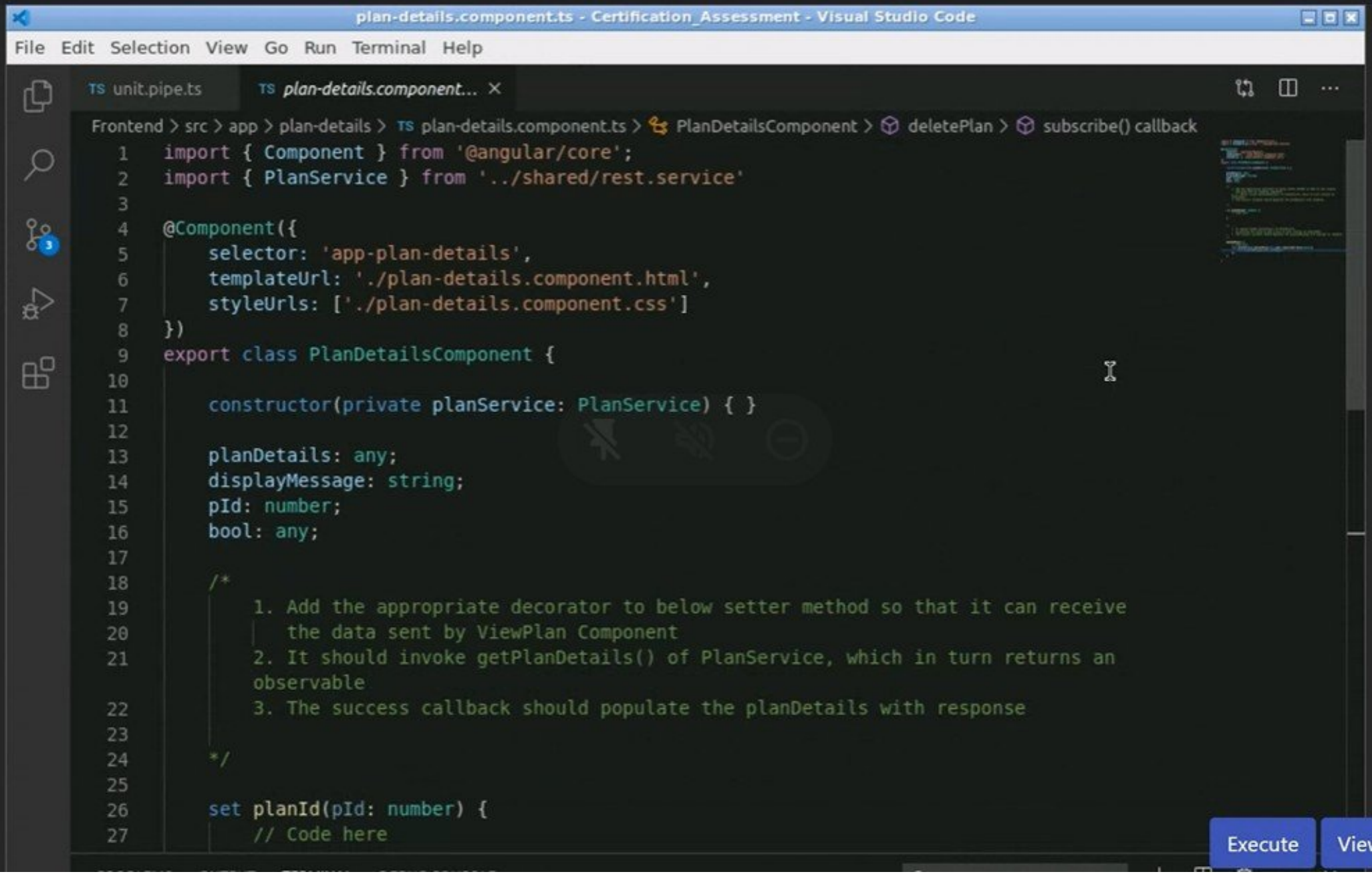

add-plan.component.html - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
unit.pipe.ts
add-plan.component.html
add-plan.component.html > div.col-md-6.col-md-offset-3 > div.panel.panel-primary > div.panel-body > form > textarea#addOns.form-control
<label for="unLimitedCalls">unLimited Calls</label>
<label for="True">True</label>
<input type="radio" id="unLimitedCalls1" name="True" value="1">
<label for="False">False</label>
<input type="radio" id="unLimitedCalls2" name="False" value="2">
<!-- <input type="radio" formControlName="unLimitedCalls"
id="unLimitedCalls1" value="True" class="form-control">
<label for="True">True</label>
<input type="radio" formControlName="unLimitedCalls"
id="unLimitedCalls2" value="False" class="form-control">
<label for="False">False</label>-->

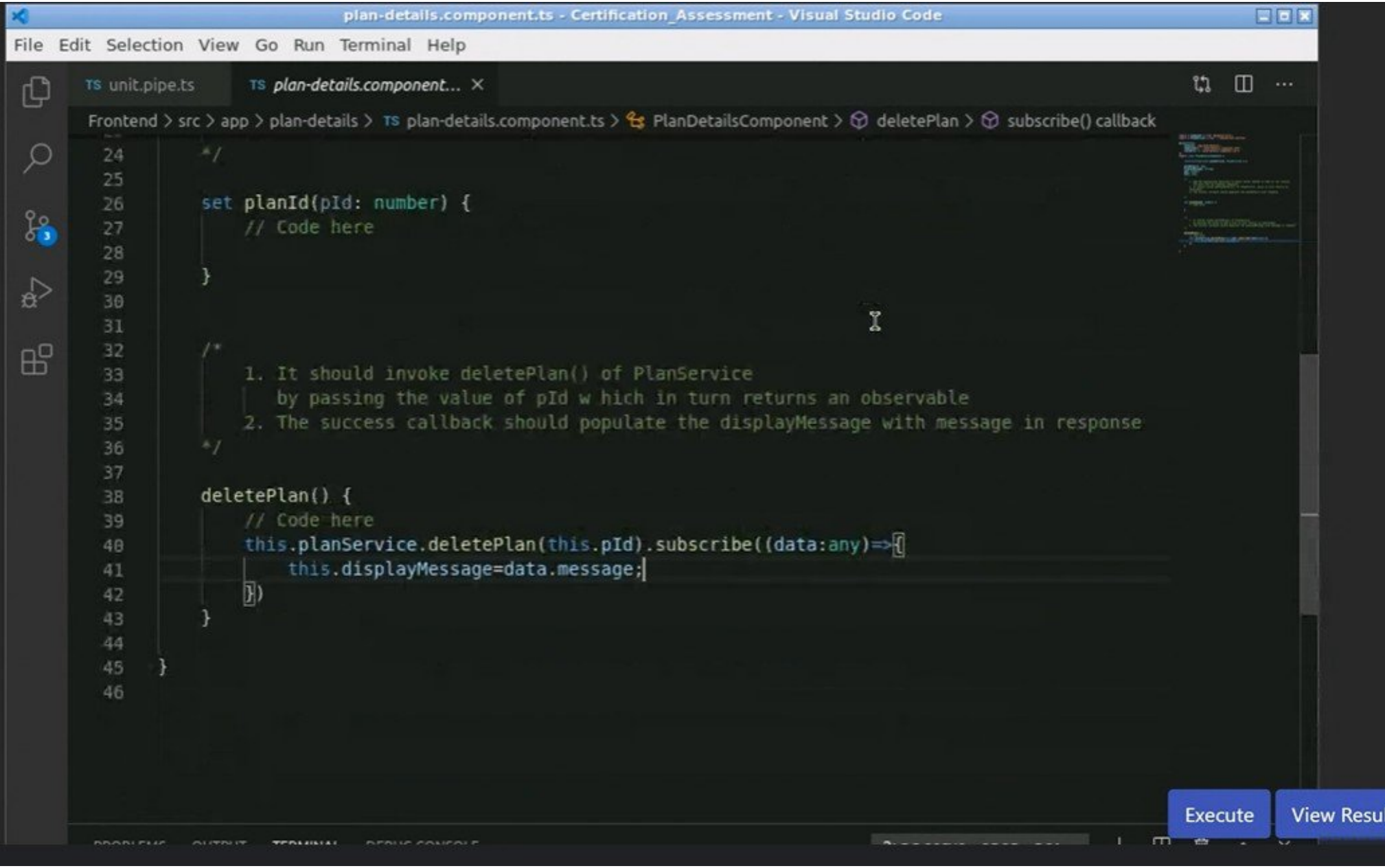
<label for="addOns">Add-ons</label>
<textarea type="text" formControlName="addOns" id="addOns"
class="form-control"></textarea>
<label for="planType">Plan Type</label>

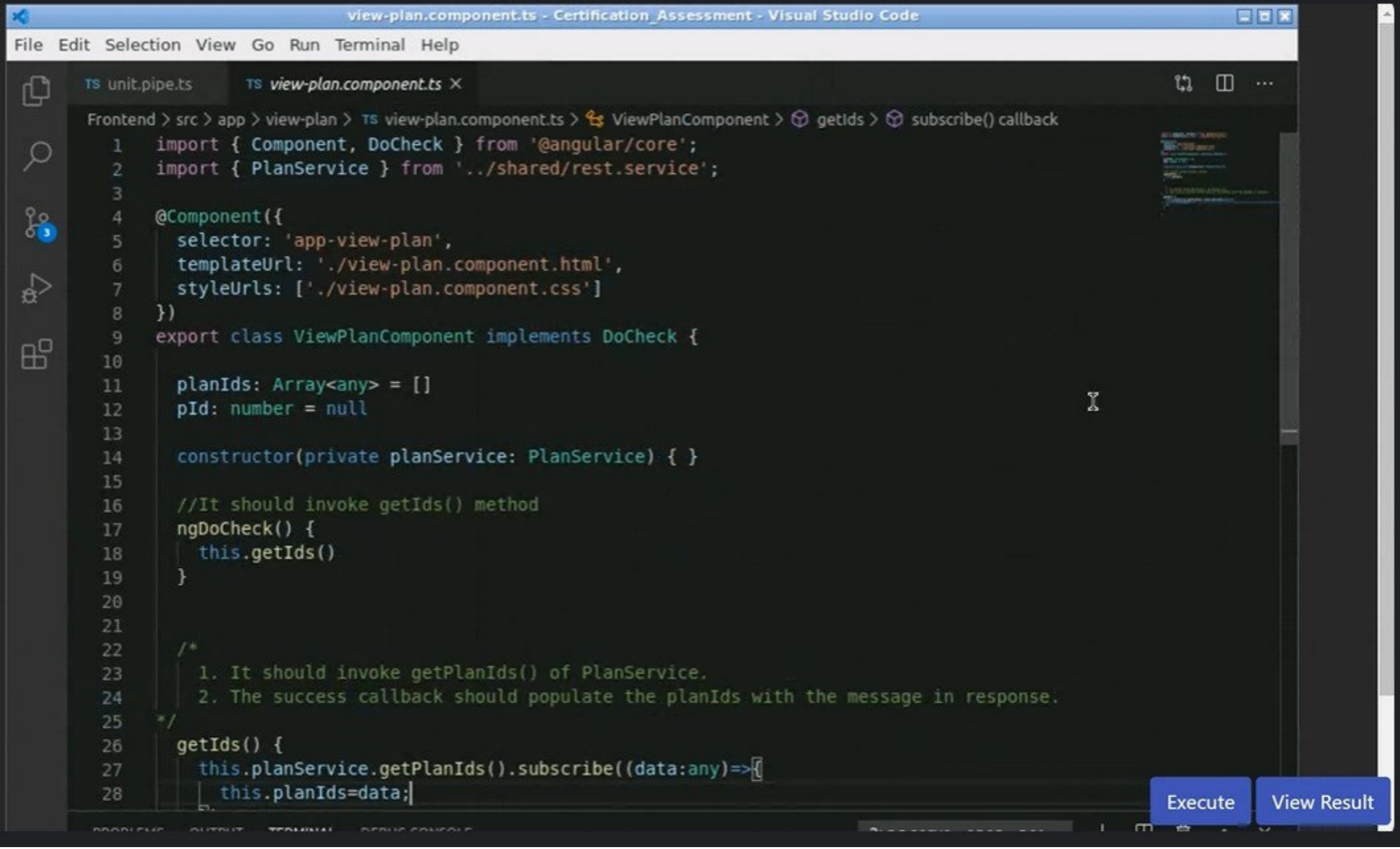
<input type="radio" formControlName="planType" id="planType1"
value="Postpaid" class="form-control">
<label for="Postpaid">Postpaid</label>
<input type="radio" formControlName="planType" id="planType2"
value="Prepaid" class="form-control">
<label for="Prepaid">Prepaid</label>

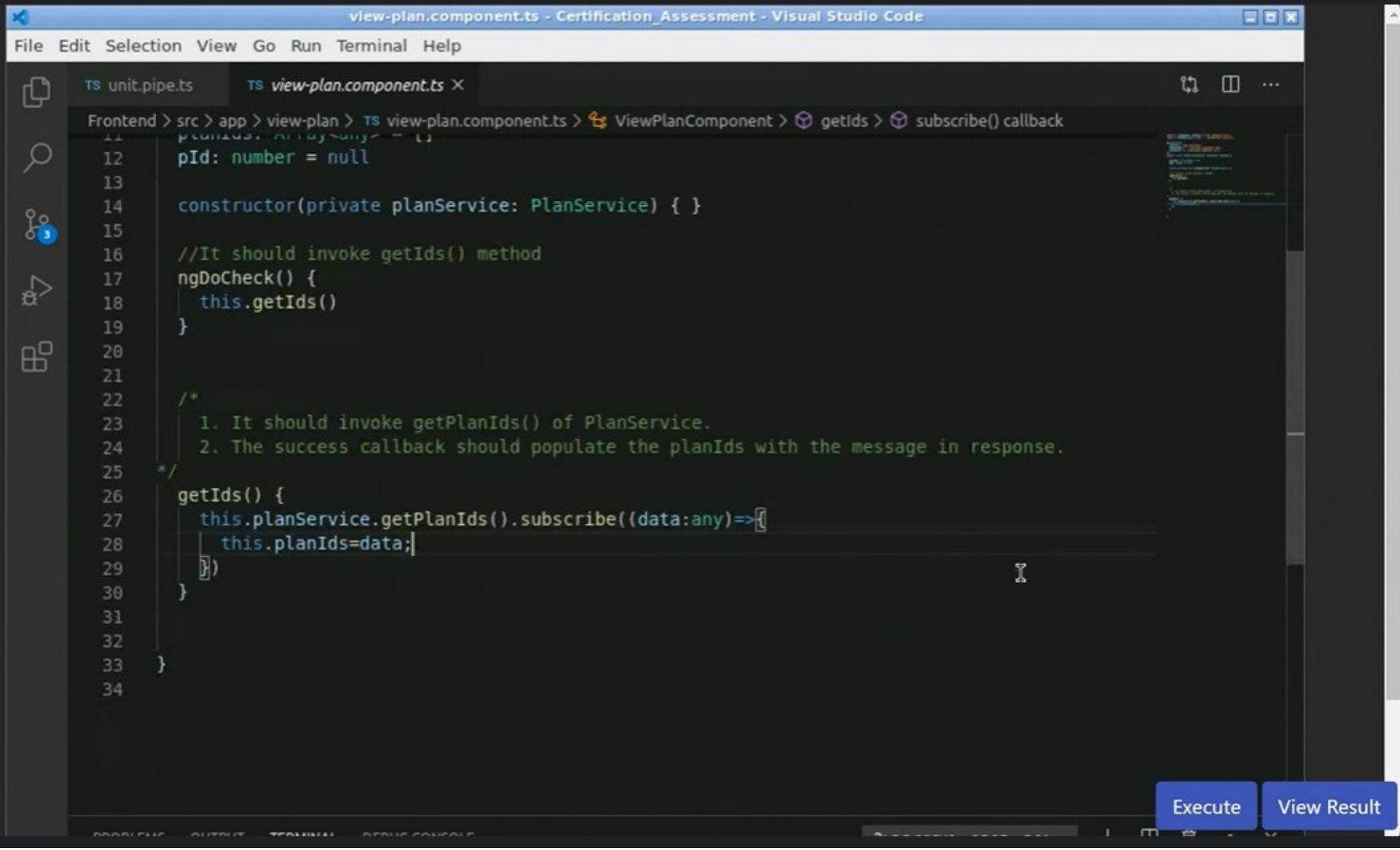
<button class="btn btn-primary" [disabled]="addPlanForm.invalid"
type="submit"></button>
</form>
Execute
View Result

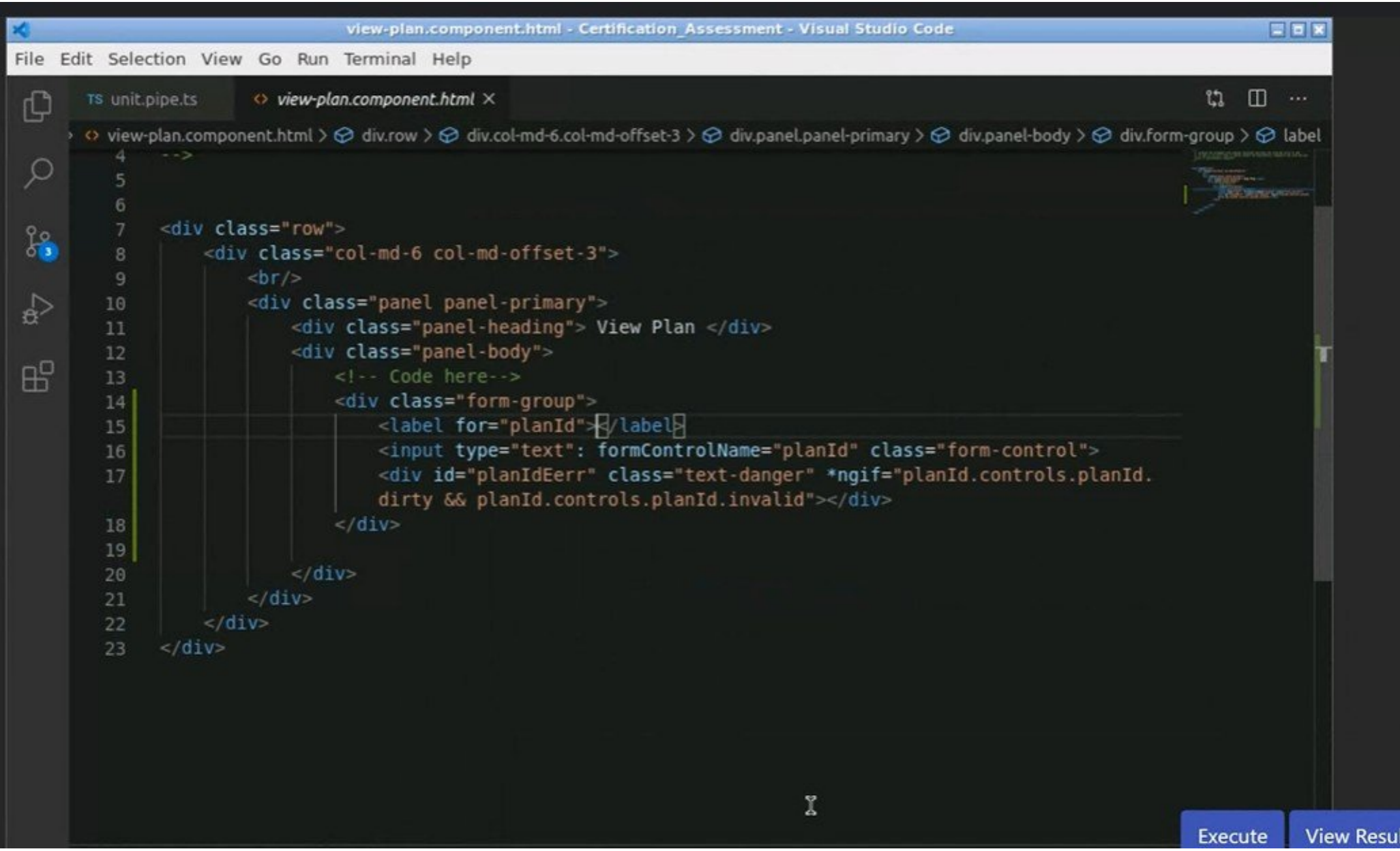

add-plan.component.html - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
unit.pipe.ts
add-plan.component.html
add-plan.component.html > div.col-md-6.col-md-offset-3 > div.panel.panel-primary > div.panel-body > form > textarea#addOns.form-control
42 <input type="radio" formControlName="planType" id="planType2"
value="Prepaid" class="form-control">
43 <label for="Prepaid">Prepaid</label>
44
45
46 <button class="btn btn-primary" [disabled]="addPlanForm.invalid"
type="submit"></button>
47 </form>
48
49
50
51
52
Execute
View Result


plan-details.component.ts - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
unit.pipe.ts
plan-details.component...
Frontend > src > app > plan-details > plan-details.component.ts > PlanDetailsComponent > deletePlan > subscribe() callback
import { Component } from '@angular/core';
import { PlanService } from '../shared/rest.service'
@Component({
selector: 'app-plan-details',
templateUrl: './plan-details.component.html',
styleUrls: ['./plan-details.component.css']
})
export class PlanDetailsComponent {
constructor(private planService: PlanService) { }
planDetails: any;
displayMessage: string;
pId: number;
bool: any;
/*
1. Add the appropriate decorator to below setter method so that it can receive
the data sent by ViewPlan Component
2. It should invoke getPlanDetails() of PlanService, which in turn returns an
observable
3. The success callback should populate the planDetails with response
*/
set planId(pId: number) {
// Code here
Execute
View Result


plan-details.component.ts - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
unit.pipe.ts
plan-details.component...
Frontend > src > app > plan-details > plan-details.component.ts > PlanDetailsComponent > deletePlan > subscribe() callback
*/
set planId(pId: number) {
// Code here
}
/*
1. It should invoke deletePlan() of PlanService
by passing the value of pId w hich in turn returns an observable
2. The success callback should populate the displayMessage with message in response
*/
deletePlan() {
// Code here
this.planService.deletePlan(this.pId).subscribe((data:any)=>{
this.displayMessage=data.message;
})
}
}
Execute
View Result


view-plan.component.ts - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
unit.pipe.ts
view-plan.component.ts
Frontend > src > app > view-plan > view-plan.component.ts > ViewPlanComponent > getIds > subscribe() callback
1 import { Component, DoCheck } from '@angular/core';
2 import { PlanService } from '../shared/rest.service';
3
4 @Component({
5 selector: 'app-view-plan',
6 templateUrl: './view-plan.component.html',
7 styleUrls: ['./view-plan.component.css']
8 })
9 export class ViewPlanComponent implements DoCheck {
10
11 planIds: Array<any> = []
12 pId: number = null
13
14 constructor(private planService: PlanService) { }
15
16 //It should invoke getIds() method
17 ngDoCheck() {
18 this.getIds()
19 }
20
21
22 /*
23 1. It should invoke getPlanIds() of PlanService.
24 2. The success callback should populate the planIds with the message in response.
25 */
26 getIds() {
27 this.planService.getPlanIds().subscribe((data:any)=>{
28 this.planIds=data;
Execute
View Result


view-plan.component.ts - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
unit.pipe.ts
view-plan.component.ts
Frontend > src > app > view-plan > view-plan.component.ts > ViewPlanComponent > getIds > subscribe() callback
pId: number = null
constructor(private planService: PlanService) { }
//It should invoke getIds() method
ngDoCheck() {
this.getIds()
}
/*
1. It should invoke getPlanIds() of PlanService.
2. The success callback should populate the planIds with the message in response.
*/
getIds() {
this.planService.getPlanIds().subscribe((data:any)=>{
this.planIds=data;
})
}
}
Execute
View Result


view-plan.component.html - Certification_Assessment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
unit.pipe.ts
view-plan.component.html
view-plan.component.html > div.row > div.col-md-6.col-md-offset-3 > div.panel.panel-primary > div.panel-body > div.form-group > label
-->

 View Plan

<!-- Code here-->

<label for="planId"></label>
<input type="text": formControlName="planId" class="form-control">
<div id="planIdEerr" class="text-danger" *ngif="planId.controls.planId.
dirty && planId.controls.planId.invalid">

Execute
View Result