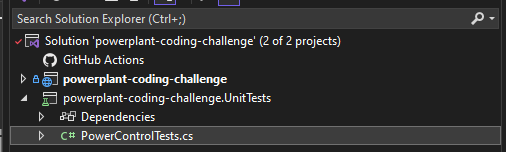
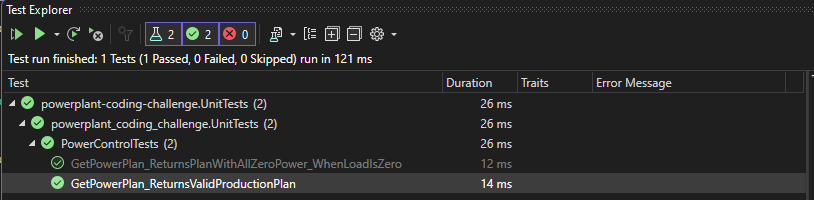
**Unit test (NUnit tests):**



|  |
| --- |
| using powerplant\_coding\_challenge.Domain;  using powerplant\_coding\_challenge.Infrastructure.Models;  using System.Diagnostics.Metrics;  namespace powerplant\_coding\_challenge.UnitTests  {  [TestFixture]  public class PowerControlTests  {  [SetUp]  public void Setup()  {  }  [Test]  public async Task GetPowerPlan\_ReturnsPlanWithAllZeroPower\_WhenLoadIsZero()  {  // Arrange  var productionLoad = new ProductionLoad  {  Load = 0, // Assume zero load  Fuels = new Fuels  {  GasEuroMWh = 13.4f,  KerosineEuroMWh = 50.8f,  CO2EuroTon = 20,  Wind = 60  },  PowerPlants = new List<PowerPlant>  {  new PowerPlant { Name = "gasfiredbig1", Type = "gasfired", Efficiency = 0.53f, PMin = 100, PMax = 460 },  new PowerPlant { Name = "gasfiredbig2", Type = "gasfired", Efficiency = 0.53f, PMin = 100, PMax = 460 },  new PowerPlant { Name = "gasfiredsomewhatsmaller", Type = "gasfired", Efficiency = 0.37f, PMin = 40, PMax = 210 },  new PowerPlant { Name = "tj1", Type = "turbojet", Efficiency = 0.3f, PMin = 0, PMax = 16 },  new PowerPlant { Name = "windpark1", Type = "windturbine", Efficiency = 1, PMin = 0, PMax = 150 },  new PowerPlant { Name = "windpark2", Type = "windturbine", Efficiency = 1, PMin = 0, PMax = 36 }  }  };  var powerControl = new PowerControl();  // Act  var productionPlan = await powerControl.GetPowerPlan(productionLoad);  // Assert  Assert.IsNotNull(productionPlan);  Assert.IsNotNull(productionPlan.Plan);  Assert.That(productionPlan.Plan.Count, Is.EqualTo(productionLoad.PowerPlants.Count));  // Ensure all power values are zero  foreach (var pp in productionPlan.Plan)  {  Assert.That(pp.P, Is.EqualTo(0));  }  }  [Test]  public async Task GetPowerPlan\_ReturnsValidProductionPlan()  {  // Arrange  var productionLoad = new ProductionLoad  {  Load = 480,  Fuels = new Fuels  {  GasEuroMWh = 13.4f,  KerosineEuroMWh = 50.8f,  CO2EuroTon = 20,  Wind = 60  },  PowerPlants = new List<PowerPlant>  {  new PowerPlant { Name = "gasfiredbig1", Type = "gasfired", Efficiency = 0.53f, PMin = 100, PMax = 460 },  new PowerPlant { Name = "gasfiredbig2", Type = "gasfired", Efficiency = 0.53f, PMin = 100, PMax = 460 },  new PowerPlant { Name = "gasfiredsomewhatsmaller", Type = "gasfired", Efficiency = 0.37f, PMin = 40, PMax = 210 },  new PowerPlant { Name = "tj1", Type = "turbojet", Efficiency = 0.3f, PMin = 0, PMax = 16 },  new PowerPlant { Name = "windpark1", Type = "windturbine", Efficiency = 1, PMin = 0, PMax = 150 },  new PowerPlant { Name = "windpark2", Type = "windturbine", Efficiency = 1, PMin = 0, PMax = 36 }  }  };  var powerControl = new PowerControl();  // Act  var productionPlan = await powerControl.GetPowerPlan(productionLoad);  // Assert  Assert.IsNotNull(productionPlan);  Assert.IsNotNull(productionPlan.Plan);  Assert.That(productionPlan.Plan.Count, Is.EqualTo(productionLoad.PowerPlants.Count));  // Ensure sum of production equals load (in this particular case scenario)  var totalProduction = productionPlan.Plan.Sum(pp => pp.P);  Assert.That(totalProduction, Is.EqualTo(productionLoad.Load));  }  }  } |

Results:



|  |  |  |
| --- | --- | --- |
| \ | Test Scenario | Result |
| 1 | GetPowerPlan\_ReturnsPlanWithAllZeroPower\_WhenLoadIsZero | OK |
| 2 | GetPowerPlan\_ReturnsValidProductionPlan | OK |