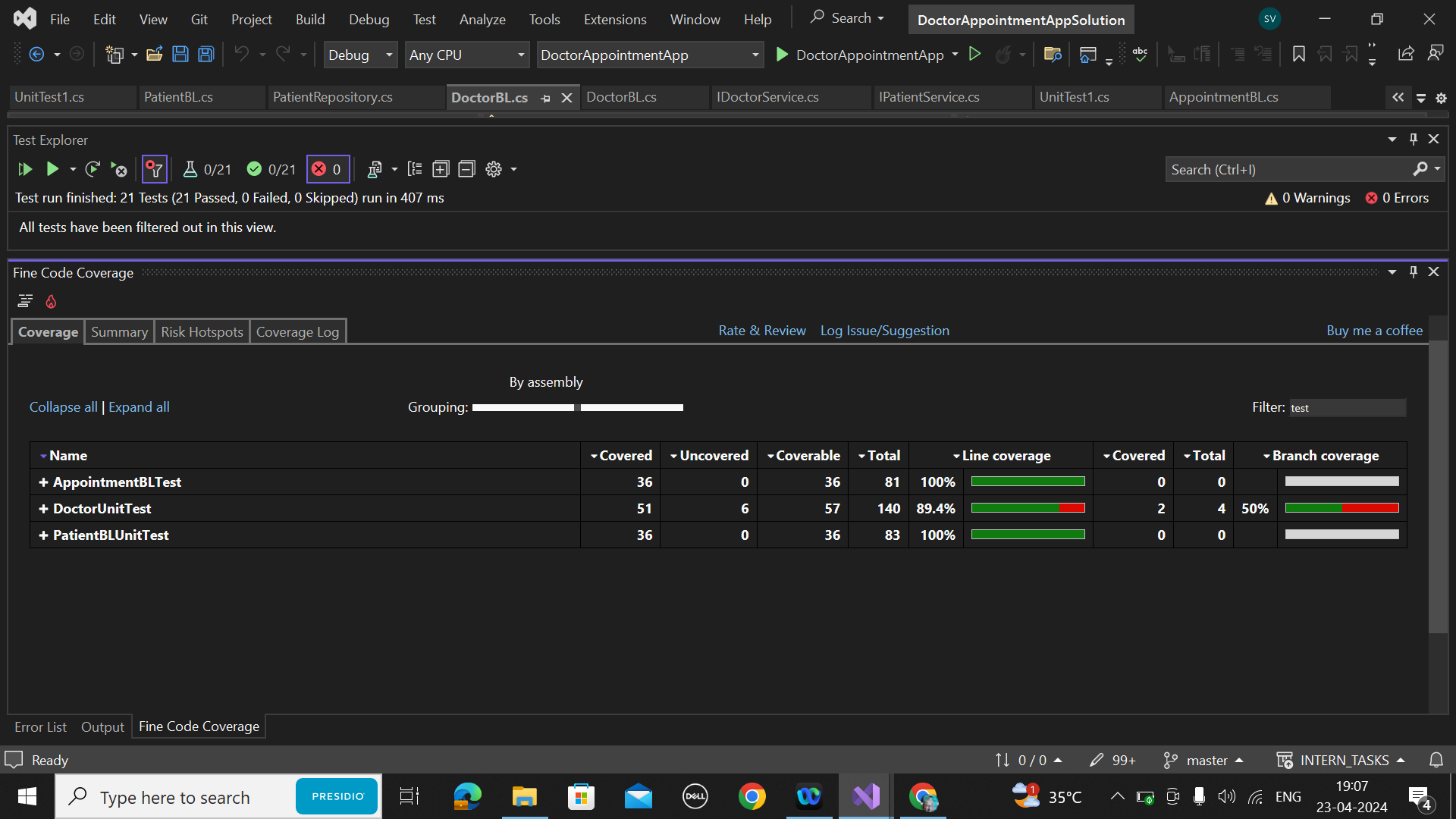
DAY 10  


DOCTORBLTEST.CS

using NUnit.Framework;

using DoctorAppointmentAppModelLibrary;

using DoctorAppointmentAppBLLibrary;

using DoctorAppointmentAppDLLibrary;

using System.Collections.Generic;

namespace DoctorAppointmentAppTests

{

[TestFixture]

public class PatientBLTests

{

PatientBL \_patientBL;

PatientRepository \_patientRepo;

[SetUp]

public void Setup()

{

\_patientRepo = new PatientRepository();

\_patientBL = new PatientBL(\_patientRepo);

}

[Test]

public void AddPatient\_ShouldReturnPatientId\_WhenPatientIsAdded()

{

var patient = new Patient { PatientId = 1, Name = "Test Patient" };

var result = \_patientBL.AddPatient(patient);

Assert.AreEqual(patient.PatientId, result);

}

[Test]

public void GetPatientById\_ShouldReturnPatient\_WhenPatientExists()

{

var patient = new Patient { PatientId = 1, Name = "Test Patient" };

\_patientBL.AddPatient(patient);

var result = \_patientBL.GetPatientById(1);

Assert.AreEqual(patient, result);

}

[Test]

public void UpdatePatient\_ShouldReturnUpdatedPatient\_WhenPatientExists()

{

var patient = new Patient { PatientId = 1, Name = "Test Patient" };

\_patientBL.AddPatient(patient);

patient.Name = "Updated Patient";

var result = \_patientBL.UpdatePatient(patient);

Assert.AreEqual(patient, result);

}

[Test]

public void AddPatient\_ShouldReturnZero\_WhenPatientIsNotAdded()

{

var patient = new Patient { PatientId = 0, Name = "Test Patient" };

var result = \_patientBL.AddPatient(patient);

Assert.AreEqual(0, result);

}

[Test]

public void GetPatientById\_ShouldReturnNull\_WhenPatientDoesNotExist()

{

var result = \_patientBL.GetPatientById(999);

Assert.IsNull(result);

}

[Test]

public void UpdatePatient\_ShouldReturnNull\_WhenPatientDoesNotExist()

{

var patient = new Patient { PatientId = 999, Name = "Nonexistent Patient" };

var result = \_patientBL.UpdatePatient(patient);

Assert.IsNull(result);

}

}

}

PATIENTBL.CS  
using NUnit.Framework;

using DoctorAppointmentAppModelLibrary;

using DoctorAppointmentAppBLLibrary;

using DoctorAppointmentAppDLLibrary;

using System.Collections.Generic;

namespace DoctorAppointmentAppTests

{

[TestFixture]

public class PatientBLTests

{

PatientBL \_patientBL;

PatientRepository \_patientRepo;

[SetUp]

public void Setup()

{

\_patientRepo = new PatientRepository();

\_patientBL = new PatientBL(\_patientRepo);

}

[Test]

public void AddPatient\_ShouldReturnPatientId\_WhenPatientIsAdded()

{

var patient = new Patient { PatientId = 1, Name = "Test Patient" };

var result = \_patientBL.AddPatient(patient);

Assert.AreEqual(patient.PatientId, result);

}

[Test]

public void GetPatientById\_ShouldReturnPatient\_WhenPatientExists()

{

var patient = new Patient { PatientId = 1, Name = "Test Patient" };

\_patientBL.AddPatient(patient);

var result = \_patientBL.GetPatientById(1);

Assert.AreEqual(patient, result);

}

[Test]

public void UpdatePatient\_ShouldReturnUpdatedPatient\_WhenPatientExists()

{

var patient = new Patient { PatientId = 1, Name = "Test Patient" };

\_patientBL.AddPatient(patient);

patient.Name = "Updated Patient";

var result = \_patientBL.UpdatePatient(patient);

Assert.AreEqual(patient, result);

}

[Test]

public void AddPatient\_ShouldReturnZero\_WhenPatientIsNotAdded()

{

var patient = new Patient { PatientId = 0, Name = "Test Patient" };

var result = \_patientBL.AddPatient(patient);

Assert.AreEqual(0, result);

}

[Test]

public void GetPatientById\_ShouldReturnNull\_WhenPatientDoesNotExist()

{

var result = \_patientBL.GetPatientById(999);

Assert.IsNull(result);

}

[Test]

public void UpdatePatient\_ShouldReturnNull\_WhenPatientDoesNotExist()

{

var patient = new Patient { PatientId = 999, Name = "Nonexistent Patient" };

var result = \_patientBL.UpdatePatient(patient);

Assert.IsNull(result);

}

}

}

APPOINTMENTBL.CS  
using DoctorAppointmentAppBLLibrary;

using DoctorAppointmentAppDLLibrary;

using DoctorAppointmentAppModelLibrary;

namespace DoctorAppointmentAppTests

{

[TestFixture]

public class AppointmentBLTests

{

AppointmentBL \_appointmentBL;

AppointmentRepository \_appointmentRepo;

[SetUp]

public void Setup()

{

\_appointmentRepo = new AppointmentRepository();

\_appointmentBL = new AppointmentBL(\_appointmentRepo);

}

[Test]

public void ScheduleAppointment\_ShouldReturnAppointmentId\_WhenAppointmentIsScheduled()

{

var appointment = new Appointment { AppointmentId = 1, AppointmentDate = DateTime.Now, Availability = true };

var result = \_appointmentBL.ScheduleAppointment(appointment);

Assert.AreEqual(appointment.AppointmentId, result);

}

[Test]

public void CancelAppointment\_ShouldReturnCancelledAppointment\_WhenAppointmentExists()

{

var appointment = new Appointment { AppointmentId = 1, AppointmentDate = DateTime.Now, Availability = true };

\_appointmentBL.ScheduleAppointment(appointment);

var result = \_appointmentBL.CancelAppointment(1);

Assert.AreEqual(appointment, result);

}

[Test]

public void RescheduleAppointment\_ShouldReturnUpdatedAppointment\_WhenAppointmentExists()

{

var appointment = new Appointment { AppointmentId = 1, AppointmentDate = DateTime.Now, Availability = true };

\_appointmentBL.ScheduleAppointment(appointment);

var result = \_appointmentBL.RescheduleAppointment(1);

Assert.AreEqual(appointment, result);

}

[Test]

public void ScheduleAppointment\_ShouldReturnZero\_WhenAppointmentIsNotScheduled()

{

var appointment = new Appointment { AppointmentId = 0, AppointmentDate = DateTime.Now, Availability = true };

var result = \_appointmentBL.ScheduleAppointment(appointment);

Assert.AreEqual(0, result);

}

[Test]

public void ScheduleAppointment\_ShouldThrowException\_WhenAppointmentIsNull()

{

Appointment appointment = null;

Assert.Throws<ArgumentNullException>(() => \_appointmentBL.ScheduleAppointment(appointment));

}

[Test]

public void CancelAppointment\_ShouldThrowException\_WhenAppointmentIdIsInvalid()

{

Assert.Throws<ArgumentException>(() => \_appointmentBL.CancelAppointment(-1));

}

[Test]

public void RescheduleAppointment\_ShouldThrowException\_WhenAppointmentIdIsInvalid()

{

Assert.Throws<ArgumentException>(() => \_appointmentBL.RescheduleAppointment(-1));

}

}

}