**Case Study: Virtual Art Gallery**

Name : RISHATHA SHAHIN B

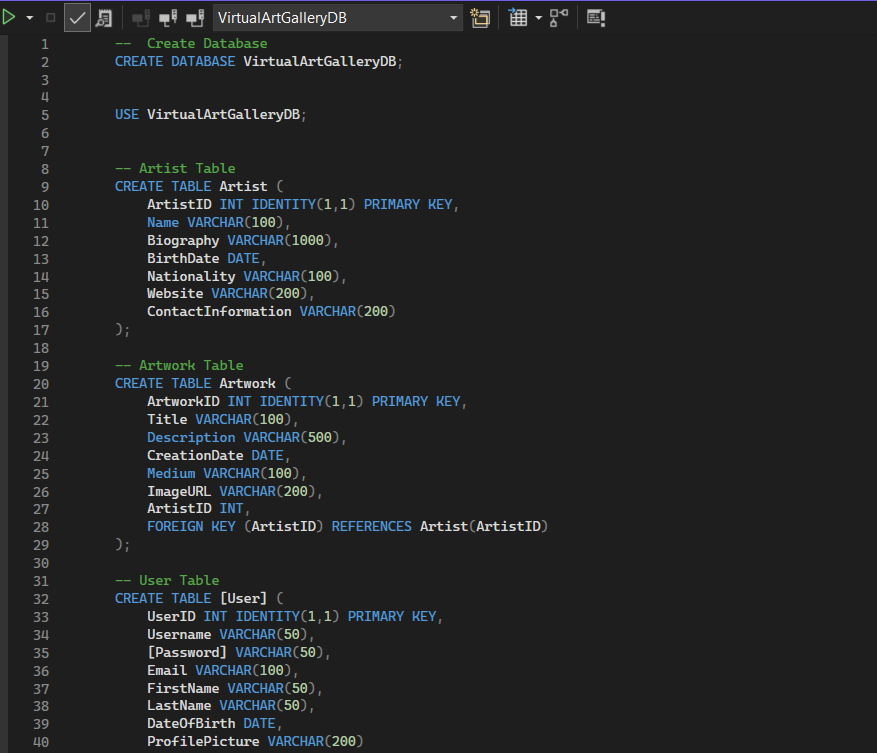
Superset ID : 5294964

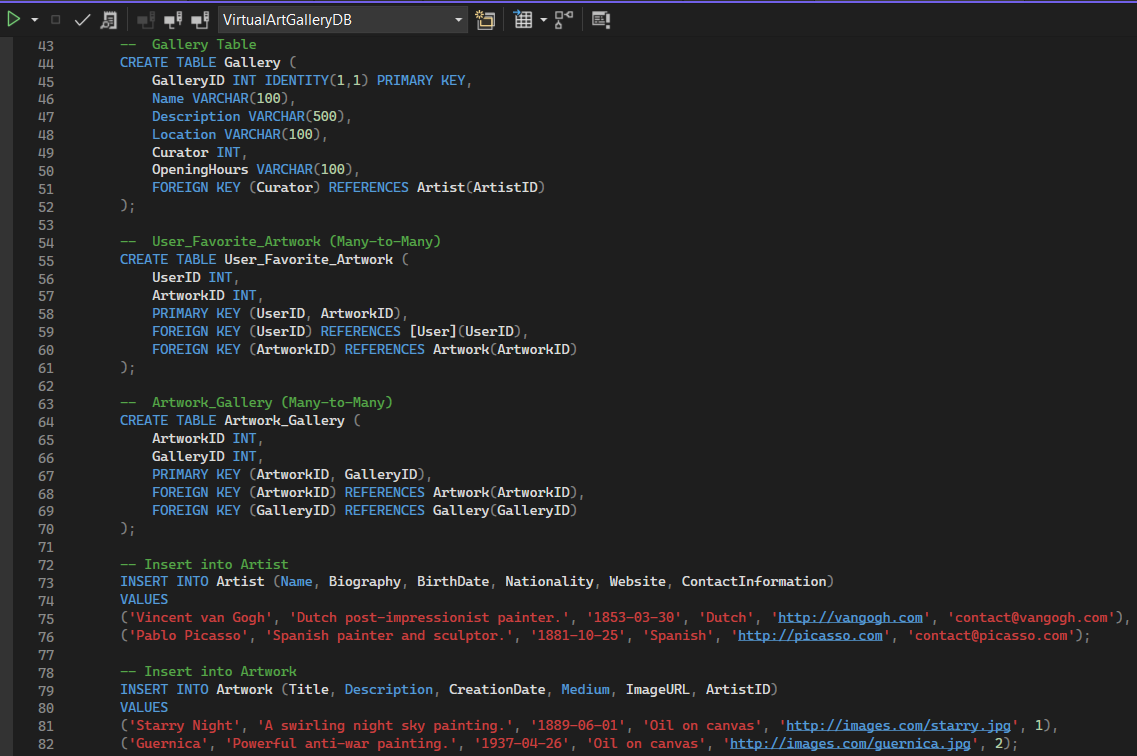
Email : rishatha2004@gmail.com

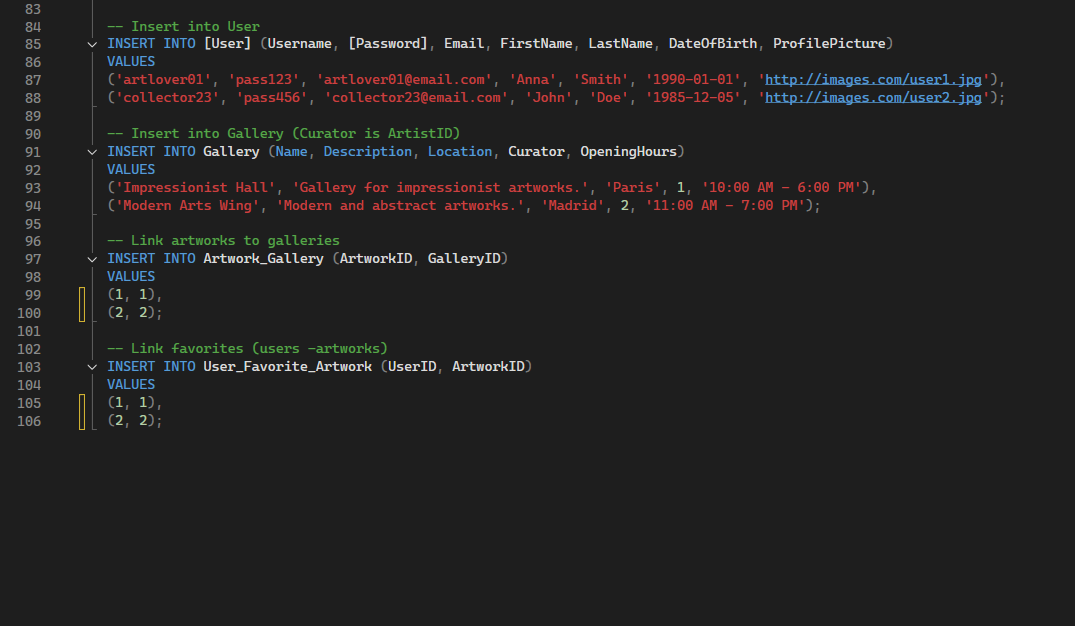
Batch : C# Batch 4

GitHub: <https://github.com/rishatha/Case-Study.git>

**Schema design:**



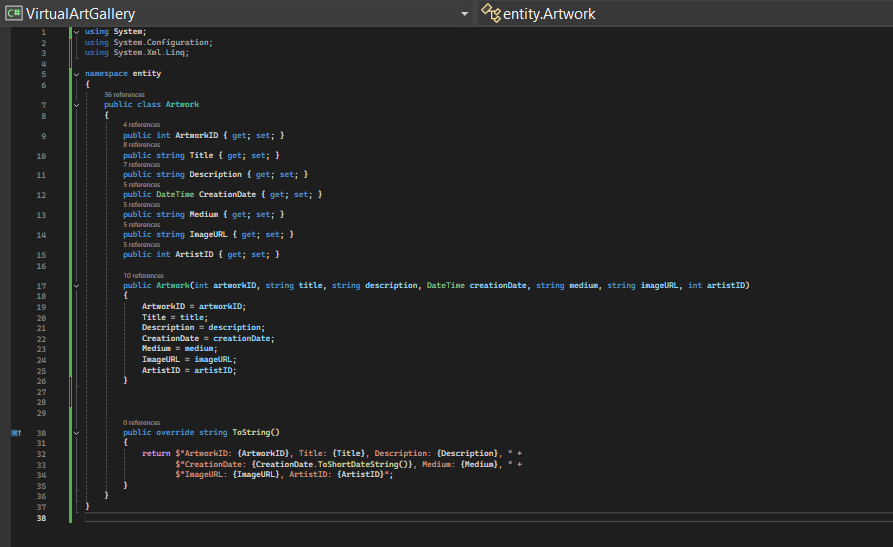




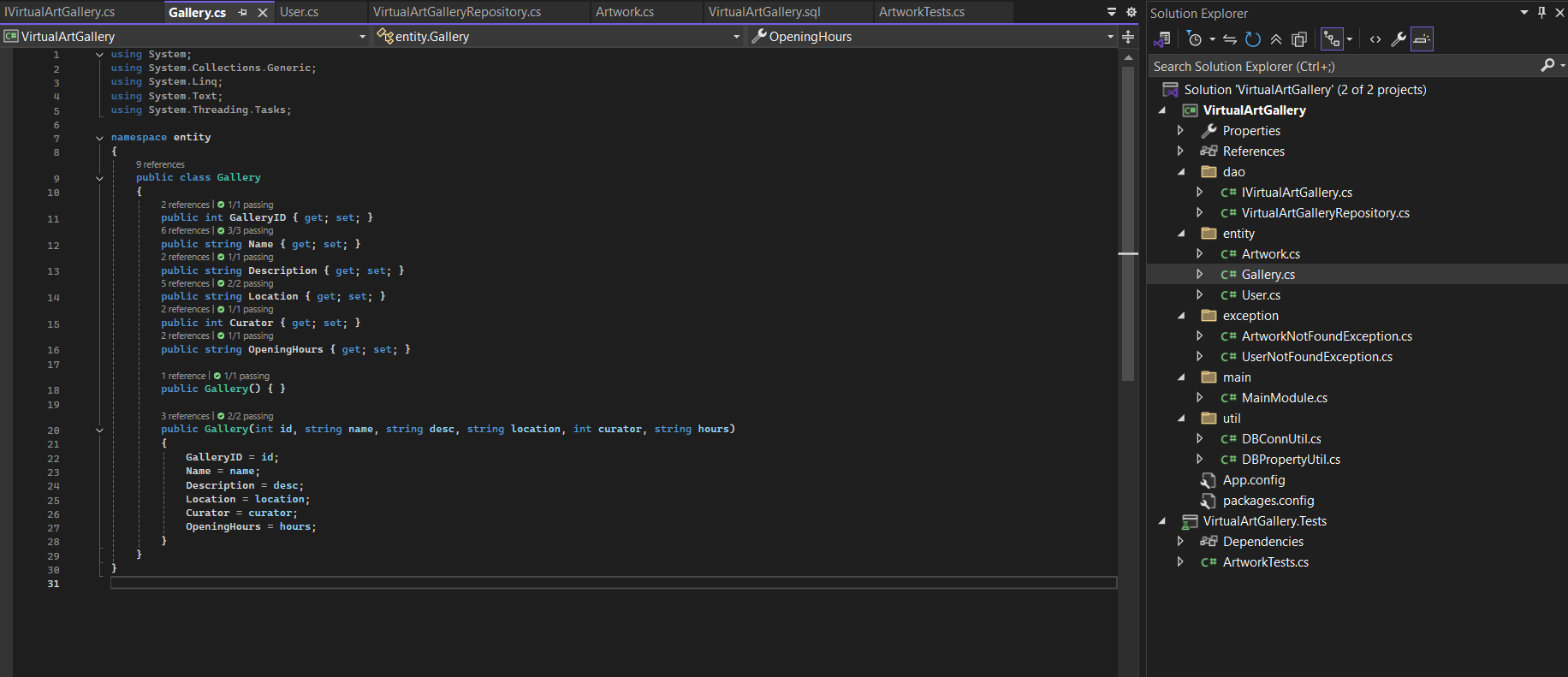
**Coding**

Create the model/entity classes corresponding to the schema within package entity with variables declared private, constructors(default and parametrized) and getters,setters )

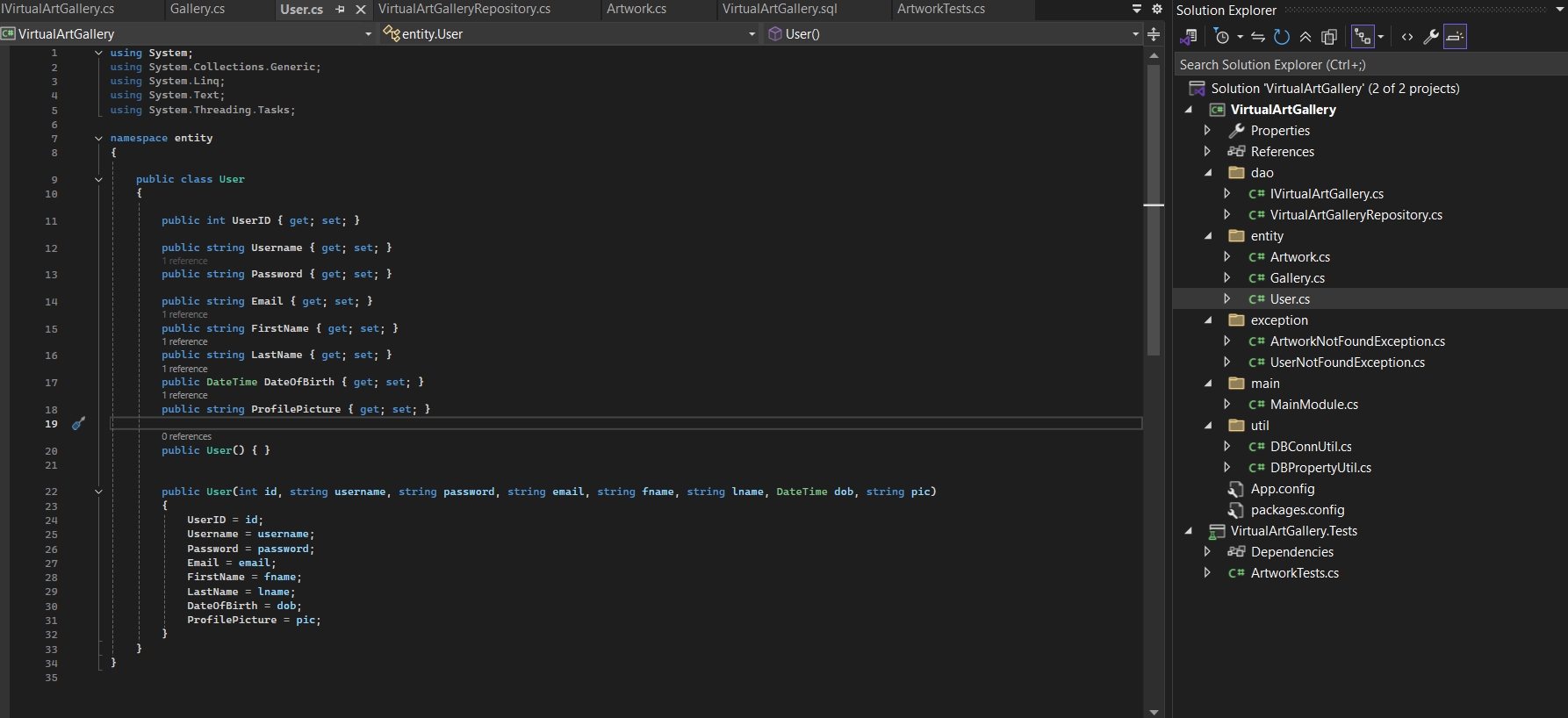
**Entity-Artwork.cs**



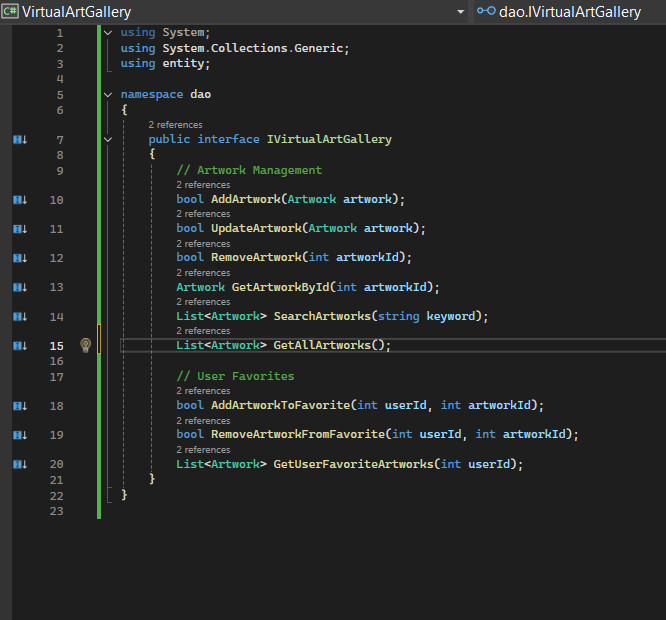
**Entity-Gallery.cs**



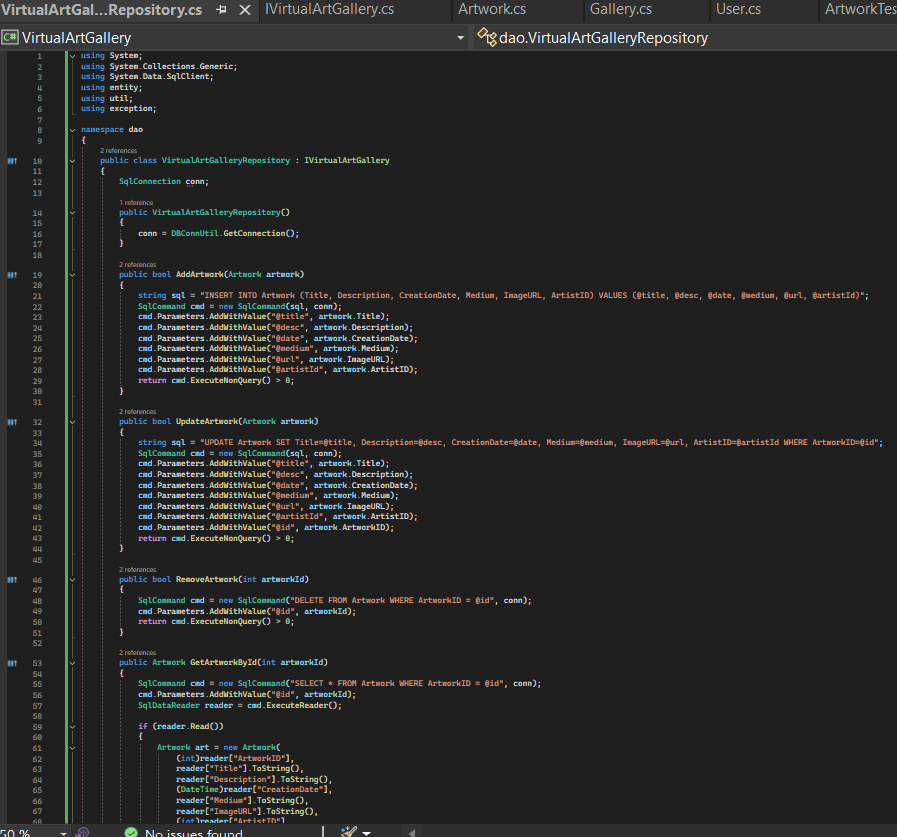
**Entity-Users.cs**

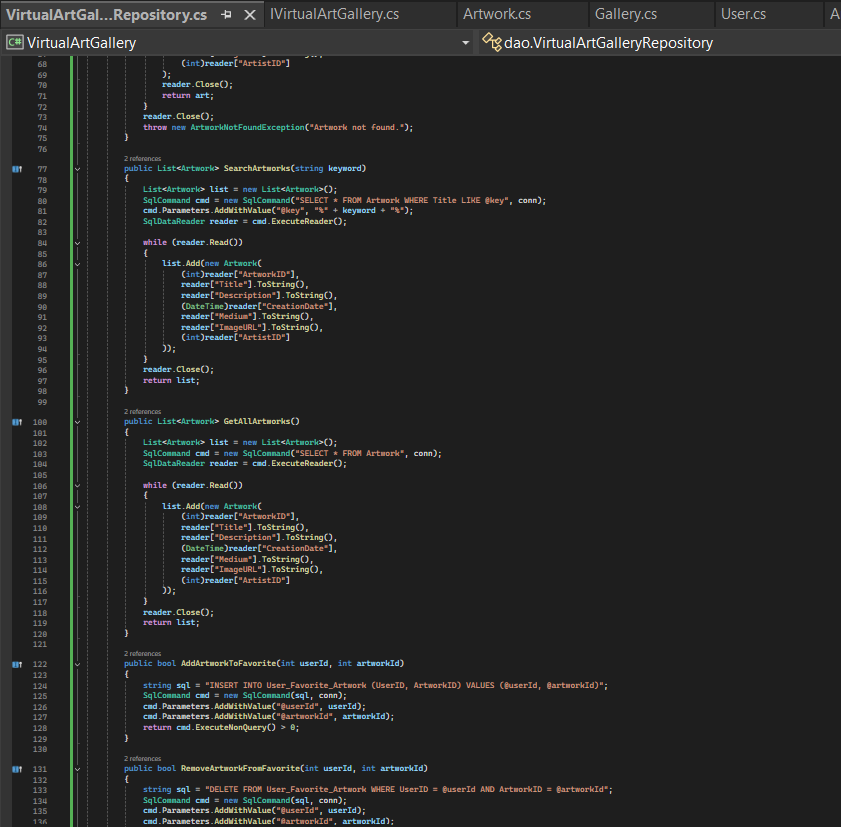


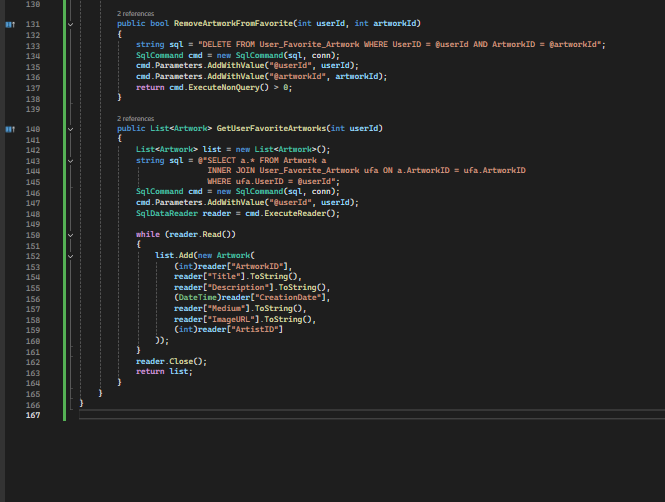
**Dao- IVirtualArtGallery**



**Dao- VirtualArtGalleryRepository**



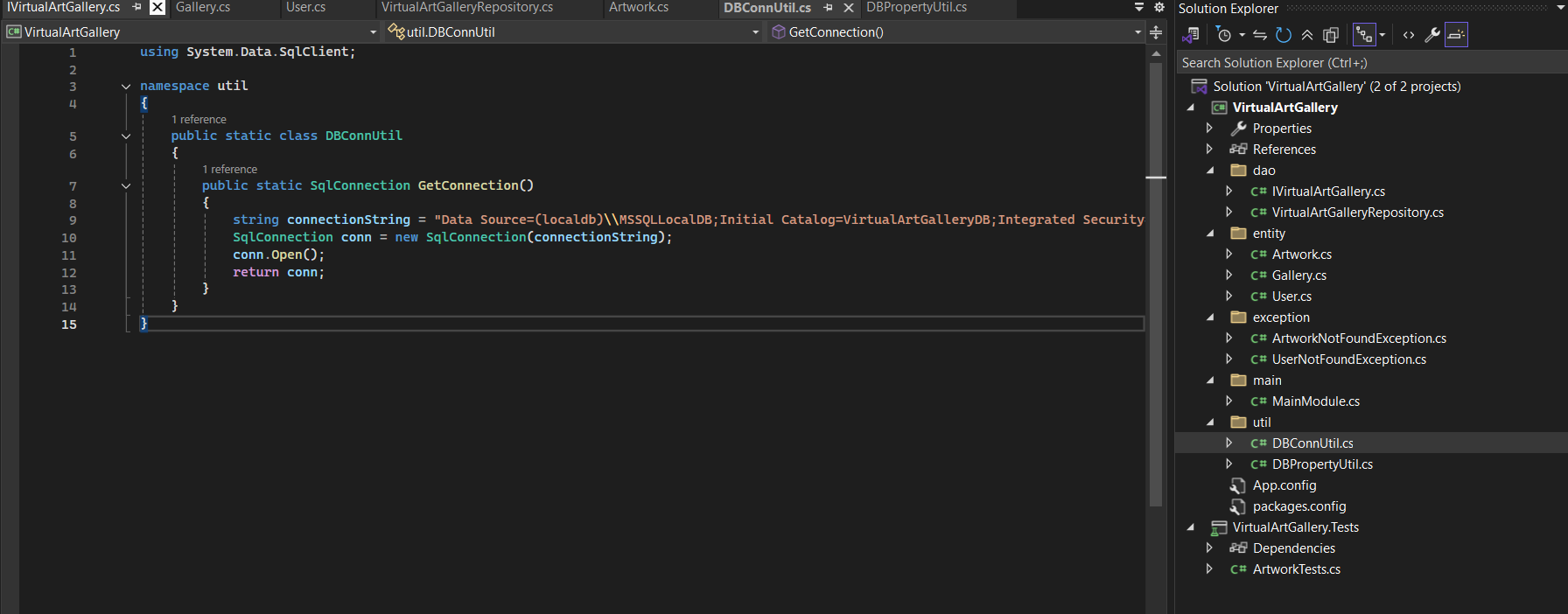




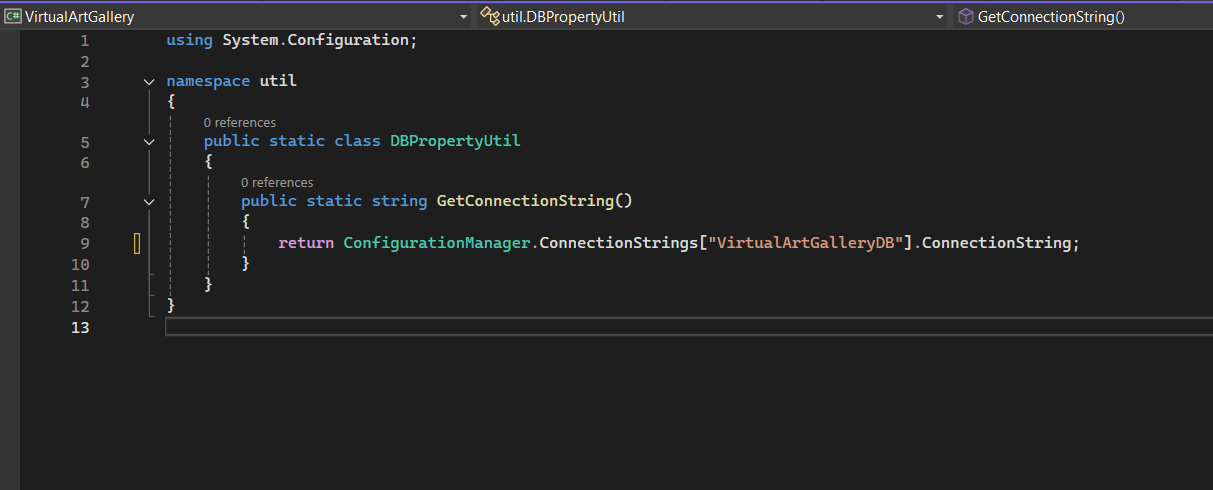
**DBConnection**

Connect your application to the SQL database

**Util- DBConnUtil**



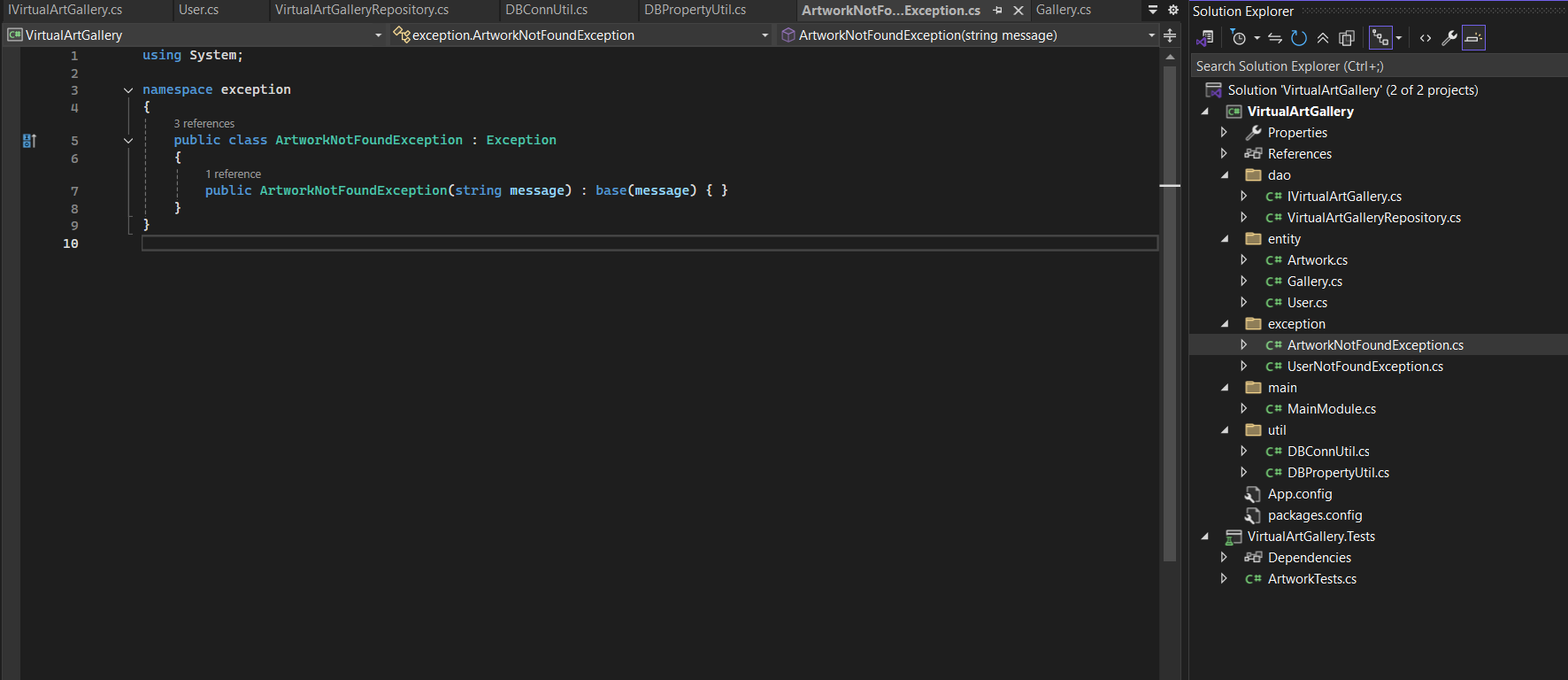
**Util- DBPropertyUtil**



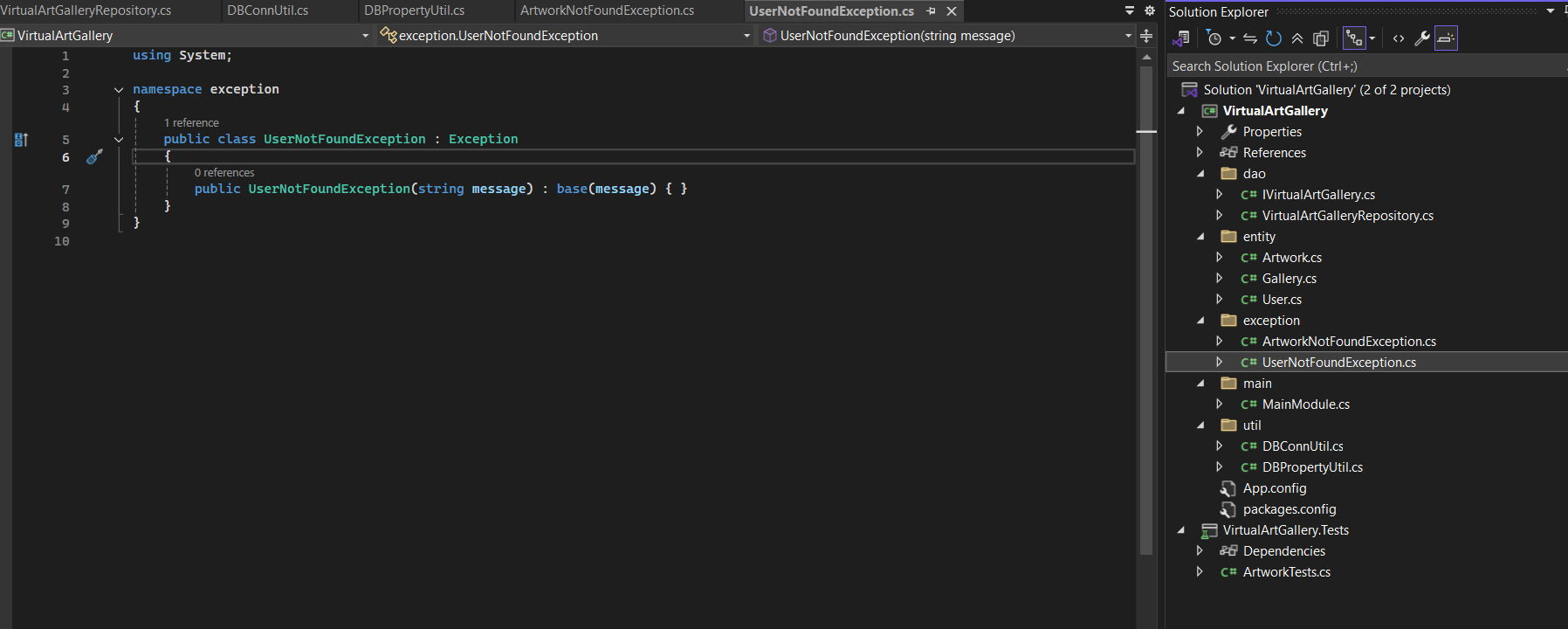
**Exception Handling**

Create the exceptions in package exceptions

**exceptions- ArtworkNotFoundException**



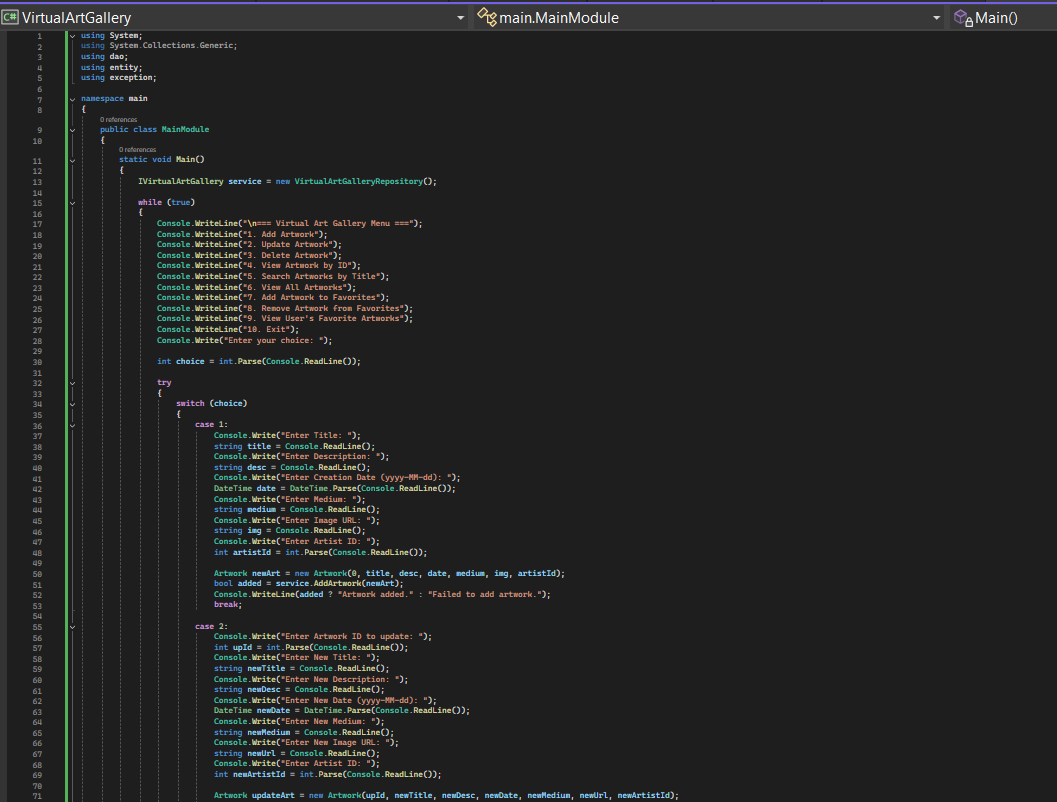
**exceptions- UserNotFoundException**

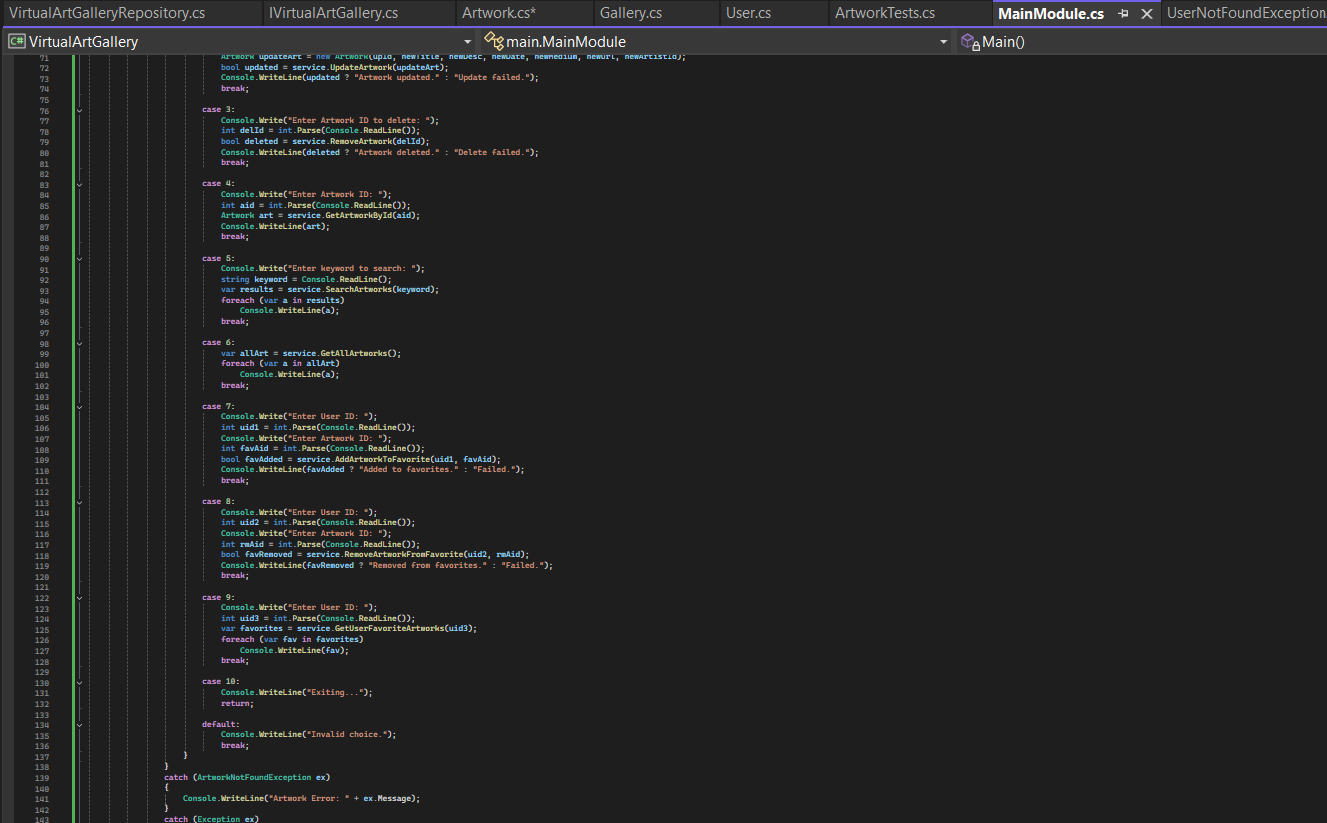


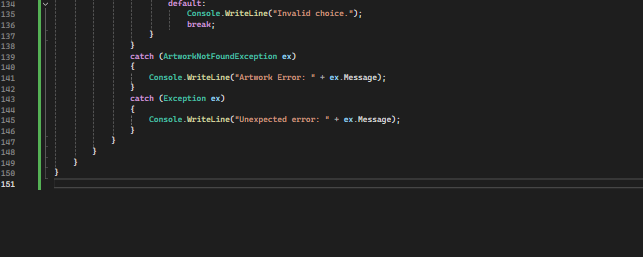
**Main Method**

Create class named MainModule with main method in main package.

**Main-MainModule**



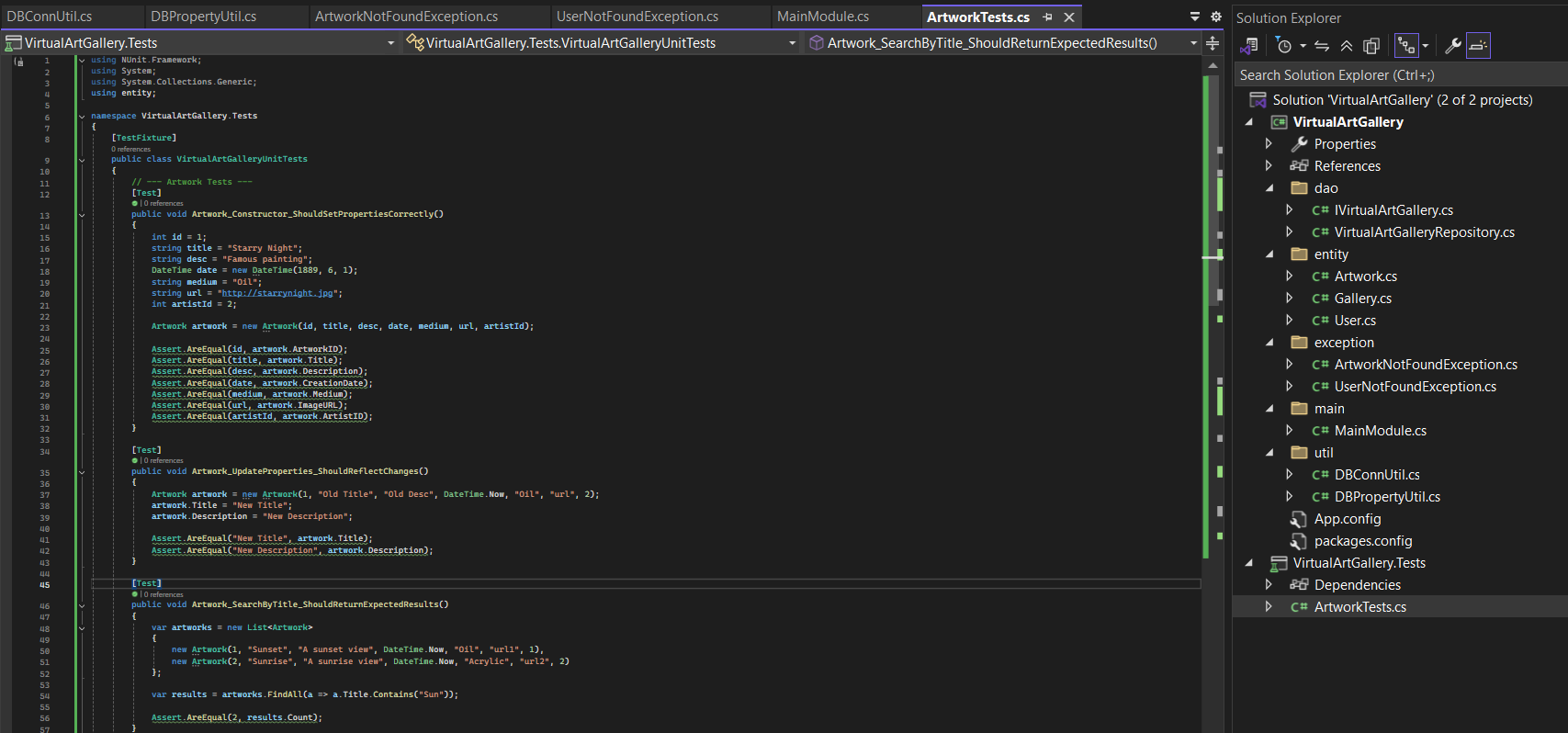


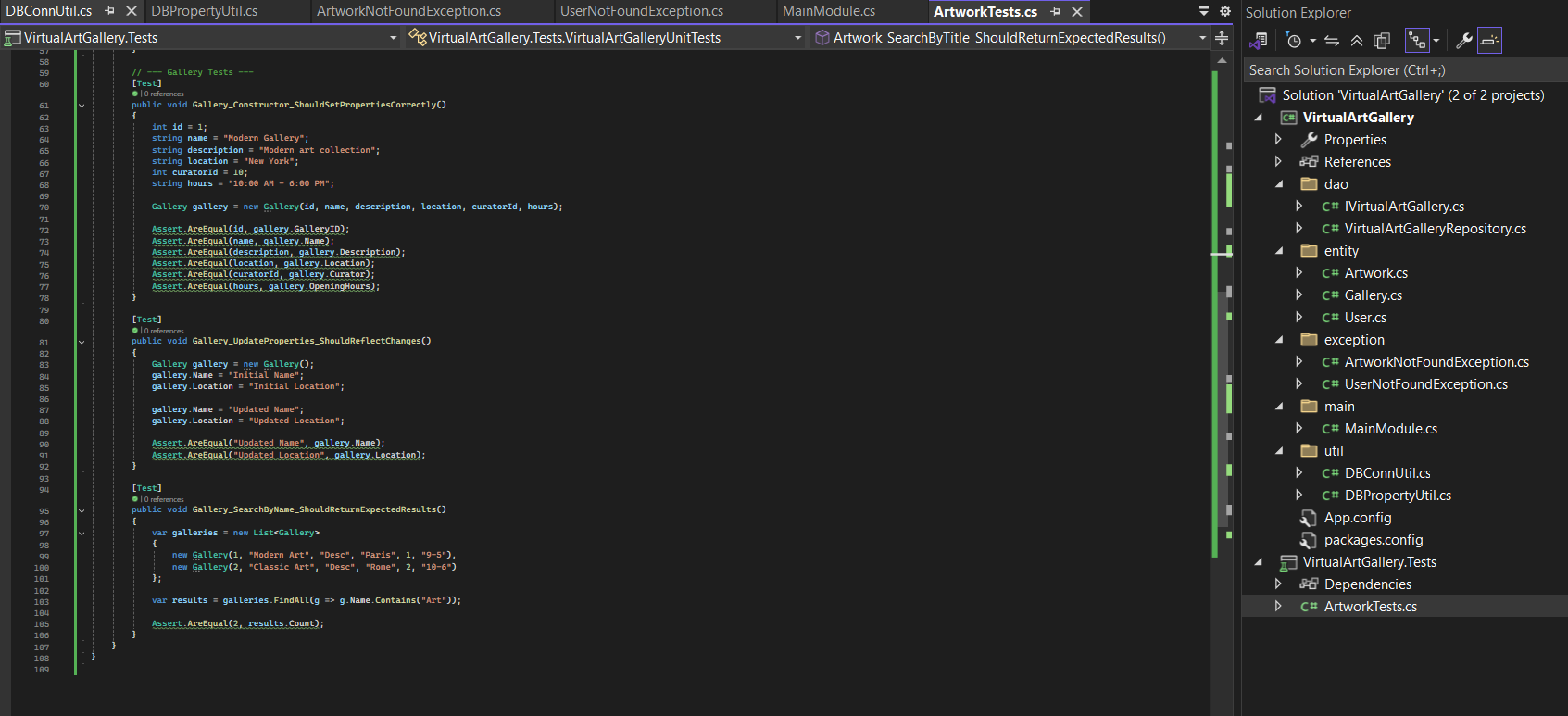


**Unit Testing**

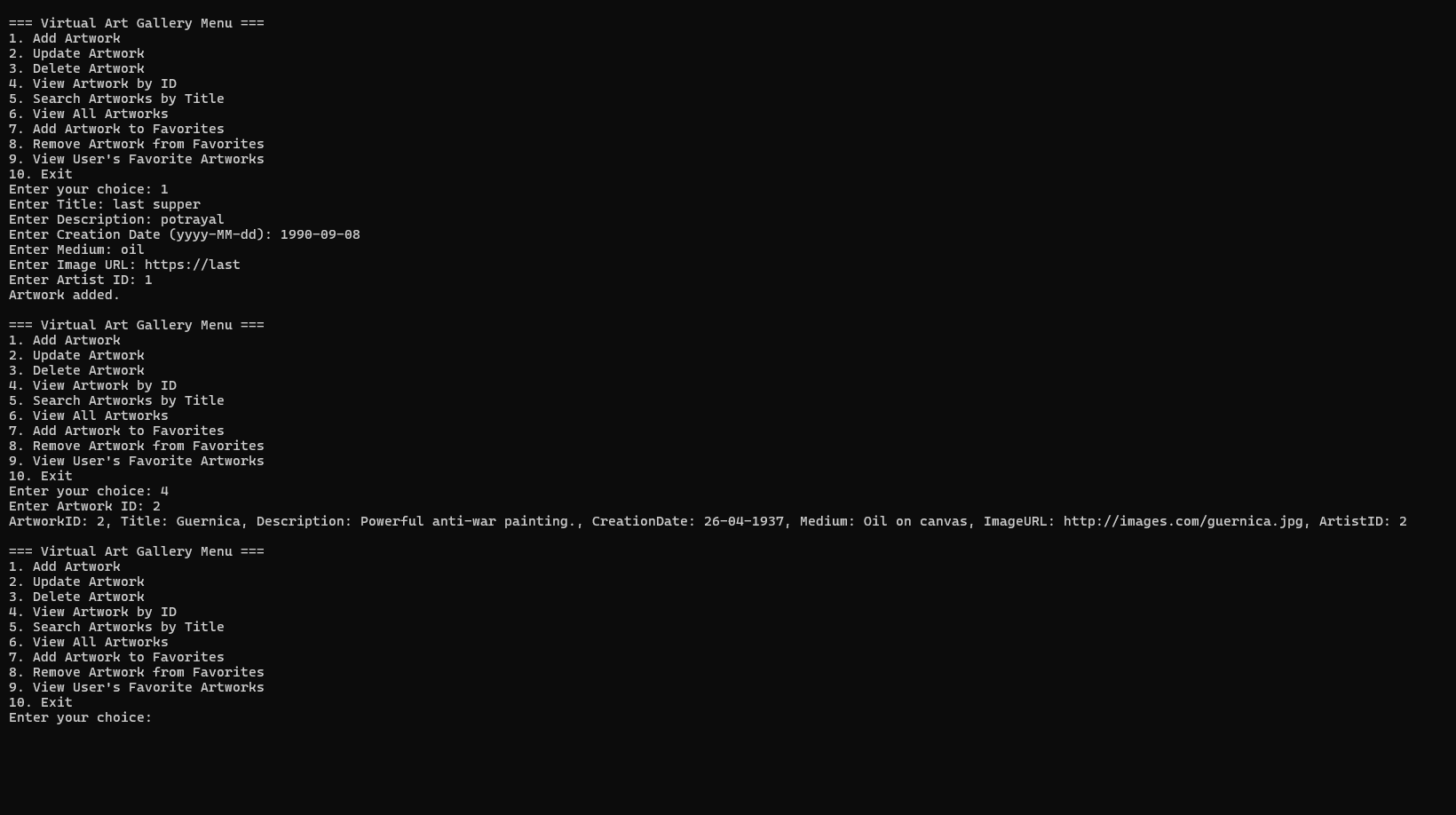
Creating Unit test cases for a Virtual Art Gallery system is essential to ensure that the system functions correctly.

**VirtualArtGallery.Tests- ArtworkTests.cs**





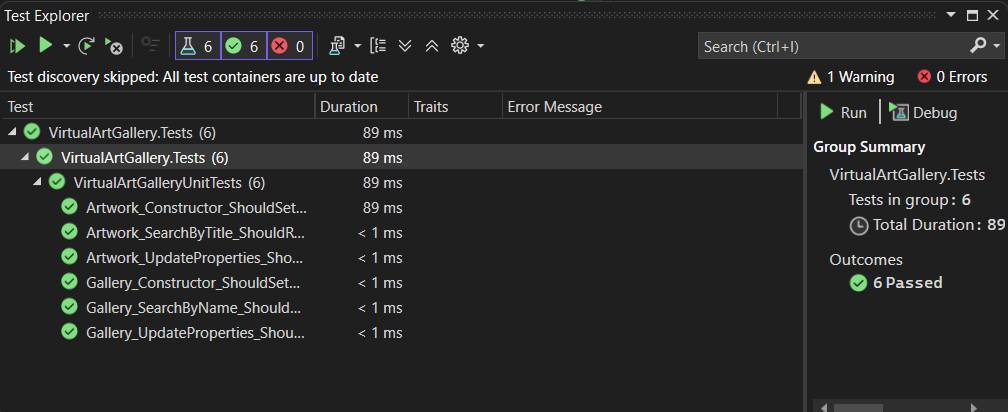
**OUTPUT:**

****

**Test Cases:**

1. Artwork Management

2. Gallery Management



**Code:**

**DataBase**

-- Create Database

CREATE DATABASE VirtualArtGalleryDB;

USE VirtualArtGalleryDB;

-- Artist Table

CREATE TABLE Artist (

ArtistID INT IDENTITY(1,1) PRIMARY KEY,

Name VARCHAR(100),

Biography VARCHAR(1000),

BirthDate DATE,

Nationality VARCHAR(100),

Website VARCHAR(200),

ContactInformation VARCHAR(200)

);

-- Artwork Table

CREATE TABLE Artwork (

ArtworkID INT IDENTITY(1,1) PRIMARY KEY,

Title VARCHAR(100),

Description VARCHAR(500),

CreationDate DATE,

Medium VARCHAR(100),

ImageURL VARCHAR(200),

ArtistID INT,

FOREIGN KEY (ArtistID) REFERENCES Artist(ArtistID)

);

-- User Table

CREATE TABLE [User] (

UserID INT IDENTITY(1,1) PRIMARY KEY,

Username VARCHAR(50),

[Password] VARCHAR(50),

Email VARCHAR(100),

FirstName VARCHAR(50),

LastName VARCHAR(50),

DateOfBirth DATE,

ProfilePicture VARCHAR(200)

);

-- Gallery Table

CREATE TABLE Gallery (

GalleryID INT IDENTITY(1,1) PRIMARY KEY,

Name VARCHAR(100),

Description VARCHAR(500),

Location VARCHAR(100),

Curator INT,

OpeningHours VARCHAR(100),

FOREIGN KEY (Curator) REFERENCES Artist(ArtistID)

);

-- User\_Favorite\_Artwork (Many-to-Many)

CREATE TABLE User\_Favorite\_Artwork (

UserID INT,

ArtworkID INT,

PRIMARY KEY (UserID, ArtworkID),

FOREIGN KEY (UserID) REFERENCES [User](UserID),

FOREIGN KEY (ArtworkID) REFERENCES Artwork(ArtworkID)

);

-- Artwork\_Gallery (Many-to-Many)

CREATE TABLE Artwork\_Gallery (

ArtworkID INT,

GalleryID INT,

PRIMARY KEY (ArtworkID, GalleryID),

FOREIGN KEY (ArtworkID) REFERENCES Artwork(ArtworkID),

FOREIGN KEY (GalleryID) REFERENCES Gallery(GalleryID)

);

-- Insert into Artist

INSERT INTO Artist (Name, Biography, BirthDate, Nationality, Website, ContactInformation)

VALUES

('Vincent van Gogh', 'Dutch post-impressionist painter.', '1853-03-30', 'Dutch', 'http://vangogh.com', 'contact@vangogh.com'),

('Pablo Picasso', 'Spanish painter and sculptor.', '1881-10-25', 'Spanish', 'http://picasso.com', 'contact@picasso.com');

-- Insert into Artwork

INSERT INTO Artwork (Title, Description, CreationDate, Medium, ImageURL, ArtistID)

VALUES

('Starry Night', 'A swirling night sky painting.', '1889-06-01', 'Oil on canvas', 'http://images.com/starry.jpg', 1),

('Guernica', 'Powerful anti-war painting.', '1937-04-26', 'Oil on canvas', 'http://images.com/guernica.jpg', 2);

-- Insert into User

INSERT INTO [User] (Username, [Password], Email, FirstName, LastName, DateOfBirth, ProfilePicture)

VALUES

('artlover01', 'pass123', 'artlover01@email.com', 'Anna', 'Smith', '1990-01-01', 'http://images.com/user1.jpg'),

('collector23', 'pass456', 'collector23@email.com', 'John', 'Doe', '1985-12-05', 'http://images.com/user2.jpg');

-- Insert into Gallery (Curator is ArtistID)

INSERT INTO Gallery (Name, Description, Location, Curator, OpeningHours)

VALUES

('Impressionist Hall', 'Gallery for impressionist artworks.', 'Paris', 1, '10:00 AM - 6:00 PM'),

('Modern Arts Wing', 'Modern and abstract artworks.', 'Madrid', 2, '11:00 AM - 7:00 PM');

-- Link artworks to galleries

INSERT INTO Artwork\_Gallery (ArtworkID, GalleryID)

VALUES

(1, 1),

(2, 2);

-- Link favorites (users -artworks)

INSERT INTO User\_Favorite\_Artwork (UserID, ArtworkID)

VALUES

(1, 1),

(2, 2);

**Entity-Artwork.cs**

using System;

using System.Configuration;

using System.Xml.Linq;

namespace entity

{

public class Artwork

{

public int ArtworkID { get; set; }

public string Title { get; set; }

public string Description { get; set; }

public DateTime CreationDate { get; set; }

public string Medium { get; set; }

public string ImageURL { get; set; }

public int ArtistID { get; set; }

public Artwork(int artworkID, string title, string description, DateTime creationDate, string medium, string imageURL, int artistID)

{

ArtworkID = artworkID;

Title = title;

Description = description;

CreationDate = creationDate;

Medium = medium;

ImageURL = imageURL;

ArtistID = artistID;

}

public override string ToString()

{

return $"ArtworkID: {ArtworkID}, Title: {Title}, Description: {Description}, " +

$"CreationDate: {CreationDate.ToShortDateString()}, Medium: {Medium}, " +

$"ImageURL: {ImageURL}, ArtistID: {ArtistID}";

}

}

}

**Entity-Gallery.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace entity

{

public class Gallery

{

public int GalleryID { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public string Location { get; set; }

public int Curator { get; set; }

public string OpeningHours { get; set; }

public Gallery() { }

public Gallery(int id, string name, string desc, string location, int curator, string hours)

{

GalleryID = id;

Name = name;

Description = desc;

Location = location;

Curator = curator;

OpeningHours = hours;

}

}

}

**Entity-Users.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace entity

{

public class User

{

public int UserID { get; set; }

public string Username { get; set; }

public string Password { get; set; }

public string Email { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public DateTime DateOfBirth { get; set; }

public string ProfilePicture { get; set; }

public User() { }

public User(int id, string username, string password, string email, string fname, string lname, DateTime dob, string pic)

{

UserID = id;

Username = username;

Password = password;

Email = email;

FirstName = fname;

LastName = lname;

DateOfBirth = dob;

ProfilePicture = pic;

}

}

}

**Dao- IVirtualArtGallery**

using System;

using System.Collections.Generic;

using entity;

namespace dao

{

public interface IVirtualArtGallery

{

// Artwork Management

bool AddArtwork(Artwork artwork);

bool UpdateArtwork(Artwork artwork);

bool RemoveArtwork(int artworkId);

Artwork GetArtworkById(int artworkId);

List<Artwork> SearchArtworks(string keyword);

List<Artwork> GetAllArtworks();

// User Favorites

bool AddArtworkToFavorite(int userId, int artworkId);

bool RemoveArtworkFromFavorite(int userId, int artworkId);

List<Artwork> GetUserFavoriteArtworks(int userId);

}

}

**Dao- VirtualArtGalleryRepository**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using entity;

using util;

using exception;

namespace dao

{

public class VirtualArtGalleryRepository : IVirtualArtGallery

{

SqlConnection conn;

public VirtualArtGalleryRepository()

{

conn = DBConnUtil.GetConnection();

}

public bool AddArtwork(Artwork artwork)

{

string sql = "INSERT INTO Artwork (Title, Description, CreationDate, Medium, ImageURL, ArtistID) VALUES (@title, @desc, @date, @medium, @url, @artistId)";

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.Parameters.AddWithValue("@title", artwork.Title);

cmd.Parameters.AddWithValue("@desc", artwork.Description);

cmd.Parameters.AddWithValue("@date", artwork.CreationDate);

cmd.Parameters.AddWithValue("@medium", artwork.Medium);

cmd.Parameters.AddWithValue("@url", artwork.ImageURL);

cmd.Parameters.AddWithValue("@artistId", artwork.ArtistID);

return cmd.ExecuteNonQuery() > 0;

}

public bool UpdateArtwork(Artwork artwork)

{

string sql = "UPDATE Artwork SET Title=@title, Description=@desc, CreationDate=@date, Medium=@medium, ImageURL=@url, ArtistID=@artistId WHERE ArtworkID=@id";

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.Parameters.AddWithValue("@title", artwork.Title);

cmd.Parameters.AddWithValue("@desc", artwork.Description);

cmd.Parameters.AddWithValue("@date", artwork.CreationDate);

cmd.Parameters.AddWithValue("@medium", artwork.Medium);

cmd.Parameters.AddWithValue("@url", artwork.ImageURL);

cmd.Parameters.AddWithValue("@artistId", artwork.ArtistID);

cmd.Parameters.AddWithValue("@id", artwork.ArtworkID);

return cmd.ExecuteNonQuery() > 0;

}

public bool RemoveArtwork(int artworkId)

{

SqlCommand cmd = new SqlCommand("DELETE FROM Artwork WHERE ArtworkID = @id", conn);

cmd.Parameters.AddWithValue("@id", artworkId);

return cmd.ExecuteNonQuery() > 0;

}

public Artwork GetArtworkById(int artworkId)

{

SqlCommand cmd = new SqlCommand("SELECT \* FROM Artwork WHERE ArtworkID = @id", conn);

cmd.Parameters.AddWithValue("@id", artworkId);

SqlDataReader reader = cmd.ExecuteReader();

if (reader.Read())

{

Artwork art = new Artwork(

(int)reader["ArtworkID"],

reader["Title"].ToString(),

reader["Description"].ToString(),

(DateTime)reader["CreationDate"],

reader["Medium"].ToString(),

reader["ImageURL"].ToString(),

(int)reader["ArtistID"]

);

reader.Close();

return art;

}

reader.Close();

throw new ArtworkNotFoundException("Artwork not found.");

}

public List<Artwork> SearchArtworks(string keyword)

{

List<Artwork> list = new List<Artwork>();

SqlCommand cmd = new SqlCommand("SELECT \* FROM Artwork WHERE Title LIKE @key", conn);

cmd.Parameters.AddWithValue("@key", "%" + keyword + "%");

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

list.Add(new Artwork(

(int)reader["ArtworkID"],

reader["Title"].ToString(),

reader["Description"].ToString(),

(DateTime)reader["CreationDate"],

reader["Medium"].ToString(),

reader["ImageURL"].ToString(),

(int)reader["ArtistID"]

));

}

reader.Close();

return list;

}

public List<Artwork> GetAllArtworks()

{

List<Artwork> list = new List<Artwork>();

SqlCommand cmd = new SqlCommand("SELECT \* FROM Artwork", conn);

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

list.Add(new Artwork(

(int)reader["ArtworkID"],

reader["Title"].ToString(),

reader["Description"].ToString(),

(DateTime)reader["CreationDate"],

reader["Medium"].ToString(),

reader["ImageURL"].ToString(),

(int)reader["ArtistID"]

));

}

reader.Close();

return list;

}

public bool AddArtworkToFavorite(int userId, int artworkId)

{

string sql = "INSERT INTO User\_Favorite\_Artwork (UserID, ArtworkID) VALUES (@userId, @artworkId)";

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.Parameters.AddWithValue("@userId", userId);

cmd.Parameters.AddWithValue("@artworkId", artworkId);

return cmd.ExecuteNonQuery() > 0;

}

public bool RemoveArtworkFromFavorite(int userId, int artworkId)

{

string sql = "DELETE FROM User\_Favorite\_Artwork WHERE UserID = @userId AND ArtworkID = @artworkId";

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.Parameters.AddWithValue("@userId", userId);

cmd.Parameters.AddWithValue("@artworkId", artworkId);

return cmd.ExecuteNonQuery() > 0;

}

public List<Artwork> GetUserFavoriteArtworks(int userId)

{

List<Artwork> list = new List<Artwork>();

string sql = @"SELECT a.\* FROM Artwork a

INNER JOIN User\_Favorite\_Artwork ufa ON a.ArtworkID = ufa.ArtworkID

WHERE ufa.UserID = @userId";

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.Parameters.AddWithValue("@userId", userId);

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

list.Add(new Artwork(

(int)reader["ArtworkID"],

reader["Title"].ToString(),

reader["Description"].ToString(),

(DateTime)reader["CreationDate"],

reader["Medium"].ToString(),

reader["ImageURL"].ToString(),

(int)reader["ArtistID"]

));

}

reader.Close();

return list;

}

}

}

**Util- DBConnUtil**

using System.Data.SqlClient;

namespace util

{

public static class DBConnUtil

{

public static SqlConnection GetConnection()

{

string connectionString = "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=VirtualArtGalleryDB;Integrated Security=True;";

SqlConnection conn = new SqlConnection(connectionString);

conn.Open();

return conn;

}

}

}

**Util- DBPropertyUtil**

using System.Configuration;

namespace util

{

public static class DBPropertyUtil

{

public static string GetConnectionString()

{

return ConfigurationManager.ConnectionStrings["VirtualArtGalleryDB"].ConnectionString;

}

}

}

**exceptions- ArtworkNotFoundException**

using System;

namespace exception

{

public class ArtworkNotFoundException : Exception

{

public ArtworkNotFoundException(string message) : base(message) { }

}

}

**exceptions- UserNotFoundException**

using System;

namespace exception

{

public class UserNotFoundException : Exception

{

public UserNotFoundException(string message) : base(message) { }

}

}

**Main-MainModule**

using System;

using System.Collections.Generic;

using dao;

using entity;

using exception;

namespace main

{

public class MainModule

{

static void Main()

{

IVirtualArtGallery service = new VirtualArtGalleryRepository();

while (true)

{

Console.WriteLine("\n=== Virtual Art Gallery Menu ===");

Console.WriteLine("1. Add Artwork");

Console.WriteLine("2. Update Artwork");

Console.WriteLine("3. Delete Artwork");

Console.WriteLine("4. View Artwork by ID");

Console.WriteLine("5. Search Artworks by Title");

Console.WriteLine("6. View All Artworks");

Console.WriteLine("7. Add Artwork to Favorites");

Console.WriteLine("8. Remove Artwork from Favorites");

Console.WriteLine("9. View User's Favorite Artworks");

Console.WriteLine("10. Exit");

Console.Write("Enter your choice: ");

int choice = int.Parse(Console.ReadLine());

try

{

switch (choice)

{

case 1:

Console.Write("Enter Title: ");

string title = Console.ReadLine();

Console.Write("Enter Description: ");

string desc = Console.ReadLine();

Console.Write("Enter Creation Date (yyyy-MM-dd): ");

DateTime date = DateTime.Parse(Console.ReadLine());

Console.Write("Enter Medium: ");

string medium = Console.ReadLine();

Console.Write("Enter Image URL: ");

string img = Console.ReadLine();

Console.Write("Enter Artist ID: ");

int artistId = int.Parse(Console.ReadLine());

Artwork newArt = new Artwork(0, title, desc, date, medium, img, artistId);

bool added = service.AddArtwork(newArt);

Console.WriteLine(added ? "Artwork added." : "Failed to add artwork.");

break;

case 2:

Console.Write("Enter Artwork ID to update: ");

int upId = int.Parse(Console.ReadLine());

Console.Write("Enter New Title: ");

string newTitle = Console.ReadLine();

Console.Write("Enter New Description: ");

string newDesc = Console.ReadLine();

Console.Write("Enter New Date (yyyy-MM-dd): ");

DateTime newDate = DateTime.Parse(Console.ReadLine());

Console.Write("Enter New Medium: ");

string newMedium = Console.ReadLine();

Console.Write("Enter New Image URL: ");

string newUrl = Console.ReadLine();

Console.Write("Enter Artist ID: ");

int newArtistId = int.Parse(Console.ReadLine());

Artwork updateArt = new Artwork(upId, newTitle, newDesc, newDate, newMedium, newUrl, newArtistId);

bool updated = service.UpdateArtwork(updateArt);

Console.WriteLine(updated ? "Artwork updated." : "Update failed.");

break;

case 3:

Console.Write("Enter Artwork ID to delete: ");

int delId = int.Parse(Console.ReadLine());

bool deleted = service.RemoveArtwork(delId);

Console.WriteLine(deleted ? "Artwork deleted." : "Delete failed.");

break;

case 4:

Console.Write("Enter Artwork ID: ");

int aid = int.Parse(Console.ReadLine());

Artwork art = service.GetArtworkById(aid);

Console.WriteLine(art);

break;

case 5:

Console.Write("Enter keyword to search: ");

string keyword = Console.ReadLine();

var results = service.SearchArtworks(keyword);

foreach (var a in results)

Console.WriteLine(a);

break;

case 6:

var allArt = service.GetAllArtworks();

foreach (var a in allArt)

Console.WriteLine(a);

break;

case 7:

Console.Write("Enter User ID: ");

int uid1 = int.Parse(Console.ReadLine());

Console.Write("Enter Artwork ID: ");

int favAid = int.Parse(Console.ReadLine());

bool favAdded = service.AddArtworkToFavorite(uid1, favAid);

Console.WriteLine(favAdded ? "Added to favorites." : "Failed.");

break;

case 8:

Console.Write("Enter User ID: ");

int uid2 = int.Parse(Console.ReadLine());

Console.Write("Enter Artwork ID: ");

int rmAid = int.Parse(Console.ReadLine());

bool favRemoved = service.RemoveArtworkFromFavorite(uid2, rmAid);

Console.WriteLine(favRemoved ? "Removed from favorites." : "Failed.");

break;

case 9:

Console.Write("Enter User ID: ");

int uid3 = int.Parse(Console.ReadLine());

var favorites = service.GetUserFavoriteArtworks(uid3);

foreach (var fav in favorites)

Console.WriteLine(fav);

break;

case 10:

Console.WriteLine("Exiting...");

return;

default:

Console.WriteLine("Invalid choice.");

break;

}

}

catch (ArtworkNotFoundException ex)

{

Console.WriteLine("Artwork Error: " + ex.Message);

}

catch (Exception ex)

{

Console.WriteLine("Unexpected error: " + ex.Message);

}

}

}

}

}

**VirtualArtGallery.Tests- ArtworkTests.cs**

using NUnit.Framework;

using System;

using System.Collections.Generic;

using entity;

namespace VirtualArtGallery.Tests

{

[TestFixture]

public class VirtualArtGalleryUnitTests

{

// --- Artwork Tests ---

[Test]

public void Artwork\_Constructor\_ShouldSetPropertiesCorrectly()

{

int id = 1;

string title = "Starry Night";

string desc = "Famous painting";

DateTime date = new DateTime(1889, 6, 1);

string medium = "Oil";

string url = "http://starrynight.jpg";

int artistId = 2;

Artwork artwork = new Artwork(id, title, desc, date, medium, url, artistId);

Assert.AreEqual(id, artwork.ArtworkID);

Assert.AreEqual(title, artwork.Title);

Assert.AreEqual(desc, artwork.Description);

Assert.AreEqual(date, artwork.CreationDate);

Assert.AreEqual(medium, artwork.Medium);

Assert.AreEqual(url, artwork.ImageURL);

Assert.AreEqual(artistId, artwork.ArtistID);

}

[Test]

public void Artwork\_UpdateProperties\_ShouldReflectChanges()

{

Artwork artwork = new Artwork(1, "Old Title", "Old Desc", DateTime.Now, "Oil", "url", 2);

artwork.Title = "New Title";

artwork.Description = "New Description";

Assert.AreEqual("New Title", artwork.Title);

Assert.AreEqual("New Description", artwork.Description);

}

[Test]

public void Artwork\_SearchByTitle\_ShouldReturnExpectedResults()

{

var artworks = new List<Artwork>

{

new Artwork(1, "Sunset", "A sunset view", DateTime.Now, "Oil", "url1", 1),

new Artwork(2, "Sunrise", "A sunrise view", DateTime.Now, "Acrylic", "url2", 2)

};

var results = artworks.FindAll(a => a.Title.Contains("Sun"));

Assert.AreEqual(2, results.Count);

}

// --- Gallery Tests ---

[Test]

public void Gallery\_Constructor\_ShouldSetPropertiesCorrectly()

{

int id = 1;

string name = "Modern Gallery";

string description = "Modern art collection";

string location = "New York";

int curatorId = 10;

string hours = "10:00 AM - 6:00 PM";

Gallery gallery = new Gallery(id, name, description, location, curatorId, hours);

Assert.AreEqual(id, gallery.GalleryID);

Assert.AreEqual(name, gallery.Name);

Assert.AreEqual(description, gallery.Description);

Assert.AreEqual(location, gallery.Location);

Assert.AreEqual(curatorId, gallery.Curator);

Assert.AreEqual(hours, gallery.OpeningHours);

}

[Test]

public void Gallery\_UpdateProperties\_ShouldReflectChanges()

{

Gallery gallery = new Gallery();

gallery.Name = "Initial Name";

gallery.Location = "Initial Location";

gallery.Name = "Updated Name";

gallery.Location = "Updated Location";

Assert.AreEqual("Updated Name", gallery.Name);

Assert.AreEqual("Updated Location", gallery.Location);

}

[Test]

public void Gallery\_SearchByName\_ShouldReturnExpectedResults()

{

var galleries = new List<Gallery>

{

new Gallery(1, "Modern Art", "Desc", "Paris", 1, "9-5"),

new Gallery(2, "Classic Art", "Desc", "Rome", 2, "10-6")

};

var results = galleries.FindAll(g => g.Name.Contains("Art"));

Assert.AreEqual(2, results.Count);

}

}

}