using System;

using System.Collections.Generic;

using System.Threading.Tasks;

using Capgemini.GreatOutdoors.BusinessLayer;

using Capgemini.GreatOutdoors.Entities;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace Capgemini.GreatOutdoors.UnitTest

{

[TestClass]

public class AddRetailerBLTest

{

/// <summary>

/// Add Retailer to the collection if it is valid

/// </summary>

[TestMethod]

public async Task AddValidRetailer()

{

//Arrange

RetailerBL retailerBL = new RetailerBL();

Retailer retailer = new Retailer() { RetailerName = "Sourav", Email = "sourav@capgemini.com", Password = "Sarthak123", RetailerMobile = "8897476406" };

bool isAdded = false;

string errorMessage = null;

//Add

try

{

isAdded = await retailerBL.AddRetailerBL(retailer);

}

catch(Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsTrue(isAdded, errorMessage);

}

}

/// <summary>

/// Retailer name can not be null

/// </summary>

[TestMethod]

public async Task RetailerNameCanNotBeNull()

{

//Arrange

RetailerBL retailerBL = new RetailerBL();

Retailer retailer = new Retailer() { RetailerName = null, Email = "sarthak@capgemini.com", Password = "Sarthak123", RetailerMobile = "8897476406" };

bool isAdded = false;

string errorMessage = null;

//Add

try

{

isAdded = await retailerBL.AddRetailerBL(retailer);

}

catch(Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally{

//assert

Assert.IsFalse(isAdded, errorMessage);

}

}

/// <summary>

/// Retailer Mobile can't be null

/// </summary>

[TestMethod]

public async Task RetailerMobileCanNotBeNull()

{

//Arrange

RetailerBL retailerBL = new RetailerBL();

Retailer retailer = new Retailer() { RetailerName = "Smith", RetailerMobile = null, Password = "Smith123#", Email = "smith@gmail.com" };

bool isAdded = false;

string errorMessage = null;

//Act

try

{

isAdded = await retailerBL.AddRetailerBL(retailer);

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsFalse(isAdded, errorMessage);

}

}

/// <summary>

/// Retailer Password can't be null

/// </summary>

[TestMethod]

public async Task RetailerPasswordCanNotBeNull()

{

//Arrange

RetailerBL retailerBL = new RetailerBL();

Retailer retailer = new Retailer() { RetailerName = "Allen", RetailerMobile = "9877766554", Password = null, Email = "allen@gmail.com" };

bool isAdded = false;

string errorMessage = null;

//Act

try

{

isAdded = await retailerBL.AddRetailerBL(retailer);

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsFalse(isAdded, errorMessage);

}

}

/// <summary>

/// Retailer Email can't be null

/// </summary>

[TestMethod]

public async Task RetailerEmailCanNotBeNull()

{

//Arrange

RetailerBL retailerBL = new RetailerBL();

Retailer retailer = new Retailer() { RetailerName = "John", RetailerMobile = "9876543210", Password = "John123#", Email = null };

bool isAdded = false;

string errorMessage = null;

//Act

try

{

isAdded = await retailerBL.AddRetailerBL(retailer);

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsFalse(isAdded, errorMessage);

}

}

/// <summary>

/// RetailerName should contain at least two characters

/// </summary>

[TestMethod]

public async Task RetailerNameShouldContainAtLeastTwoCharacters()

{

//Arrange

RetailerBL retailerBL = new RetailerBL();

Retailer retailer = new Retailer() { RetailerName = "J", RetailerMobile = "9877897890", Password = "John123#", Email = "john@gmail.com" };

bool isAdded = false;

string errorMessage = null;

//Act

try

{

isAdded = await retailerBL.AddRetailerBL(retailer);

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsFalse(isAdded, errorMessage);

}

}

/// <summary>

/// RetailerMobile should be a valid mobile number

/// </summary>

[TestMethod]

public async Task RetailerMobileRegExp()

{

//Arrange

RetailerBL retailerBL = new RetailerBL();

Retailer retailer = new Retailer() { RetailerName = "John", RetailerMobile = "9877", Password = "John123#", Email = "john@gmail.com" };

bool isAdded = false;

string errorMessage = null;

//Act

try

{

isAdded = await retailerBL.AddRetailerBL(retailer);

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsFalse(isAdded, errorMessage);

}

}

/// <summary>

/// Password should be a valid password as per regular expression

/// </summary>

[TestMethod]

public async Task RetailerPasswordRegExp()

{

//Arrange

RetailerBL retailerBL = new RetailerBL();

Retailer retailer = new Retailer() { RetailerName = "John", RetailerMobile = "9877897890", Password = "John", Email = "john@gmail.com" };

bool isAdded = false;

string errorMessage = null;

//Act

try

{

isAdded = await retailerBL.AddRetailerBL(retailer);

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsFalse(isAdded, errorMessage);

}

}

/// <summary>

/// Email should be a valid email as per regular expression

/// </summary>

[TestMethod]

public async Task RetailerEmailRegExp()

{

//Arrange

RetailerBL retailerBL = new RetailerBL();

Retailer retailer = new Retailer() { RetailerName = "John", RetailerMobile = "9877897890", Password = "John123#", Email = "john" };

bool isAdded = false;

string errorMessage = null;

//Act

try

{

isAdded = await retailerBL.AddRetailerBL(retailer);

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsFalse(isAdded, errorMessage);

}

}

[TestClass]

public class GetAllRetailerByBL

{

///<summary>

/// Retrieve all retailers

///</summary>

public async Task ValidRetailerName()

{

//arrange

RetailerBL retailerBL = new RetailerBL();

Retailer retailer = new Retailer() { RetailerName = "Devansh", RetailerMobile = "70234511335", Email = "devansh@gmail.com", Password = "Devansh123" };

bool isAddded = false;

bool valuesGot = false;

string errorMessage = null;

List<Retailer> retailers = retailerBL.GetAllRetailersBL();

= await retailerBL.AddRetailerBL(retailer);

//act

//assert

}

}

}

}