|  |  |
| --- | --- |
| ***Название проекта*** | Телефонный справочник |
| ***Номер версии*** | №1 |
| ***Имя тестера*** | Максим |
| ***Даты тестирования*** | 16.05.2024 |

*Тестирование приложения (Телефонный справочник)*

*Тестирование приложения с использованием TestCase*

|  |  |
| --- | --- |
| ***TestCase #*** | 1 |
| ***Приоритет теста*** | Высокий |
| ***Название тестирования/Имя*** | Добавление нового контакта |
| ***Резюме испытания*** | Проверка возможности добавления нового контакта в телефонный справочник |
| ***Шаги тестирования*** | 1. Открыть приложение телефонный справочник  2. Нажать на кнопку "Добавить контакт"  3. Заполнить все поля (имя, фамилия, телефонный номер)  4. Нажать на кнопку "Сохранить" |
| ***Данные тестирования*** | - Имя: Иван  - Фамилия: Иванов  - Телефонный номер: 1234567890 |
| ***Ожидаемый результат*** | Новый контакт успешно добавлен в телефонный справочник |
| ***Фактический результат*** | Новый контакт успешно добавлен в телефонный справочник |
| ***Предпосылки*** | Приложение телефонный справочник должно быть установлено и запущено |
| ***Постусловия*** | Новый контакт должен быть добавлен в телефонный справочник |
| ***Статус (Pass/Fail)*** | Pass |
| ***Комментарии*** | Тестирование (Добавление нового контакта) прошло успешно |

Визуальное тестирование приложения

|  |  |  |
| --- | --- | --- |
| ***Подготовка к тестированию*** | Убедитесь, что приложение установлено и запущено на устройстве, на котором будет проводиться тестирование. |  |
| ***Проверка главной страницы*** | Откройте приложение и проверьте, что все основные элементы, такие как заголовок, поиск, кнопка добавления контакта, список контактов, отображаются корректно и имеют соответствующий дизайн. |  |
| ***Проверка поиска контактов*** | Введите имя или номер телефона в поле поиска и убедитесь, что результаты поиска отображаются корректно. Проверьте, что поиск работает как в пустой базе данных, так и с заполненной. |  |
| ***Проверка добавления контакта*** | Нажмите на кнопку добавления контакта и заполните все поля формы. Убедитесь, что все поля имеют правильные метки, их содержимое сохраняется и новый контакт появляется в списке. |  |
| ***Проверка редактирования контакта*** | Выберите контакт из списка и нажмите на кнопку редактирования. Измените информацию и сохраните. Убедитесь, что изменения сохранены, и контакт обновлен в списке. |  |
| ***Проверка удаления контакта*** | Выберите контакт и нажмите на кнопку удаления. Подтвердите удаление и убедитесь, что контакт удален из списка. |  |
| ***Проверка работы приложения на разных устройствах и в разных разрешениях экрана*** | Убедитесь, что приложение корректно отображается и функционирует на разных устройствах с разными размерами экрана и ориентацией. |  |
| ***Проверка состояния загрузки*** | Убедитесь, что при загрузке или обновлении данных приложение отображает соответствующую индикацию, сообщение о загрузке. |  |
| ***Проверка состояния ошибки*** | Попытайтесь вызвать ошибку, например, удалив все контакты или введя некорректные данные. |  |
| ***Проверка работы с пробелами и специальными символами*** | Введите пробелы и специальные символы в поля формы и убедитесь, что приложение корректно обрабатывает такие данные. |  |
| ***Вывод*** | Вывод по работе приложения | Приложение работает успешно. Без видимых ошибок и вылетов. |

*Тестирование с помощью UnitTest*

**Код:**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Moq;

using System;

using System.Collections.Generic;

using Телефонный\_справочник;

namespace Телефонный\_справочник\_Tests

{

[TestClass]

public class UnitTest1

{

private Mock<IContactRepository> \_mockContactRepository;

private PhoneBookService \_phoneBookService;

[TestMethod]

public void Setup()

{

\_mockContactRepository = new Mock<IContactRepository>();

\_phoneBookService = new PhoneBookService(\_mockContactRepository.Object);

}

public void AddContact\_WhenCalled\_ShouldAddContactToRepository()

{

// Arrange

var contact = new Contact { Name = "John Doe", PhoneNumber = "1234567890" };

// Act

\_phoneBookService.AddContact(contact);

// Assert

\_mockContactRepository.Verify(repo => repo.Add(contact), Times.Once);

}

public void GetAllContacts\_WhenCalled\_ShouldReturnAllContactsFromRepository()

{

// Arrange

var contacts = new List<Contact>

{

new Contact { Name = "John Doe", PhoneNumber = "1234567890" },

new Contact { Name = "Jane Doe", PhoneNumber = "0987654321" }

};

\_mockContactRepository.Setup(repo => repo.GetAllContacts(contacts));

// Act

var result = \_phoneBookService.GetAllContacts();

// Assert

Assert.AreEqual(contacts, result);

}

}

internal class Contact

{

public string Name { get; set; }

public string PhoneNumber { get; set; }

}

internal class PhoneBookService

{

private IContactRepository @object;

public PhoneBookService(IContactRepository @object)

{

this.@object = @object;

}

internal void AddContact(Contact contact)

{

throw new NotImplementedException();

}

internal List<Contact> GetAllContacts()

{

throw new NotImplementedException();

}

}

internal interface IContactRepository

{

void Add(Contact contact);

void GetAllContacts(List<Contact> contacts);

}

}

***Результат:***

