**SweetShop.Tests – Unit Test Report**

**Project:** SweetShop.Mvc, SweetShop.API  
**Test Framework:** xUnit  
**Mocking Framework:** Moq  
**Author:** Zaheer Abbas  
**Date:** 20-Sep-2025

**1. Overview**

The SweetShop.Tests project contains unit tests for the SweetsController in the SweetShop.Mvc project. The tests are designed to verify the following controller actions:

1. Displaying all sweets (Shop action).
2. Creating a new sweet (Create action).
3. Editing an existing sweet (Edit action).
4. Deleting a sweet (DeleteConfirmed action).

Moq is used to mock the SweetService dependency, and the tests focus on verifying service calls and returned results.

**2. Test Environment**

* **Framework:** .NET Core / ASP.NET Core MVC
* **Unit Test Framework:** xUnit
* **Mocking Library:** Moq
* **Web Host Environment Mock:** IWebHostEnvironment with temporary path
* **Tested Controller:** SweetsController

**3. Test Cases**

| **Test Case ID** | **Method** | **Description** | **Input** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC01 | Shop | Returns all sweets | null filters | List<Sweet> with 2 items | Pass |
| TC02 | Create (POST) | Adds a new sweet | Sweet{Name="Barfi", Price=40} | Redirect to AdminIndex; AddSweetAsync called once | Pass |
| TC03 | Edit (POST) | Updates an existing sweet | Sweet{Id=1, Name="Ladoo Special"} | Redirect to AdminIndex; UpdateSweetAsync called once | Pass |
| TC04 | DeleteConfirmed | Deletes a sweet by ID | SweetID=1 | Redirect to AdminIndex; DeleteSweetAsync called once | Pass |

**4. Test Details**

**4.1 Shop\_ShouldReturnAllSweets**

* **Purpose:** Verify that the Shop action returns all sweets from the service.
* **Mock Setup:** GetAllSweetsAsync() returns 2 sweets.
* **Assertion:**

Assert.Equal(2, sweets.Count);

* **Result:** Pass – 2 sweets returned successfully.

**4.2 Create\_Post\_ShouldCallAddSweetAsync**

* **Purpose:** Verify that posting a new sweet calls AddSweetAsync and redirects.
* **Mock Setup:** AddSweetAsync(It.IsAny<Sweet>()) returns true.
* **Action:** POST Create(sweet)
* **Assertions:**

Assert.Equal("AdminIndex", result.ActionName);

mockService.Verify(s => s.AddSweetAsync(It.Is<Sweet>(sw => sw.Name == "Barfi")), Times.Once);

* **Result:** Pass – Sweet added and redirect verified.

**4.3 Edit\_Post\_ShouldCallUpdateSweetAsync**

* **Purpose:** Verify that editing a sweet calls UpdateSweetAsync and redirects.
* **Mock Setup:** UpdateSweetAsync(It.IsAny<Sweet>()) returns true.
* **Action:** POST Edit(sweet)
* **Assertions:**

Assert.Equal("AdminIndex", result.ActionName);

mockService.Verify(s => s.UpdateSweetAsync(It.Is<Sweet>(sw => sw.Id == 1)), Times.Once);

* **Result:** Pass – Sweet updated and redirect verified.

**4.4 DeleteConfirmed\_ShouldCallDeleteSweetAsync**

* **Purpose:** Verify that deleting a sweet calls DeleteSweetAsync and redirects.
* **Mock Setup:** DeleteSweetAsync(It.IsAny<int>()) returns true.
* **Action:** DeleteConfirmed(1)
* **Assertions:**

Assert.Equal("AdminIndex", result.ActionName);

mockService.Verify(s => s.DeleteSweetAsync(1), Times.Once);

* **Result:** Pass – Sweet deleted and redirect verified.

**5. Summary**

All unit tests for the SweetsController passed successfully.

**Total Tests:** 4  
**Passed:** 4  
**Failed:** 0  
**Pass Percentage:** 100%

**Conclusion:** The controller methods are correctly invoking service methods and returning the expected results. The basic CRUD operations for sweets are verified and confirmed to be functional in isolation.