# **Unit Test Record**

## Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Revision Description** |
| V1.0 | February 28, 2014 | Suwichak Fungprasert | Create User unit test record |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Reviewer

|  |  |  |
| --- | --- | --- |
| **Name** | **Version Approved** | **Date** |
| Mr.Tanadol Parnong  Ms.Chonticha Prachpreang  Mr. Suwichak Fungprasert | 1.0 | February 28, 2014 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Table of Contents**

**Unit Test Record** 1

Revision history 1

Reviewer 1

**References** 3

**Introduction** 3

**Responsibility** 3

**Test Item** 3

**Test tools** 3

**Test Environment requirement** 3

**Features to be tested** 3

The Appendix A 4

Test case of Unit Test (UTC) 5

**UTC\_TC 01** 5

**UTC\_TC 02** 7

**UTC\_TC 03** 10

**UTC\_TC 04** 12

**UTC\_TC 05** 15

**UTC\_TC 06** 17

**UTC\_TC 07** 18

**UTC\_TC 08** 19

**UTC\_TC 09** 20

**UTC\_TC 10** 28

**UTC\_TC 11** 30

**UTC\_TC 12** 31

**UTC\_TC 13** 36

**UTC\_TC 14** 37

**UTC\_TC 15** 38

**UTC\_TC 16** 39

**UTC\_TC 17** 41

**UTC\_TC 18** 42

# **References**

User requirement specification

# **Introduction**

The unit testing about verification method in each the sequence diagram.

# **Responsibility**

|  |  |
| --- | --- |
| **Task** | **Team member name** |
| Write UTC test plan | Tanadol Parn-ong |
| Verify UTC test plan | Suwichak Fungprasertkul |

# **Test Item**

Media E-shopping v1.0

# **Test tools**

Microsoft Visual Studio 2013

# **Test Environment requirement**

Windows 7 SP1 or Windows 8

# **Features to be tested**

All of User requirement specification

# The Appendix A

admin

|  |  |  |
| --- | --- | --- |
| adminId | adminUsername | adminPassword |
| 1 | “admin” | “1234” |

Customers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| customerId | customerUsername | customerPassword | customerName | customerAddress | customerPicture |
| 1 | “milk” | “milk” | Suwichak Fungprasertkul | New York | cdp2.jpg |

payment

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| payment Id | ShoppingCartId | paymentType | paymentNumber | Total price | peceivedprice | paymentStatus |
| 1 | 1 | 2 | 6533332 | 2000.00 | NULL | 0 |

product

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| productId | productName | productDescription | productPrice | productPicture | isDeleted |
| 1 | “product1” | “It is good” | 2000.00 | pic1.JPG | 0 |

shoppingcart

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| shoppingcartId | customerId | status | Create\_date | Checkout\_date |
| 1 | 1 | 2 | 2014-02-09 13:34:50 | 2014-02-09 13:35:04 |

shoppinglist

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| shoppingListId | shoppingcartId | productId | Quantity shoppingProduct | isDelete |
| 1 | 1 | 1 | 1 | 0 |

# Test case of Unit Test (UTC)

## **UTC\_TC 01**

**Test scenario**

: The test case is for check the equal value between user input and data in the database.

**UTC\_TC01**

public ActionResult LoginAdmin(String adminUsername, String adminPassword)

**Prerequisites or Test input**

: The test data is available at Appendix A

**UTC\_TC 01 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 01 | 1 | Test with the valid data in the database.  **If(Session["admin"] == null)** | 1 | //Arrange  adminController controller = new adminController();  //Act  var result = controller.LoginAdmin("admin", "1234");  //Assert  result.AssertActionRedirect().ToAction("Index"); | adminUsername = “admin”, adminPassword = “1234” | RedirectToAction.RouteValues.Value[0] = “Index” | RedirectToAction.RouteValues.Value[0] = “Index” | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 01 | 2 | Test with the valid data in the database.  **If(Session["admin"] != null)** | 1 | //Arrange  adminController controller = new adminController();  //Act  var result = controller.LoginAdmin("admin", "1234");  //Assert  result.AssertActionRedirect().ToAction("Index"); | adminUsername = “admin”, adminPassword = “1234” | RedirectToAction.RouteValues.Value[0] = “Index” | RedirectToAction.RouteValues.Value[0] = “Index” | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 01 | 3 | Test with the invalid data in the database. | 1 | //Arrange  adminController controller = new adminController();  //Act  var result = controller.LoginAdmin("admin", "46546") as ViewResult;  //Assert  Assert.AreEqual("Invalid Username and/or Password", result.ViewBag.LoginIssue); | adminUsername = “admin”, adminPassword = “12” | View.base.Values.[0] = “Invalid Username and/or Password" | View.base.Values.[0] = “Invalid Username and/or Password" | Pass |

## **UTC\_TC 02**

**Test scenario**

: The test case is for check the add product by inputting product infomation. Which the method addProduct has been celled.

**UTC\_TC02**

public ActionResult AddProduct(String productName, String productDescription, Double productPrice, HttpPostedFileBase file)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 02 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 02 | 1 | User input valid information. Which the method addProduct has been celled. | 1 | //Arrange  adminController controller = new adminController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic1.JPG");  file.Setup(x => x.ContentType).Returns("jpeg");  //Act  var result = controller.AddProduct("product1", "It is good", 2000.00, file.Object);  //Assert  result.AssertActionRedirect().ToAction("ProductList"); | productName = “product1”, productDescription = (“It is good”), productPrice = 2000.00, file = pic1.JPG | RedirectToAction.RouteValues.Value[0] = “ProductList” | RedirectToAction.RouteValues.Value[0] = “ProductList” | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 02 | 2 | User input invalid information (**productPrice format invalid**). Which the method addProduct has been celled. | 1 | //Arrange  adminController controller = new adminController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic1.JPG");  file.Setup(x => x.ContentType).Returns("jpeg");  //Act  var result = controller.AddProduct("product1", "It is good", 1000.1111, file.Object) as ViewResult;  //Assert  Assert.AreEqual("The product price must be only number with 2 decimal point", result.ViewBag.AddProductIssue); | productName = “product1”, productDescription = (“It is good”), productPrice = 1000.1111, file = pic1.JPG | View.base.ViewBag.ViewData.Value[0] = “The product price must be only number with 2 decimal point " | View.base.ViewBag.ViewData.Value[0] = “The product price must be only number with 2 decimal point " | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 02 | 3 | User input invalid information (**product picture format invalid**). Which the method addProduct has been celled. | 1 | //Arrange  adminController controller = new adminController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic1.gif");  file.Setup(x => x.ContentType).Returns("gif");  //Act  var result = controller.AddProduct("product1", "It is good", 2000, file.Object) as ViewResult;  //Assert  Assert.AreEqual("No picture file is uploaded", result.ViewBag.AddProductIssue); | productName = “product1”, productDescription = (“It is good”), productPrice = 2000, file = pic1.gif | View.base.ViewBag.ViewData.Value[0] = “No picture file is uploaded " | View.base.ViewBag.ViewData.Value[0] = “No picture file is uploaded " | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 02 | 4 | User input invalid information (**not have product picture file**). Which the method addProduct has been celled. | 1 | //Arrange  adminController controller = new adminController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic1.gif");  file.Setup(x => x.ContentType).Returns("gif");  //Act  var result = controller.AddProduct("product1", "It is good", 2190, null) as ViewResult;  //Assert  Assert.AreEqual("No picture file is uploaded", result.ViewBag.AddProductIssue); | productName = “product1”, productDescription = (“It is good”), productPrice = 2190, file = None | View.base.ViewBag.ViewData.Value[0] = “No picture file is uploaded " | View.base.ViewBag.ViewData.Value[0] = “No picture file is uploaded " | Pass |

## **UTC\_TC 03**

**Test scenario**

: The test case is for check the display product information. Which the method productInfo has been called.

**UTC\_TC03**

public ActionResult ProductInfo(int id)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 03 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 03 | 1 | To test the display product information. Which the method productInfo has been celled. | 1 | //Arrange  adminController controller = new adminController();  //Act  var viewresult = controller.ProductInfo(1) as ViewResult;  product result = (product)viewresult.Model;  //Assert  Assert.AreEqual(1, result.productId);  Assert.AreEqual("product1", result.productName);  Assert.AreEqual("It is good", result.productDescription);  Assert.AreEqual("pic1.JPG", result.productPicture); | id = 1 | View.base.ViewData.Model = [productId = 1, productName = “product1”, productDescription = (“It is good”), productPrice = 2000.00, productPicture = pic1.JPG] | View.base.ViewData.Model = [productId = 1, productName = “product1”, productDescription = (“It is good”), productPrice = 2000.00, productPicture = pic1.JPG] | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 03 | 2 | To test the display product information. Which the method productInfo has been celled. (**Not have a product**) | 1 | //Arrange  adminController controller = new adminController();  //Act  var result = controller.ProductInfo(2000) as HttpNotFoundResult;  //Assert  Assert.AreEqual(404, result.StatusCode); | Id = 2000 | HttpNotFound.base.StatusCode = 404 | HttpNotFound.base.StatusCode = 404 | Pass |

## **UTC\_TC 04**

**Test scenario**

: The test case is for check the edit product information by user. Which the method productInfo has been celled.

**UTC\_TC04**

public ActionResult ProductInfo(int productId, String productName, String productDescription, Double productPrice, HttpPostedFileBase file)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 04 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 04 | 1 | To test the edit product information by user. Which the method productInfo has been celled. | 1 | //Arrange  adminController controller = new adminController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic1.JPG");  file.Setup(x => x.ContentType).Returns("jpeg");  //Act  var result = controller.ProductInfo(1,"product1", "It is good", 2000.00, file.Object);  //Assert  result.AssertActionRedirect().ToAction("ProductList"); | productId = “1”,  productName = “product1”, productDescription = (“It is good”), productPrice = 2000.00, file = pic1.JPG | RedirectToAction.RouteValues.Value[0] = “ProductList” | RedirectToAction.RouteValues.Value[0] = “ProductList” | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 04 | 2 | To test the edit product information by user. Which the method productInfo has been celled. | 1 | //Arrange  adminController controller = new adminController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic1.JPG");  file.Setup(x => x.ContentType).Returns("jpeg");  //Act  var result = controller.ProductInfo(1, "product1", "It is good", 1000.1111, file.Object) as ViewResult;  //Assert  Assert.AreEqual("The product price must be only number with 2 decimal point", result.ViewBag.AddProductIssue); | productId = “1”,  productName = “product1”, productDescription = (“It is good”), productPrice = 1000.1111, file = pic1.JPG | View.base.Values.[0] = “The product price must be only number with 2 decimal point " | View.base.Values.[0] = “The product price must be only number with 2 decimal point " | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 04 | 3 | To test the edit product information by user. Which the method productInfo has been celled. | 1 | //Arrange  adminController controller = new adminController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic1.GIF");  file.Setup(x => x.ContentType).Returns("gif");  //Act  var result = controller.ProductInfo(1, "product1", "It is good", 2000.00, file.Object) as ViewResult;  //Assert  Assert.AreEqual("No picture file is uploaded", result.ViewBag.AddProductIssue); | productId = “1”,  productName = “product1”, productDescription = (“It is good”), productPrice = 2000.00, file = pic1.gif | View.base.Values.[0] = “No picture file is uploaded " | View.base.Values.[0] = “No picture file is uploaded " | Pass |

## **UTC\_TC 05**

**Test scenario**

: The test case is for check the displaying shopping cart detail. Which the method ShoppingCartDetail has been celled.

**UTC\_TC05**

public ActionResult ShoppingCartDetail(int id)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 05 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 05 | 1 | The displaying shopping cart detail. Which the method shoppingCartDetail has been celled. | 1 | //Arrange  adminController controller = new adminController();  //Act  var viewResult = controller.ShoppingCartDetail(1) as ViewResult;  List<shoppingitemviewmodel> result = (List<shoppingitemviewmodel>)viewResult.Model;  //Assert  Assert.AreEqual(1, result[0].product.productId);  Assert.AreEqual("product1", result[0].product.productName);  Assert.AreEqual("It is good", result[0].product.productDescription);  Assert.AreEqual("pic1.JPG", result[0].product.productPicture);  Assert.AreEqual(1, result[0].quantityShopingProduct);  Assert.AreEqual(1, result[0].shoppingCartId);  Assert.AreEqual(1, result[0].shoppingListId); | Id = 1 | View.base.ViewData.Model = [product = CBSDProject.Models.product, quantityShoppingProduct = 1, shoppingCartId = 1, shoppingListId = 1] | View.base.ViewData.Model = [product = CBSDProject.Models.product, quantityShoppingProduct = 1, shoppingCartId = 1, shoppingListId = 1] | Pass |

## **UTC\_TC 06**

**Test scenario**

: The test case is for check the displaying payment Information. Which the method PaymentInfo has been celled.

**UTC\_TC06**

public ActionResult PaymentInfo(int id)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 06 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 06 | 1 | To test the displaying payment Information. Which the method PaymentInfo has been celled. | 1 | //Arrange  adminController controller = new adminController();  //Act  var viewResult = controller.PaymentInfo(1) as ViewResult;  payment result = (payment)viewResult.Model;  //Assert  Assert.AreEqual(1, result.paymentId);  Assert.AreEqual(1, result.shoppingCartId);  Assert.AreEqual(2, result.paymentType);  Assert.AreEqual(6533332, result.paymentNumber);  Assert.AreEqual(2000.00, result.totalprice);  Assert.AreEqual(null, result.recievedprice);  Assert.AreEqual(0, result.paymentStatus); | Id = 1 | View.base.ViewData.Model = [paymentId = 1, shoppingCartId = 1, paymentType = 2, paymentNumber = 6533332, totalprice = 2000.00, recievedprice = NULL, paymentStatus = 0] | View.base.ViewData.Model = [paymentId = 1, shoppingCartId = 1, paymentType = 2, paymentNumber = 6533332, totalprice = 2000.00, recievedprice = NULL, paymentStatus = 0] | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 06 | 1 | To test the displaying payment Information. Which the method PaymentInfo has been celled. **(Not have a payment**) | 1 | //Arrange  adminController controller = new adminController();  //Act  var result = controller.PaymentInfo(2000) as HttpNotFoundResult;  //Assert  Assert.AreEqual(404, result.StatusCode); | Id = 2000 | HttpNotFound.base.StatusCode = 404 | HttpNotFound.base.StatusCode = 404 | Pass |

## **UTC\_TC 07**

**Test scenario**

: The test case is for check the edit payment Information by user. Which the method PaymentInfo has been celled.

**UTC\_TC07**

public ActionResult PaymentInfo(payment payment)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 07 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 07 | 1 | To test the edit payment Information by user. Which the method PaymentInfo has been celled. | 1 | //Arrange  adminController controller = new adminController();  payment mockPayment = new payment { paymentId = 1, paymentNumber = 12226654, paymentStatus = 0, paymentType = 1, recievedprice = null , totalprice = 2500.00 , shoppingCartId = 1};  //Act  var result = controller.PaymentInfo(mockPayment);  //Assert  result.AssertActionRedirect().ToAction("PaymentList"); | payment = mockpayment | RedirectToAction.RouteValues.Value[0] = “PaymentList” | RedirectToAction.RouteValues.Value[0] = “PaymentList” | Pass |

## **UTC\_TC 08**

**Test scenario**

: The test case is for check working of viewBag and the display list of product Information by user. Which the method Index has been celled.

**UTC\_TC08**

public ActionResult Index(String msg)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 08 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 08 | 1 | To test working of viewBag and the display list of product Information by user. Which the method Index has been celled. | 1 |  | msg = “Has product” | View.base.ViewBag.ViewData.Value[0] = “Has product”, View.base.ViewData.Model[0] = [productId = 1, productName = “product1”, productDescription = (“It is good”), productPrice = 2000.00, productPicture = pic1.JPG] | View.base.ViewBag.ViewData.Value[0] = “Has product”, View.base.ViewData.Model[0] = [productId = 1, productName = “product1”, productDescription = (“It is good”), productPrice = 2000.00, productPicture = pic1.JPG] | Pass |

## **UTC\_TC 09**

**Test scenario**

: The test case is for check the register customer by inputting customer information. Which the method Register has been celled.

**UTC\_TC09**

public ActionResult Register(String customerUsername, String customerPassword, String customerEmail, String customerName, String customerAddress, HttpPostedFileBase file)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 09 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 09 | 1 | To test working of the register customer by inputting customer infomation by user. Which the method Register has been celled. | 1 | //Arrange  homeController controller = new homeController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic2.JPG");  file.Setup(x => x.ContentType).Returns("jpeg");  //Act  var result = controller.Register("cus1", "1234", "tony", "Thailand" , file.Object);  //Assert  result.AssertActionRedirect().ToAction("Index"); | customerUsername = “cus1”, customerPassword = “1234”, customerName = “tony”, customerAddress = “Thailand”, file = pic2.JPG | RedirectToAction.RouteValues.Value[0] = “Index” | RedirectToAction.RouteValues.Value[0] = “Index” | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 09 | 2 | To test working of the register customer by inputting customer infomation by user. Which the method Register has been celled. (**customerUsername format invalid**) | 1 | //Arrange  homeController controller = new homeController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic2.JPG");  file.Setup(x => x.ContentType).Returns("jpeg");  //Act  var result = controller.Register("%%#$%", "1234", "tony", "Thailand", file.Object) as ViewResult;  //Assert  Assert.AreEqual("The username must be only 4-10 alphabet characters", result.ViewBag.RegistrationIssue); | customerUsername = “%%#$%”, customerPassword = “1234”,  customerName = “tony”, customerAddress = “Thailand”, file = pic2.JPG | View.base.ViewBag.ViewData.Value[0] = “The username must be only 4-10 alphabet characters " | View.base.ViewBag.ViewData.Value[0] = “The username must be only 4-10 alphabet characters " | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 09 | 3 | To test working of the register customer by inputting customer infomation by user. Which the method Register has been celled. (**customerPassword format invalid**) | 1 | //Arrange  homeController controller = new homeController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic2.JPG");  file.Setup(x => x.ContentType).Returns("jpeg");  //Act  var result = controller.Register("cus1", "%%", "tony", "Thailand", file.Object) as ViewResult;  //Assert  Assert.AreEqual("The password must be only 4-10 characters", result.ViewBag.RegistrationIssue); | customerUsername = “cus1”, customerPassword = “%%”,  customerName = “tony”, customerAddress = “Thailand”, file = pic2.JPG | View.base.ViewBag.ViewData.Value[0] = “The password must be only 4-10 characters" | View.base.ViewBag.ViewData.Value[0] = “The password must be only 4-10 characters" | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 09 | 4 | To test working of the register customer by inputting customer infomation by user. Which the method Register has been celled. (**customerName format invalid**) | 1 | //Arrange  homeController controller = new homeController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic2.JPG");  file.Setup(x => x.ContentType).Returns("jpeg");  //Act  var result = controller.Register("cus1", "1234", "$%#$$", "Thailand", file.Object) as ViewResult;  //Assert  Assert.AreEqual("The name must be the combination of alphabet and space only", result.ViewBag.RegistrationIssue); | customerUsername = “cus1”, customerPassword = “1234”,  customerName = “$%#$$”, customerAddress = “Thailand”, file = pic2.JPG | View.base.ViewBag.ViewData.Value[0] = “The name must be the combination of alphabet and space only " | View.base.ViewBag.ViewData.Value[0] = “The name must be the combination of alphabet and space only " | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 09 | 5 | To test working of the register customer by inputting customer infomation by user. Which the method Register has been celled. (**file format invalid**) | 1 | //Arrange  homeController controller = new homeController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic2.gif");  file.Setup(x => x.ContentType).Returns("gif");  //Act  var result = controller.Register("cus1", "1234", "tony", "Thailand", file.Object) as ViewResult;  //Assert  Assert.AreEqual("No picture file is uploaded.", result.ViewBag.RegistrationIssue); | customerUsername = “cus1”, customerPassword = “1234”, customerName = “tony”, customerAddress = “Thailand”, file = pic2.gif | View.base.ViewBag.ViewData.Value[0] = “No picture file is uploaded." | View.base.ViewBag.ViewData.Value[0] = “No picture file is uploaded." | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 09 | 6 | To test working of the register customer by inputting customer infomation by user. Which the method Register has been celled. (**not have file** ) | 1 | //Arrange  homeController controller = new homeController();  //Act  var result = controller.Register("cus1", "1234", "tony", "Thailand", null) as ViewResult;  //Assert  Assert.AreEqual("No picture file is uploaded.", result.ViewBag.RegistrationIssue); | customerUsername = “cus1”, customerPassword = “1234”,  customerName = “tony”, customerAddress = “Thailand”, file = None | View.base.ViewBag.ViewData.Value[0] = “No picture file is uploaded." | View.base.ViewBag.ViewData.Value[0] = “No picture file is uploaded." | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 09 | 7 | To test working of the register customer by inputting customer infomation by user. Which the method Register has been celled. (**Existing user in database**) | 1 | //Arrange  homeController controller = new homeController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("cdp1.JPG");  file.Setup(x => x.ContentType).Returns("jpeg");  //Act  var result = controller.Register("milk", "milk", "Suwichak Fungprasertkul", "New York", file.Object) as ViewResult;  //Assert  Assert.AreEqual("Existing User, Please Use Different Username.", result.ViewBag.RegistrationIssue); | customerUsername = “milk”, customerPassword = “milk”,  customerName = “Suwichak Fungprasertkul”, customerAddress = “New York”, file = cdp1.JPG | View.base.ViewBag.ViewData.Value[0] = “Existing User, Please Use Different Username.” | View.base.ViewBag.ViewData.Value[0] = “Existing User, Please Use Different Username.” | Pass |

## **UTC\_TC 10**

**Test scenario**

: The test case is for check the customer log in by verifying value between user input and customer data in the database. Which the method Login has been celled.

**UTC\_TC10**

public ActionResult Login(String customerUsername, String customerPassword)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 10 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 10 | 1 | To test working of the customer log in by verifying value between user input and customer data in the database. Which the method Login has been celled.  **If(Session["customer"] == null)** | 1 | //Arrange  homeController controller = new homeController();  //Act  var result = controller.Login("milk","milk");  //Assert  result.AssertActionRedirect().ToAction("Index"); | customerUsername = “milk”, customerPassword = “milk” | RedirectToAction.RouteValues.Value[0] = “Index” | RedirectToAction.RouteValues.Value[0] = “Index” | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 10 | 2 | To test working of the customer log in by verifying value between user input and customer data in the database. Which the method Login has been celled.  **If(Session["customer"] != null)** | 1 | //Arrange  homeController controller = new homeController();  //Act  var result = controller.Login("milk","milk");  //Assert  result.AssertActionRedirect().ToAction("Index"); | customerUsername = “milk”, customerPassword = “milk” | RedirectToAction.RouteValues.Value[0] = “Index” | RedirectToAction.RouteValues.Value[0] = “Index” | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 10 | 3 | To test working of the customer log in by verifying value between user input and customer data in the database. Which the method Login has been celled. | 1 | //Arrange  homeController controller = new homeController();  //Act  var result = controller.Login("milk", "1234") as ViewResult;  //Assert  Assert.AreEqual("Invalid Username and/or Password", result.ViewBag.LoginIssue); | customerUsername = “milk”, customerPassword = “1234” | View.base.Values.[0] = “Invalid Username and/or Password" | View.base.Values.[0] = “Invalid Username and/or Password" | Pass |

## **UTC\_TC 11**

**Test scenario**

: The test case is for check the customer search product by inputting keyword. Which the method SearchResult has been celled.

**UTC\_TC11**

public ActionResult SearchResult(String keyword)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 11 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 11 | 1 | To test working of the customer search product by inputting keyword. Which the method SearchResult has been celled. | 1 | //Arrange  homeController controller = new homeController();  //Act  var result = controller.SearchResult("product1") as ViewResult;  List<product> productList = (List<product>)result.Model;  //Assert  Assert.AreEqual(1, productList[0].productId);  Assert.AreEqual("product1", productList[0].productName);  Assert.AreEqual("It is good", productList[0].productDescription);  Assert.AreEqual("pic1.JPG", productList[0].productPicture); | keyword = “product1” | View.base.ViewData.Model[0] = [productId = 1, productName = “product1”, productDescription = (“It is good”), productPrice = 2000.00, productPicture = pic1.JPG] | View.base.ViewData.Model[0] = [productId = 1, productName = “product1”, productDescription = (“It is good”), productPrice = 2000.00, productPicture = pic1.JPG] | Pass |

## **UTC\_TC 12**

**Test scenario**

: The test case is for check the edit customer information by inputting customer information. Which the method EditInfo has been celled.

**UTC\_TC12**

public ActionResult EditInfo(String customerId, String customerUsername, String customerPassword, String customerName, String customerAddress, HttpPostedFileBase file)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 12 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 012 | 1 | To test working of the edit customer information by inputting customer information. Which the method EditInfo has been celled. | 1 | //Arrange  homeController controller = new homeController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic2.JPG");  file.Setup(x => x.ContentType).Returns("jpeg");  //Act  var result = controller.EditInfo(1, "cus1", "1234", "tony", "Thailand", file.Object);  //Assert  result.AssertActionRedirect().ToAction("Index"); | customerId = 1, customerUsername = “cus1”, customerPassword = “1234”, customerName = “tony”, customerAddress = “Thailand”, file = pic2.JPG | RedirectToAction.RouteValues.Value[0] = “Index” | RedirectToAction.RouteValues.Value[0] = “Index” | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 012 | 2 | To test working of the edit customer information by inputting customer information. Which the method EditInfo has been celled. (**customerUsername format invalid**) | 1 | //Arrange  homeController controller = new homeController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic2.JPG");  file.Setup(x => x.ContentType).Returns("jpeg");  //Act  var result = controller.EditInfo(1, "%%#$%", "1234", "tony", "Thailand", file.Object) as ViewResult;  //Assert  Assert.AreEqual("The username must be only 4-10 alphabet characters ", result.ViewBag.EditInfoIssue); | customerId = 1,  customerUsername = “%%#$%”, customerPassword = “1234”, customerName = “tony”, customerAddress = “Thailand”, file = pic2.JPG | View.base.ViewBag.ViewData.Value[0] = “The username must be only 4-10 alphabet characters " | View.base.ViewBag.ViewData.Value[0] = “The username must be only 4-10 alphabet characters " | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 012 | 3 | To test working of the edit customer information by inputting customer information. Which the method EditInfo has been celled. (**customerPassword format invalid**) | 1 | //Arrange  homeController controller = new homeController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic2.JPG");  file.Setup(x => x.ContentType).Returns("jpeg");  //Act  var result = controller.EditInfo(1, "cus1", "%%", "tony", "Thailand", file.Object) as ViewResult;  //Assert  Assert.AreEqual("The password must be only 4-10 characters ", result.ViewBag.EditInfoIssue); | customerId = 1,  customerUsername = “cus1”, customerPassword = “%%”, customerName = “tony”, customerAddress = “Thailand”, file = pic2.JPG | View.base.ViewBag.ViewData.Value[0] = “The password must be only 4-10 characters " | View.base.ViewBag.ViewData.Value[0] = “The password must be only 4-10 characters " | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 12 | 4 | To test working of the edit customer information by inputting customer information. Which the method EditInfo has been celled. (**customerName format invalid**) | 1 | //Arrange  homeController controller = new homeController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic2.JPG");  file.Setup(x => x.ContentType).Returns("jpeg");  //Act  var result = controller.EditInfo(1, "cus1", "1234", "$%#$$", "Thailand", file.Object) as ViewResult;  //Assert  Assert.AreEqual("The name must be the combination of alphabet and space only ", result.ViewBag.EditInfoIssue); | customerId = 1,  customerUsername = “cus1”, customerPassword = “1234”, customerName = “$%#$$”, customerAddress = “Thailand”, file = pic2.JPG | View.base.ViewBag.ViewData.Value[0] = “The name must be the combination of alphabet and space only " | View.base.ViewBag.ViewData.Value[0] = “The name must be the combination of alphabet and space only " | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 12 | 5 | To test working of the edit customer information by inputting customer information. Which the method EditInfo has been celled. (**file format invalid**) | 1 | //Arrange  homeController controller = new homeController();  var file = new Mock<HttpPostedFileBase>();  file.Setup(x => x.FileName).Returns("pic2.gif");  file.Setup(x => x.ContentType).Returns("gif");  //Act  var result = controller.EditInfo(1, "cus1", "1234", "tony", "Thailand", file.Object) as ViewResult;  //Assert  Assert.AreEqual("No picture file is uploaded.", result.ViewBag.EditInfoIssue); | customerId = 1,  customerUsername = “cus1”, customerPassword = “1234”, customerName = “tony”, customerAddress = “Thailand”, file = pic2.gif | View.base.ViewBag.ViewData.Value[0] = “No picture file is uploaded." | View.base.ViewBag.ViewData.Value[0] = “No picture file is uploaded." | Pass |

## **UTC\_TC 13**

**Test scenario**

: The test case is for check the add product to cart. Which the method AddToCart has been celled.

**UTC\_TC13**

public ActionResult AddToCart(int productId, int quantityShoppingProduct)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 13 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 013 | 1 | To test working of the add product to cart. Which the method AddToCart has been celled. (**If(Session["customer"] != null)**) | 1 | //Arrange  homeController controller = new homeController();  //Act  var result = controller.AddToCart(1,1);  //Assert  result.AssertActionRedirect().ToAction("ViewShoppingCart"); | productId = 1 , quantityShoppingProduct = 1 | RedirectToAction.RouteValues.Value[0] = “ViewShoppingCart” | RedirectToAction.RouteValues.Value[0] = “ViewShoppingCart” | Pass |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 013 | 2 | To test working of the add product to cart. Which the method AddToCart has been celled. (**If(Session["customer"] == null)**) | 1 | //Arrange  homeController controller = new homeController();  //Act  var result = controller.AddToCart(1,1);  //Assert  result.AssertActionRedirect().ToAction("ViewShoppingCart"); | productId = 1 , quantityShoppingProduct = 1 | RedirectToAction.RouteValues.Value[0] = “Index” | RedirectToAction.RouteValues.Value[0] = “Index” | Pass |

## **UTC\_TC 14**

**Test scenario**

: The test case is for check the edit quantity product to cart. Which the method EditQtyCart has been celled.

**UTC\_TC14**

public ActionResult EditQtyCart(int shoppingListId, int quantityShoppingProduct)

**Prerequisites or Test input** : The Appendix A is available.

**UTC\_TC 14 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 014 | 1 | To test working of the edit quantity product to cart. Which the method EditQtyCart has been celled. | 1 | //Arrange  homeController controller = new homeController();  //Act  var result = controller.EditQtyCart(1,1);  //Assert  result.AssertActionRedirect().ToAction("ViewShoppingCart"); | shoppingListId = 1 , quantityShoppingProduct = 1 | RedirectToAction.RouteValues.Value[0] = “ViewShoppingCart” | RedirectToAction.RouteValues.Value[0] = “ViewShoppingCart” | Pass |

## **UTC\_TC 15**

**Test scenario**

: The test case is for check the remove product in cart. Which the method RemoveShoppingList has been celled.

**UTC\_TC15**

public ActionResult RemoveShoppingList(int shoppingListId)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 15 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 015 | 1 | To test working of the remove product in cart. Which the method RemoveShoppingList has been celled. | 1 | //Arrange  homeController controller = new homeController();  //Act  var result = controller.RemoveShoppingList(1);  //Assert  result.AssertActionRedirect().ToAction("ViewShoppingCart"); | shoppingListId = 1 | RedirectToAction.RouteValues.Value[0] = “ViewShoppingCart” | RedirectToAction.RouteValues.Value[0] = “ViewShoppingCart” | Pass |

## **UTC\_TC 16**

**Test scenario**

: The test case is for check the check out a cart. Which the method Checkout has been celled.

**UTC\_TC16**

public ActionResult Checkout(int shoppingListId, double price)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 16 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 016 | 1 | To test working of the check out a cart. Which the method Checkout has been celled. | 1 | //Arrange  homeController controller = new homeController();  //Act  var viewResult = controller.Checkout(1 , 2000.00) as ViewResult;  checkoutviewmodel result = (checkoutviewmodel)viewResult.Model;    //Assert  Assert.AreEqual(result.shoppingitem[0].product.productId, 1);  Assert.AreEqual(result.shoppingitem[0].product.productName, "product1");  Assert.AreEqual(result.shoppingitem[0].product.productDescription, "It is good");  Assert.AreEqual(result.shoppingitem[0].product.productPrice, 2000.0);  Assert.AreEqual(result.shoppingitem[0].product.productPicture, "pic1.JPG");  Assert.AreEqual(result.shoppingitem[0].product.isDeleted, false);  Assert.AreEqual(result.shoppingitem[0].shoppingCartId, 1);  Assert.AreEqual(result.shoppingitem[0].shoppingListId, 1);  Assert.AreEqual(result.shoppingitem[0].quantityShopingProduct, 1);  Assert.AreEqual(result.payment.paymentId, 1);  Assert.AreEqual(result.payment.paymentNumber, 6533332);  Assert.AreEqual(result.payment.paymentStatus, 0 );  Assert.AreEqual(result.payment.paymentType, 2);  Assert.AreEqual(result.payment.recievedprice, null);  Assert.AreEqual(result.payment.totalprice, 2000.00); | shoppingListId = 1, price = 2000.00 | View.base.ViewData.Model = [List<shoppingitemviewmodel>, payment] | View.base.ViewData.Model = [List<shoppingitemviewmodel>, payment] | Pass |

## **UTC\_TC 17**

**Test scenario**

: The test case is for check the check out confirm a cart. Which the method ConfirmCheckout has been celled.

**UTC\_TC17**

public ActionResult ConfirmCheckout(payment Payment)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 17 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 017 | 1 | To test working of the check out confirm a cart. Which the method ConfirmCheckout has been celled. | 1 | //Arrange  homeController controller = new homeController();  payment mockPayment = new payment { paymentNumber = 1, paymentStatus = 1, paymentType = 1, recievedprice = null, totalprice = 2000.00, shoppingCartId = 1, paymentId = 1 };  //Act  var result = controller.ConfirmCheckout(mockPayment);  //Assert  result.AssertActionRedirect().ToAction("Index"); | payment = mockpayment | View.base.ViewBag.ViewData.Value[0] = “check out successful "  ,  RedirectToAction.RouteValues.Value[0] = “Index” | View.base.ViewBag.ViewData.Value[0] = “check out successful "  , RedirectToAction.RouteValues.Value[0] = “Index” | Pass |

## **UTC\_TC 18**

**Test scenario**

: The test case is for check the customer display shopping lists in owner cart. Which the method ShoppingCartDetail has been celled.

**UTC\_TC18**

public ActionResult ShoppingCartDetail(int id)

**Prerequisites or Test input**

: The Appendix A is available.

**UTC\_TC 18 relationship table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | case | Test case description | Test Step# | Execution test steps | Test Data | Expect result | Actual result | Pass/Fail |
| UTC\_TC 017 | 1 | To test working of the customer display shopping lists in owner cart. Which the method ShoppingCartDetail has been celled. (**Has exist shopping lists data in cart**) | 1 | //Arrange  homeController controller = new homeController();  //Act  var viewResult = controller.ShoppingCartDetail(1) as ViewResult;  List<shoppingitemviewmodel> result = (List<shoppingitemviewmodel>)viewResult.Model;  //Assert  Assert.AreEqual(result[0].product.productId, 1);  Assert.AreEqual(result[0].product.productName, "product1");  Assert.AreEqual(result[0].product.productDescription, "It is good");  Assert.AreEqual(result[0].product.productPrice, 2000.0);  Assert.AreEqual(result[0].product.productPicture, "pic1.JPG");  Assert.AreEqual(result[0].product.isDeleted, false);  Assert.AreEqual(result[0].shoppingCartId, 1);  Assert.AreEqual(result[0].shoppingListId, 1);  Assert.AreEqual(result[0].quantityShopingProduct, 1); | id = 1 | View.base.ViewData.Model[0] = [product = CBSDProject.Models.product, quantityShoppingProduct = 1, shoppingCartId = 1, shoppingListId = 1] | View.base.ViewData.Model[0] = [product = CBSDProject.Models.product, quantityShoppingProduct = 1, shoppingCartId = 1, shoppingListId = 1] | Pass |