# **Integration testing record**

# **Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Revision Description** |
| V1.0 | February 28, 2014 | Mr. Suwichak Fungprasertkul | Create Integration test record documentation |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Reviewer

|  |  |  |
| --- | --- | --- |
| **Name** | **Version Approved** | **Date** |
| Mr.Tanadol Parnong  Ms.Chonticha Prachpreang  Mr. Suwichak Fungprasertkul | V1.0 | February 28, 2014 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Table of Contents**

[**Integration testing** 1](#_Toc381534765)

[**Revision history** 1](#_Toc381534766)

[Reviewer 1](#_Toc381534767)

[**References** 3](#_Toc381534768)

[**Introduction** 3](#_Toc381534769)

[**Responsibility** 3](#_Toc381534770)

[**Test Item** 3](#_Toc381534771)

[**Test tools** 3](#_Toc381534772)

[**Features to be tested** 3](#_Toc381534773)

[Database for testing 3](#_Toc381534774)

[Test case of Integration test (IT) 5](#_Toc381534775)

[**IT\_TC 01** 5](#_Toc381534776)

[**IT\_TC 02** 6](#_Toc381534777)

[**IT\_TC 03** 7](#_Toc381534778)

[**IT\_TC 04** 8](#_Toc381534779)

[**IT\_TC 05** 9](#_Toc381534780)

[**IT\_TC 06** 10](#_Toc381534781)

[**IT\_TC 07** 11](#_Toc381534782)

[**IT\_TC 08** 12](#_Toc381534783)

[**IT\_TC 09** 13](#_Toc381534784)

# **References**

High Level Design Document (Sequence Diagram)

# **Introduction**

Integration Test is the one of the important test that should be considered of. Because in the software development process, we also should consider how well the related methods behave when it has been acted together. Especially for Object Oriented Programming software development.

# **Responsibility**

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

# **Test Item**

* CBSD Project

# **Test tools**

* Microsoft visual studio 2013

# **Test Environment requirement**

* Windows 7 SP1 or Windows 8
* MySQL 5.6 or later

# **Features to be tested**

* All interface to sending the data between the controller and models.

# Database for testing

: Integration test need to request data in database

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 |

shoppinlist

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

# Test case of Integration test (IT)

## **IT\_TC 01**

**Test scenario**

: The controller manipulates data from database via model entities called admin model.

**Prerequisites or Test input**

: The testing is available data at database.

**IT\_TC 01 relationship table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Test case description | Execution test steps | Expect result | Actual result |
| IT\_TC 01 | The controller manipulates data from database via model entities called admin model. | //Arrange  IEShoppingRepository db = new eshoppingEntities();  admin tempa = new admin() { adminUsername = "milk", adminPassword = "bella" };  int size = 0;  //Act-Create Part  db.admins.Add(tempa);  db.SaveChanges();  List<admin> list = db.admins.ToList<admin>();  size = list.Count;  admin result = list[list.Count - 1];  //Assert-Create Part  Assert.IsTrue(result.adminUsername == "milk");  Assert.IsTrue(result.adminPassword == "bella");  //Act-Delete part  db.admins.Remove(result);  list = db.admins.ToList<admin>();  //Assert-Delete Part  Assert.AreEqual(list.Count, size); | Pass | Pass |

## **IT\_TC 02**

**Test scenario**

: The controller manipulates data from database via model entities called customer model.

**Prerequisites or Test input**

: The testing is available data at database.

**IT\_TC 02 relationship table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Test case description | Execution test steps | Expect result | Actual result |
| IT\_TC 01 | The controller manipulates data from database via model entities called customer model. | //Arrange  IEShoppingRepository db = new eshoppingEntities();  customer tempc = new customer() { customerUsername = "milk", customerPassword = "bella", customerName = "Suwichak Fungprasertkul", customerAddress = "Twilight Zone", customerPicture = "jacob.jpg"};  int size = 0;  //Act-Create Part  db.customers.Add(tempc);  db.SaveChanges();  List<customer> list = db.customers.ToList<customer>();  size = list.Count;  customer result = list[list.Count - 1];  //Assert-Create Part  Assert.IsTrue(result.customerUsername == "milk");  Assert.IsTrue(result.customerPassword == "bella");  Assert.IsTrue(result.customerName == "Suwichak Fungprasertkul");  Assert.IsTrue(result.customerAddress == "Twilight Zone");  Assert.IsTrue(result.customerPicture == "jacob.jpg");  //Act-Delete part  db.customers.Remove(result);  list = db.customers.ToList<customer>();  //Assert-Delete Part  Assert.AreEqual(list.Count, size); | Pass | Pass |

## **IT\_TC 03**

**Test scenario**

: The controller manipulates data from database via model entities called payment model.

**Prerequisites or Test input**

: The testing is available data at database.

**IT\_TC 03 relationship table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Test case description | Execution test steps | Expect result | Actual result |
| IT\_TC 03 | The controller manipulates data from database via model entities called payment model. | //Arrange  IEShoppingRepository db = new eshoppingEntities();  payment tempp = new payment() { paymentNumber = 4115646, paymentStatus = 0, paymentType = 2, recievedprice = 2000.00, shoppingCartId = 1, totalprice = 1000.00 };  int size = 0;  //Act-Create Part  db.payments.Add(tempp);  db.SaveChanges();  List<payment> list = db.payments.ToList<payment>();  size = list.Count;  payment result = list[list.Count - 1];  //Assert-Create Part  Assert.IsTrue(result.paymentNumber == 4115646);  Assert.IsTrue(result.paymentStatus == 0);  Assert.IsTrue(result.paymentType == 2);  Assert.IsTrue(result.recievedprice == 2000.00);  Assert.IsTrue(result.shoppingCartId == 1);  Assert.IsTrue(result.totalprice == 1000.00);  //Act-Delete part  db.payments.Remove(result);  list = db.payments.ToList<payment>();  //Assert-Delete Part  Assert.AreEqual(list.Count, size); | Pass | Pass |

## **IT\_TC 04**

**Test scenario**

: The controller manipulates data from database via model entities called product model.

**Prerequisites or Test input**

: The testing is available data at database.

**IT\_TC 04 relationship table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Test case description | Execution test steps | Expect result | Actual result |
| IT\_TC 04 | The controller manipulates data from database via model entities called product model. | //Arrange  IEShoppingRepository db = new eshoppingEntities();  product tempp = new product() { productDescription = "This is product descirption", productName = "Product A", productPicture = "dummyProduct.jpg", productPrice = 0.00, isDeleted = false };  int size = 0;  //Act-Create Part  db.products.Add(tempp);  db.SaveChanges();  List<product> productlist = db.products.ToList<product>();  size = productlist.Count;  product result = productlist[productlist.Count - 1];    //Assert-Create Part  Assert.IsTrue(result.productName == "Product A");  Assert.IsTrue(result.productDescription == "This is product descirption");  Assert.IsTrue(result.productPrice == 0.00);  //Act-Delete part  db.products.Remove(result);  productlist = db.products.ToList<product>();  //Assert-Delete Part  Assert.AreEqual(productlist.Count,size); | Pass | Pass |

## **IT\_TC 05**

**Test scenario**

: The controller manipulates data from database via model entities called shoppingcart model.

**Prerequisites or Test input**

: The testing is available data at database.

**IT\_TC 05 relationship table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Test case description | Execution test steps | Expect result | Actual result |
| IT\_TC 05 | The controller manipulates data from database via model entities called shoppingcart model. | //Arrange  IEShoppingRepository db = new eshoppingEntities();  System.DateTime timestamp = DateTime.Now;  shoppingcart temps = new shoppingcart() { customerId = 1, status = 0, created\_date = timestamp, checkedout\_date = null };  int size = 0;  //Act-Create Part  db.shoppingcarts.Add(temps);  db.SaveChanges();  List<shoppingcart> list = db.shoppingcarts.ToList<shoppingcart>();  size = list.Count;  shoppingcart result = list[list.Count - 1];  //Assert-Create Part  Assert.IsTrue(result.customerId == 1);  Assert.IsTrue(result.status == 0);  Assert.IsTrue(result.created\_date == timestamp);  Assert.IsTrue(result.checkedout\_date == null);  //Act-Delete part  db.shoppingcarts.Remove(result);  list = db.shoppingcarts.ToList<shoppingcart>();  //Assert-Delete Part  Assert.AreEqual(list.Count, size); | Pass | Pass |

## **IT\_TC 06**

**Test scenario**

: The controller manipulates data from database via model entities called shoppinglist model.

**Prerequisites or Test input**

: The testing is available data at database.

**IT\_TC 06 relationship table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Test case description | Execution test steps | Expect result | Actual result |
| IT\_TC 06 | The controller manipulates data from database via model entities called shoppinglist model. | //Arrange  IEShoppingRepository db = new eshoppingEntities();  shoppinglist temps = new shoppinglist() { shoppingCartId = 1, quantityShopingProduct = 1 , productId = 1, IsDeleted = false};  int size = 0;  //Act-Create Part  db.shoppinglists.Add(temps);  db.SaveChanges();  List<shoppinglist> list = db.shoppinglists.ToList<shoppinglist>();  size = list.Count;  shoppinglist result = list[list.Count - 1];  //Assert-Create Part  Assert.IsTrue(result.shoppingCartId == 1);  Assert.IsTrue(result.quantityShopingProduct == 1);  Assert.IsTrue(result.productId == 1);  Assert.IsTrue(result.IsDeleted == false);  //Act-Delete part  db.shoppinglists.Remove(result);  list = db.shoppinglists.ToList<shoppinglist>();  //Assert-Delete Part  Assert.AreEqual(list.Count, size); | Pass | Pass |

## **IT\_TC 07**

**Test scenario**

: The controller display data from database via model entities called checkoutviewmodel model.

**Prerequisites or Test input**

: The testing is available data at database.

**IT\_TC 07 relationship table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Test case description | Execution test steps | Expect result | Actual result |
| IT\_TC 07 | The controller display data from database via model entities called checkoutviewmodel model. | //Arrange  IEShoppingRepository db = new eshoppingEntities();  product tempp = new product() { productDescription = "This is product descirption", productName = "Product A", productPicture = "dummyProduct.jpg", productPrice = 0.00, isDeleted = false };  payment temppy = new payment() { paymentNumber = 4115646, paymentStatus = 0, paymentType = 2, recievedprice = 2000.00, shoppingCartId = 1, totalprice = 1000.00 };    //Act-Display Part  db.products.Add(tempp);  db.SaveChanges();  db.payments.Add(temppy);  db.SaveChanges();  List<product> productlist = db.products.ToList<product>();  product presult = productlist[productlist.Count - 1];  shoppingitemviewmodel sivm = new shoppingitemviewmodel() { shoppingListId = 1, shoppingCartId = 1, product = presult, quantityShopingProduct = 1};  List<shoppingitemviewmodel> listSIVM = new List<shoppingitemviewmodel>();  listSIVM.Add(sivm);  List<payment> list = db.payments.ToList<payment>();  payment pyresult = list[list.Count - 1];  checkoutviewmodel result = new checkoutviewmodel() { shoppingitem = listSIVM, payment = pyresult };    //Assert-Display Part  Assert.AreEqual(result.shoppingitem[0].product.productName, "Product A");  Assert.AreEqual(result.shoppingitem[0].product.productDescription, "This is product descirption");  Assert.AreEqual(result.shoppingitem[0].product.productPrice, 0.0);  Assert.AreEqual(result.shoppingitem[0].product.productPicture, "dummyProduct.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.paymentNumber, 4115646);  Assert.AreEqual(result.payment.paymentStatus, 0 );  Assert.AreEqual(result.payment.paymentType, 2);  Assert.AreEqual(result.payment.recievedprice, 2000.00);  Assert.AreEqual(result.payment.shoppingCartId, 1);  Assert.AreEqual(result.payment.totalprice, 1000.00); | Pass | Pass |

## **IT\_TC 08**

**Test scenario**

: The controller display data from database via model entities called shoppingitemviewmodel model.

**Prerequisites or Test input**

: The testing is available data at database.

**IT\_TC 08 relationship table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Test case description | Execution test steps | Expect result | Actual result |
| IT\_TC 08 | The controller display data from database via model entities called shoppingitemviewmodel model. | //Arrange  IEShoppingRepository db = new eshoppingEntities();  product tempp = new product() { productDescription = "This is product descirption", productName = "Product A", productPicture = "dummyProduct.jpg", productPrice = 0.00, isDeleted = false };    //Act-Display Part  db.products.Add(tempp);  db.SaveChanges();    List<product> productlist = db.products.ToList<product>();  product presult = productlist[productlist.Count - 1];  shoppingitemviewmodel result = new shoppingitemviewmodel() { shoppingListId = 1, shoppingCartId = 1, product = presult, quantityShopingProduct = 1 };    //Assert-Display Part  Assert.AreEqual(result.product.productName, "Product A");  Assert.AreEqual(result.product.productDescription, "This is product descirption");  Assert.AreEqual(result.product.productPrice, 0.0);  Assert.AreEqual(result.product.productPicture, "dummyProduct.jpg");  Assert.AreEqual(result.product.isDeleted, false);  Assert.AreEqual(result.shoppingCartId, 1);  Assert.AreEqual(result.shoppingListId, 1);  Assert.AreEqual(result.quantityShopingProduct, 1); | Pass | Pass |

## **IT\_TC 09**

**Test scenario**

: The controller display data from database via model entities called paymenthistory model.

**Prerequisites or Test input**

: The testing is available data at database.

**IT\_TC 09 relationship table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Test case description | Execution test steps | Expect result | Actual result |
| IT\_TC 09 | The controller display data from database via model entities called paymenthistory model. | //Arrange  IEShoppingRepository db = new eshoppingEntities();  System.DateTime timestamp = DateTime.Now;  shoppingcart temps = new shoppingcart() { customerId = 1, status = 0, created\_date = timestamp, checkedout\_date = null };  payment temppy = new payment() { paymentNumber = 4115646, paymentStatus = 0, paymentType = 2, recievedprice = 2000.00, shoppingCartId = 1, totalprice = 1000.00 };  //Act-Display Part    db.payments.Add(temppy);  db.SaveChanges();  db.shoppingcarts.Add(temps);  db.SaveChanges();  List<payment> plist = db.payments.ToList<payment>();  payment pyresult = plist[plist.Count - 1];  List<shoppingcart> slist = db.shoppingcarts.ToList<shoppingcart>();  shoppingcart sresult = slist[slist.Count - 1];  paymenthistory result = new paymenthistory() { payment = pyresult, shoppingcart = sresult };  //Assert-Display Part  Assert.IsTrue(result.shoppingcart.customerId == 1);  Assert.IsTrue(result.shoppingcart.status == 0);  Assert.IsTrue(result.shoppingcart.created\_date == timestamp);  Assert.IsTrue(result.shoppingcart.checkedout\_date == null);  Assert.AreEqual(result.payment.paymentNumber, 4115646);  Assert.AreEqual(result.payment.paymentStatus, 0);  Assert.AreEqual(result.payment.paymentType, 2);  Assert.AreEqual(result.payment.recievedprice, 2000.00);  Assert.AreEqual(result.payment.shoppingCartId, 1);  Assert.AreEqual(result.payment.totalprice, 1000.00); | Pass | Pass |