Test Plan

Diabetes Care

By

Jirayu Chinpongsuwan 552115011

Department of Software Engineering
College of Arts, Media and Technology
Chiang Mai University

Project Advisor
Dr. Prompong Sugunnasil

Document Name	Diabetes Care -Test Plan-V.0.2.docx	Owner	Jirayu	Page	1 / 83
Document Type	Test Plan	Release Date	July30,2016	Print Date	July30,2016

Document History

Document Name	Version	Status	Date	Viewable	Reviewer	Responsible
Document						
diabetes –Test plan-V.0.1.docx	Add : Chapter One | Introduction Add: Chapter Two | Test Procedure Add: Chapter Three | Software Testing	Draft	February 27, 2016	JC, PS	JC, PS	JC, PS
diabetes –Test plan-V.0.1.docx	Update: Chapter Three | Software Testing Add: SystemTest	Draft	February 29, 2016	JC, PS	JC, PS	JC, PS
diabetes –Test plan-V.0.1.docx	Update: System test Add: Unit test	Draft	July 10,2016	JC, PS	JC, PS	JC, PS
diabetes –Test plan-V.0.1.docx	Update: System test Update: Unit test	Draft	July 23,2016	JC, PS	JC, PS	JC, PS
diabetes –Test plan-V.0.1.docx	Update: Unit test	Draft	July 25,2016	JC, PS	JC, PS	JC, PS
diabetes –Test plan-V.0.1.docx	Update: Unit test	Release	July 30,2016	JC, PS	JC, PS	JC, PS

*PS: Dr. Prompong Sugunnasil
*JC: Mr. Jirayu Chinpongsuwan

Table Content

Chapter One | Introduction

A Software Test Plan is a document describing the testing scope and activities. It is the basis for formally testing any software or product in a project. For this Test Plan Document is provide for testing the software product cover the features in phase of implementation the progress report II.

1.1 Document purpose

Test plan document is a document used to determine and explain testing process of Diabetes care web application. The process of testing is purpose to find the software error that may occur during development process to make sure that the error has been reduce. Therefore, this can guarantee the quality of the software before delivery that result in the application will be the most efficiency. The test case will design base on Software Requirement Specification document and Software Design Document.

1.2 Scope

The Test Plan document covers recording and result of testing for each function in Diabetes web application to find the error or failure of the system. Also the test plan is made to measure the user requirements and system requirement. For the content of this document is consist of part of testing schedule, testing process, and data of testing to estimate an expect result of test before can get the actual result in the test record.

1.3 Definitions, Acronyms, and Abbreviation

1.3.1 Key Definitions

Diabetes	"Name of project"
System	"Computer programs, procedure and associated documentation and data pertain to the operation of computer system" [IEEE90]

Requirement	"The period of time in the software life cycle during which the requirement for a software product defines and document" [IEEE90]
Test	An activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component. [IEEE90]
Unit testing	Testing of an individual hardware or software units or groups of related units. [IEEE90]

1.3.2 Key Acronyms and Abbreviation

TC	Test case
ID	Identify number
UT	Unit test
ST	System test

Chapter Two | Test Procedure

2.1 Test Strategy

Tester test Integration and Developer test unit level Step of Integration Test following:

1. Design test case
2. Prepare testing data
3. Determine expected result
4. Perform testing
5. Result of testing is to be record
6. Result and test files will be place in the project library

2.2 Pass/Fail Criteria

Test record of this Software Testing Document will device to two parts

1. Actual output	Actual output is a real output of the application from testing.
2. Result of Testing	Result of Testing is a result of the testing process there are two kinds is:
Pass	Pass is a result that actual output has data as same as expect output. When the result of testing is pass tester will record data.
Fail	Fail is a result that data of actual output difference from expect output. When the result of testing is fail tester will comment error and send record to developer for edit or repair that error then send new application to tester for retested.

2.3 Test Environment

Hardware

Dell inspiron 5459

- Intel core i5
- RAM 4GB 64-bit, Operating System, HDD 1 TB

Chapter Three | Software Testing

3.1 Objective of Testing

This testing has objective for make all functions are consistency and reliability with a highest quality by check correctness of Diabetes web application. To make sure that the result of testing has record are correctly, every test have to follow test from this part of Test Plan document.

3.2 Scope of Testing

3.2.1 Unit Test

Use Case ID	Use case name
UT01	AccountController: AddTodayFood()
UT02	AccountController: addTodayfood()
UT03	AccountController: RemoveTodayFood()
UT04	AccountController: HealthPlan()
UT05	AccountController: AddGlycemic()
UT06	AccountController: addUserGlycemic()
UT07	AccountController: AddFood()
UT08	AccountController: addUSerFood()
UT09	AccountController: DeleteFood()
UT10	AccountController: AddMedicine()
UT11	AccountController: addMedicineData()
UT12	AccountController: DeleteMedicine()
UT13	AccountController: EditUserProfile()
UT14	AccountController: Register()
UT15	AccountController: Login()
UT16	AccountController:LogOff()
UT17	FoodController:Create()
UT18	FoodController:Edit()
UT19	FoodController:Delete()
UT20	ExercisesController:Create()
UT21	ExercisesController: Edit()
UT22	ExercisesController:Delete()
UT23	ManageController ChangePassword()

Table 1: Black Box testing technique (UT01-UT23)

3.2.2 System Test

Use Case ID	Use case name
ST01	Login to the system.
ST02	Logout to the system.
ST03	Nurse can registration patient to Web Application.
ST04	Patient can view information of his on profile page.
ST05	Patient can update info on his profile page.
ST06	Patient can change password.
ST07	Admin can add user to system.
ST08	Admin can edit information of user.
ST09	Admin can delete user from system
ST10	Nutritionists can add food to the database.
ST11	Nutritionists can edit food on the database.
ST12	Nutritionists can delete food from the database
ST13	Nurse/Doctor can add food to the database.
ST14	Nurse/Doctor can edit food on the database.
ST15	Nurse/Doctor can delete food from the database
ST16	Patient can select the foods from database.
ST17	Patient can select the activity from database.
ST18	Patient can view body mass value that calculated.
ST19	Patient can record blood sugar.
ST20	Patient can view statistic graph of blood sugar.
ST21	Patient can get recommend about blood sugar from system.
ST22	Patient can select food.
ST23	Patient can submit and view about analyze result.

Table 2: Black Box testing technique (ST01-ST23)

3.3 System Test

Black Box technique (ST01-ST23)

Use Case ID	Use case name
ST01	Login to the system.
ST02	Logout to the system.
ST03	Nurse can registration patient to Web Application.
ST04	Patient can view information of his on profile page.
ST05	Patient can update info on his profile page.
ST06	Patient can change password.
ST07	Admin can add user to system.
ST08	Admin can edit information of user.
ST09	Admin can delete user from system
ST10	Nutritionists can add food to the database.
ST11	Nutritionists can edit food on the database.
ST12	Nutritionists can delete food from the database
ST13	Nurse/Doctor can add food to the database.
ST14	Nurse/Doctor can edit food on the database.
ST15	Nurse/Doctor can delete food from the database
ST16	Patient can select the activity from database.
ST17	Patient can select the foods from database.
ST18	Patient can view body mass value that calculated.
ST19	Patient can record blood sugar.
ST20	Patient can view statistic graph of blood sugar.
ST21	Patient can get recommend about blood sugar from system.
ST22	Patient can select food.
ST23	Patient can submit and view about analyze result.

Table 3: Black Box testing technique (ST01-ST23)

3.3.1 Feature#1 Authentication system.

ST01: Login to the system.

Description	Test for **UC01**: Login to the system.
Precondition	Test data must be prepared.
Prerequisites or Test input	**Table: AspUser** Username: atomsuperza Password: 12345678
Test Script	1. User go to login page. 2. User inputs information on login form. 3. User clicks “Login” button.

Test Case:

Case	Description	Input	Expect Output
1	Test login with all correct information.	Username = “atomsuperza” Password = 12345678”	The system redirect to page “Account/Index”.
2	Test login when input invalid email or password.	Username=”atom12” Password = “12345678”	The system display error message " Invalid login attempt.." 

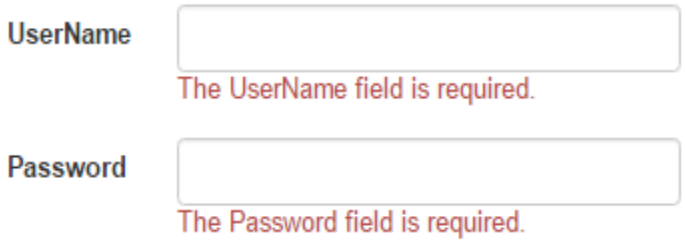
3	Test login with blank information.	Username = “” Password = “”	The system display error message “The UserName field is required.”, “The Password field is required.” 

Table 4: Test Case of **ST02:** Login to the system.

ST02: Logout the system.

Description	Test for **UC02**: Logout the system.
Precondition	Test data must be prepared.
Prerequisites or Test input	-
Test Script	1. User clicks “Logout” button.

Test Case:

Case	Description	Input	Expect Output
1	Test for logout the system.	Click “Logout” button on navigation.	The system redirect to login page.

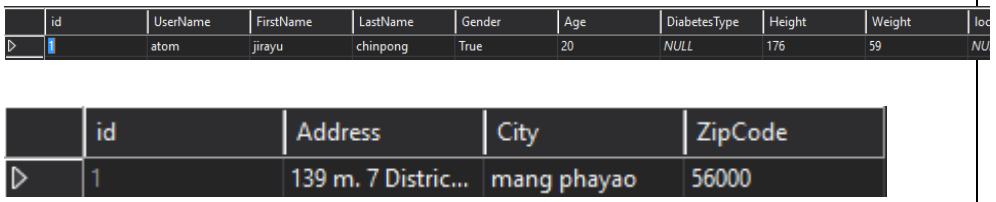
Table 5: Test Case of **ST03:** Logout the system.

3.3.2 Feature#2: Account management.

ST03: Register to the system.

Description	Test for **UC03**: Nurse can registration patient to Web Application.
Precondition	Test data must be prepared.
Prerequisites or Test input	-
Test Script	1. User go to register page. 2. User inputs information on register form. 3. User clicks “Submit” button.

Test Case:

Case	Description	Input	Expect Output
1	Test register with all correct information.	Username = “atom” Email = “se552115011@vr.camt.info” Password = “12345678” Re-password = “12345678” First Name= “Jirayu” Last Name= “Chinpong” Height = “176” Weight= “59” Age = “20” Address = “139 m. 7 District.	The system save the user information into database and redirect to progress page. **Result:**  id UserName FirstName LastName Gender Age DiabetesType Height Weight 1 atom jirayu chinpong True 20 NULL 176 59 id Address City ZipCode 1 139 m. 7 Distric... mang phayao 56000

		tom” City= “Phayao” Zip Code = “56000”	
2	Test register when input existed email.	Email = “aaaaaa”	The system display error message “The Email field is not a valid e-mail address.”
3	Test register when password and re-password is not match.	Password = “12345678” Re-password = “12345677”	The system display error message “The password and confirmation password do not match”
4	Test register with blank information.	Username = “” Email = “” Password = “” Re-password = “” First Name= “” Last Name= “” Height = “” Weight= “” Age = “” Address = “ ” City= “ ” Zip Code = “ ”	The system display error message “field is required.”

Table 6: Test Case of **ST01:** Register to the system.

ST04: Patient can view information of his on profile page.

Description	Test for **UC04**: Patient can view information of his on profile page.
Precondition	Test data must be prepared.
Prerequisites or Test input	**Table: AspNet** First name: Jirayu / qwe Last name: Chinpong / wqr Height: 176 / 123 Weight: 56 / 213 Age: 20 / 123 Address: 139 m. 7 District. tom / 139 m. 7 District. tom City: phayao / phayao Zipcode: 2500 / 2500
Test Script	1. Patient go to a profile page

Test Case:

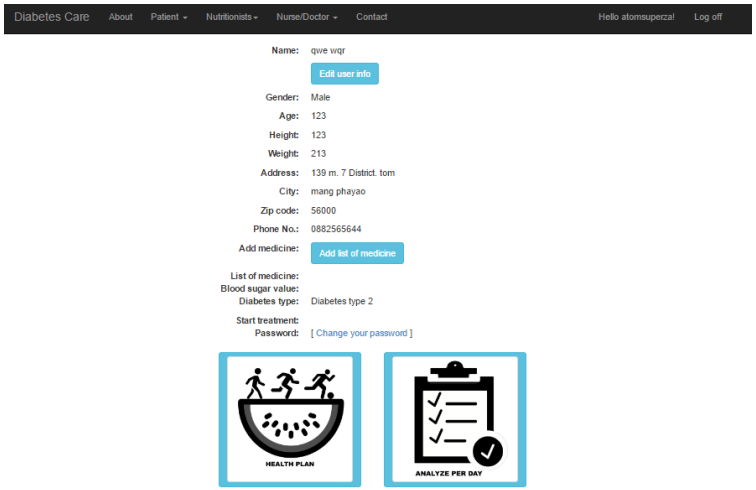
Case	Description	Input	Expect Output
1	Test thesystem shall display user profile.	First Name= “Jirayu” Last Name= “Chinpong” Height = “176” Weight= “59” Age = “20” Address = “139 m. 7 District. tom” City= “Phayao” Zip Code = “56000”	The system shall redirect to “Manage” page and display pertinent information. 

Table 7: Test Case of **ST05:** Custom notification.

ST05: Patient can update info on his profile page.

Description	Test for **UC05**: Patient can update info on his profile page.
Precondition	Test data must be prepared.
Prerequisites or Test input	The test data are username, email, password, re-password, first name, last name, height, weight, age, gender, address, diabetes type.
Test Script	1. Patient click username that show on layout page. 2. Patient click update info. 3. Patient input new info. 4. Patient click “submit”.

Test Case:

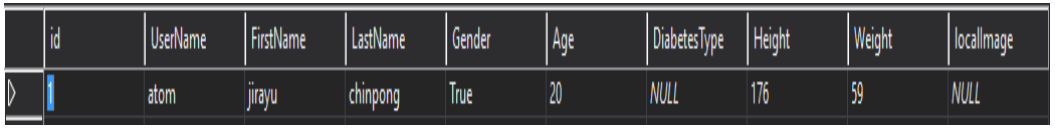
Case	Description	Input	Expect Output
1	Test for successfully edit personal information.	First Name= “Jirayuu” Last Name= “Chinpongsuwan” Height = “177” Weight= “60” Age = “20”	The system save the user information into database. **Result:** 
2	Test register with blank information.	Username = “” Email = “” Password = “” Re-password = “”	The system display error message “field is required.”
3	Test input email in wrong format.	Email = “a”	The system display error message “The Email field is not a valid e-mail address.”

Table 8: Test Case of **ST05:** Patient can update info on his profile page.

ST06: Patient can change password.

Description	Test for **UC06:** Patient can change password.
Precondition	Test data must be prepared.
Prerequisites or Test input	**Table: AspUser** password | 12345678
Test Script	1. Patient go profile page 2. Patient click “change password” 3. Patient input new password. 4. Patient clicks “submit” button.

Test Case:

Case	Description	Input	Expect Output
1	Test for successfully information.	Current password = “12345678” New password = “1234567789” Confirm password= “123456789”	The system get new password and save change in the database
2	Test change password when input invalid current password.	Current password = “12345555”	The system display error message " current password invalid."
3	Test change password when input New password and Confirm password not the same.	New password = “12345678” Confirm password= “22345679”	The system display error message “confirm password not match”
4	Test create plan with blank information.	Current password = “ ” New password = “ ”	The system display error message “field is required.”

Table 9: Test Case of **ST06:** Patient can change password.

ST07: Admin can add user to system.

Description	Test for **UC07:** Admin can add nutritionists to system.
Precondition	Test data must be prepared. Username= “Nutritionists” Email “Nutritionists@n.com” Password= “nutritionists” Roles “Nutritionists”
Prerequisites or Test input	**Table: AspUser** Username | Nutritionists | Doctor Email | Nutritionists@n.com | Doctor@d.com Password | nutritionists | 12345678 Role | Nutritionists | Doctor
Test Script	1. Admin login to system by username and password for admin 2. Admin click Administration on layout. 3. Admin click “Account management”. 4. Admin click “Create New user account”. 5. Admin select Roles “Nutrtionists”.

Test Case:

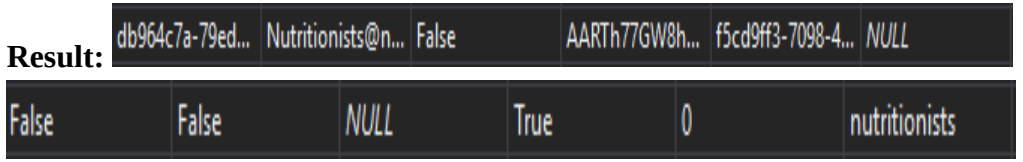
Case	Description	Input	Expect Output
1	Test for successfully create plan information.	Username= "Nutritionists" Email "Nutritionists@n.com" Password= "nutritionists" Roles "Nutritionists"	The system shall record the data on the database. **Result:** 
2	Test create plan with blank information.	Username= "" Email ="" Password= "" Roles ""	The system display error message "field is required."
3	Test input password 5 digits.	Password " nutri"	The system shall display error message "Failed to create the user.".
4	Test input email wrong format.	Email = "Nurtiton.com"	The system shall display error message "Email wrong format".

Table 10: Test Case of **ST07:** Admin can add nutritionists to system.

ST08: Admin can edit information of user.

<table>
<tr><td>Description</td><td>Test for UC07: Admin can edit information of nutritionists.</td></tr>
<tr><td>Precondition</td><td>Test data must be prepared.
Username= “Nutritionists2”
Role= “nutrition”</td></tr>
<tr><td>Prerequisites or Test input</td><td>Table: AspUser
<table>
<tr><td>Username</td><td>Nutritionists</td><td>Doctor</td></tr>
<tr><td>Email</td><td>Nutritionists@n.com</td><td>Doctor@d.com</td></tr>
<tr><td>Password</td><td>nutritionists</td><td>12345678</td></tr>
<tr><td>Role</td><td>Nutritionists</td><td>Doctor</td></tr>
</table></td></tr>
<tr><td>Test Script</td><td>1. Admin go to manage page
2. Admin click nutritionists.
3. Admin click “edit”.
4. Admin input new username, password, name
5. Admin click “submit”</td></tr>
</table>

Test Case:

Case	Description	Input	Expect Output
1	Test for the admin can edit the user roles .	Username= “Nutritionists2” Role= “doctor”	The system shall record the data on the database. **Result:** Username= “Nutritionists2” Role= “doctor”
2	When the admin dose not input new info of user account.	Username “ ” Password “ ” Role= “”	The system display error message “field is required.”

Table 11: Test Case of **ST08:** Admin can edit information of nutritionists.

ST09: Admin can delete user from system

<table>
<tr><td>Description</td><td>Test for UC07: Admin can delete nutritionists from system</td></tr>
<tr><td>Precondition</td><td>Test data must be prepared.{ Username = nutritionists}</td></tr>
<tr><td>Prerequisites or Test input</td><td>Table: AspUser
<table>
<tr><td>Username</td><td>Nutritionists</td><td>Doctor</td></tr>
<tr><td>Email</td><td>Nutritionists@n.com</td><td>Doctor@d.com</td></tr>
<tr><td>Password</td><td>nutritionists</td><td>12345678</td></tr>
<tr><td>Role</td><td>Nutritionists</td><td>Doctor</td></tr>
</table></td></tr>
<tr><td>Test Script</td><td>1. Admin go to manage page
2. Admin click nutritionists.
3. Admin select nutritionists.
4. Admin click “Delete”.</td></tr>
</table>

Test Case:

Case	Description	Input	Expect Output
1	Test for the system shall delete user account from the system	Username = nutritionists	The system shall delete the nutritionists from the database. atomsuperza Edit | Delete doctor Edit | Delete nutritionists Edit | Delete **After delete** atomsuperza Edit | Delete doctor Edit | Delete

Table 12: Test Case of **ST09:** Admin can delete nutritionists from system

3.3.3 Feature#3: Nutrition management.

ST10: Nutritionists can add food to the database.

<table>
<tr><td>Description</td><td>Test for UC08: Nutritionists can add food to the database.</td></tr>
<tr><td>Precondition</td><td>Test data must be prepared.
Food{ food name = “ banana2”, glycemic index =110 ,calorie = 110}</td></tr>
<tr><td>Prerequisites or Test input</td><td>Table: Food
<table>
<tr><td>food name</td><td>banana1</td><td>banana2</td></tr>
<tr><td>glycemic</td><td>100</td><td>110</td></tr>
<tr><td>calorie</td><td>100</td><td>110</td></tr>
</table></td></tr>
<tr><td>Test Script</td><td>1 Nutritionists go to food management page
2 Nutritionists click “create new”.
3 Nutritionists input food_name, glycemic index, calorie
4 Nutritionists click “submit”</td></tr>
</table>

Test Case:

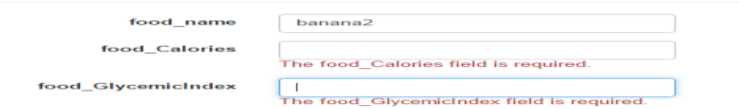
Case	Description	Input	Expect Output
1	Test for the system shall add the information of food. When the nutritionists add new food.	Food_name= “banana” food_Calories = “100” food_GlycemicIndex = “20”	The system shall record the data on the database. **Result:** 
2	Test when the nutritionists dose not input glycemic inde and calorie	Food_name= “” food_Calories = “” food_GlycemicIndex = “”	The system display error message “The food_Calories field is required.”, “The food_GlycemicIndex field is required.” 
3	Test input character in “food_Calories” and “food_GlycemicIndex”.	food_Calories = “abc” food_GlycemicIndex = “abc”	The system shall display error message “The field food_Calories must be a number.” And “The field food_GlycemicIndex must be a number.” 

Table 13: Test Case of **ST10:** Nutritionists can add food to the database.

ST11: Nutritionists can edit food on the database.

<table>
<tr><td>Description</td><td>Test for UC08: Nutritionists can edit food on the database.</td></tr>
<tr><td>Precondition</td><td>Test data must be prepared.
Food {food name = “ banana2”
glycemic index =110
calorie = 110}</td></tr>
<tr><td>Prerequisites or Test input</td><td>Table: Food
<table>
<tr><td>food name</td><td>banana1</td><td>banana2</td></tr>
<tr><td>glycemic</td><td>100</td><td>110</td></tr>
<tr><td>calorie</td><td>100</td><td>110</td></tr>
</table></td></tr>
<tr><td>Test Script</td><td>1 Nutritionists go to food management page
3 Nutritionists click “edit”.
4 Nutritionists input new food_name, glycemic index, calorie
5 Nutritionists click “submit”</td></tr>
</table>

Test Case:

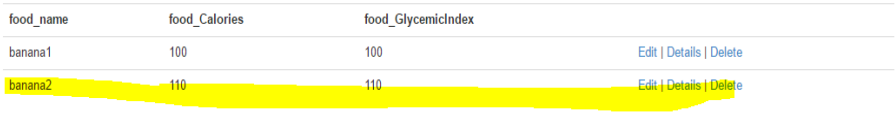
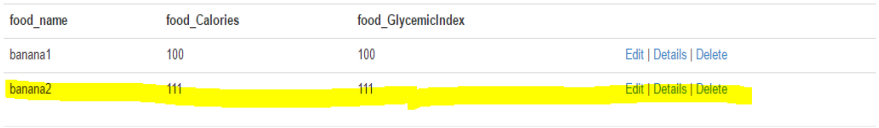
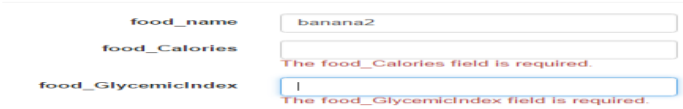
Case	Description	Input	Expect Output
1	Test for the system shall edit the information of food. When the nutritionists edit the information of food.	food name = “ banana2” glycemic index =111 calorie = 111	The system shall record the data on the database. **Result: before edit**  **After** 
2	Test when the nutritionists dose not input glycemic inde and calorie	food name = “ banana2” glycemic index = “ “ calorie = “ “	The system display error message “The food_Calories field is required.”, “The food_GlycemicIndex field is required.” 
3	Test input character in “food_Calories” and “food_GlycemicIndex ”.	glycemic index =110 calorie = 110	The system shall display error message “The field food_Calories must be a number.” And “The field food_GlycemicIndex must be a number.” 

Table 14: Test Case of **ST11:** Nutritionists can edit food on the database.

ST12: Nutritionists can delete food from the database.

<table>
<tr><td>Description</td><td>Test for UC08: Nutritionists can delete food from the database.</td></tr>
<tr><td>Precondition</td><td>Test data must be prepared.
Food {food name = {“banana1”, “ banana2”},
glycemic index = {100,110},
calorie = {100,110}}</td></tr>
<tr><td>Prerequisites or Test input</td><td>Table: Food
<table>
<tr><td>food name</td><td>banana1</td><td>banana2</td></tr>
<tr><td>glycemic</td><td>100</td><td>110</td></tr>
<tr><td>calorie</td><td>100</td><td>110</td></tr>
</table></td></tr>
<tr><td>Test Script</td><td>1 Nutritionists go to food management page
2 Nutritionists select food.
3 Nutritionists click “Delete”.</td></tr>
</table>

Test Case:

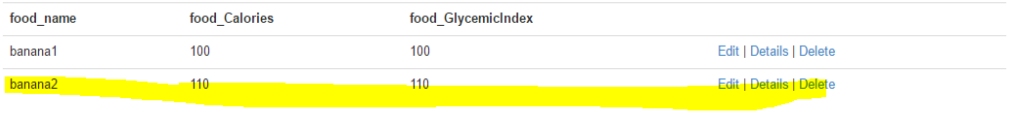
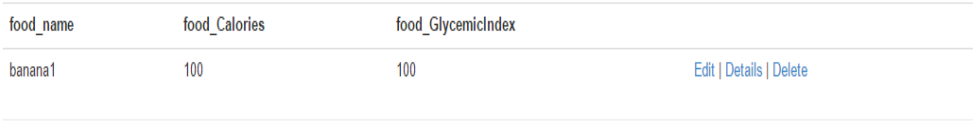
Case	Description	Input	Expect Output
1	Test for the system shall delete exercises from list. When the doctor or nurse click delete.	food name = “ banana2” glycemic index =110 calorie = 110	The system shall delete the food from the database.  **When delete** 

Table 15: Test Case of **ST12:** Nutritionists can delete food from the database.

3.3.4 Feature#4: Activity management.

ST13: Nurse/Doctor can add activity to the database.

<table>
<tr><td>Description</td><td>Test for UC09: Nurse/Doctor can add activity to the database.</td></tr>
<tr><td>Precondition</td><td>Test data must be prepared.
Symptom = {“ symptom1”}
suggestion= {“suggestion1”}
ProperExercise ={ “ProperExercise1”}
ExerciseInappropriate = {“ExerciseInappropriate1”}
StepExercise = {StepExercise1”}</td></tr>
<tr><td>Prerequisites or Test input</td><td>Table: Exercise
<table>
<tr><td>Symptom</td><td>Symptom</td><td>Symptom1</td></tr>
<tr><td>suggestion</td><td>suggestion</td><td>Suggestion1</td></tr>
<tr><td>ProperExercise</td><td>ProperExercise</td><td>ProperExercise1</td></tr>
<tr><td>ExerciseInappropriate</td><td>ExerciseInappropriate</td><td>ExerciseInappropriate1</td></tr>
<tr><td>StepExercise</td><td>StepExercise</td><td>StepExercise1</td></tr>
</table></td></tr>
<tr><td>Test Script</td><td>1 Nurse/Doctor go to activity management page
2 Nurse/Doctor click “create new”.
3 Nurse/Doctor input symptom, suggestion, ProperExercise, ExerciseInappropriate, StepExercise.
4 Nurse/Doctor click “submit”</td></tr>
</table>

Test Case:

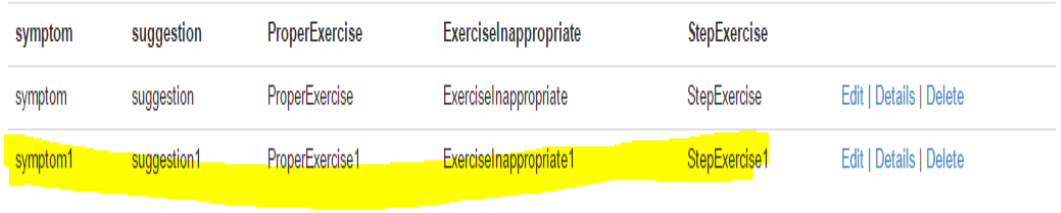
Case	Description	Input	Expect Output
1	Test for the system shall add the exercises to the database. When the doctor or nurse add new exercises.	Symptom = {“ symptom1”} suggestion= {“suggestion1”} ProperExercise ={ “ProperExercise1”} ExerciseInappropriate = {“ExerciseInappropriate1”} StepExercise = {StepExercise1”}	The system shall record the data on the database. **Result:** 

Table 16: Test Case of **ST13:** Nurse/Doctor can add activity to the database.

ST14: Nurse/Doctor can edit activity on the database.

<table>
<tr><td>Description</td><td>Test for UC09: Nurse/Doctor can edit activity on the database.</td></tr>
<tr><td>Precondition</td><td>Test data must be prepared.
Symptom = {“ symptom1”}
suggestion= {“suggestion1”}
ProperExercise ={ “ProperExercise1”}
ExerciseInappropriate = {“ExerciseInappropriate1”}
StepExercise = {StepExercise1”}</td></tr>
<tr><td>Prerequisites or Test input</td><td>Table: Exercise
<table>
<tr><td>Symptom</td><td>Symptom</td><td>Symptom1</td></tr>
<tr><td>suggestion</td><td>suggestion</td><td>Suggestion1</td></tr>
<tr><td>ProperExercise</td><td>ProperExercise</td><td>ProperExercise1</td></tr>
<tr><td>ExerciseInappropriate</td><td>ExerciseInappropriate</td><td>ExerciseInappropriate1</td></tr>
<tr><td>StepExercise</td><td>StepExercise</td><td>StepExercise1</td></tr>
</table></td></tr>
<tr><td>Test Script</td><td>1 Nurse/Doctor go to activity management page
2 Nurse/Doctor click “edit”.
3 Nurse/Doctor input symptom, suggestion, ProperExercise, ExerciseInappropriate, StepExercise.
4 Nurse/Doctor click “submit”</td></tr>
</table>

Test Case:

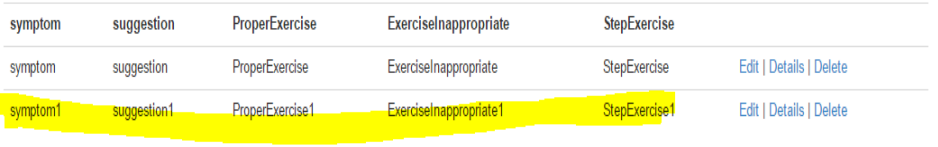
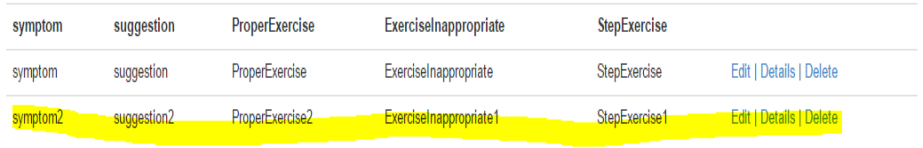
Case	Description	Input	Expect Output
1	Test for the system shall edit the information of exercises. When the doctor or nurse edit the information of exercises.	Symptom = {“ symptom2”} suggestion= {“suggestion2”} ProperExercise ={ “ProperExercise2”} ExerciseInappropriate = {“ExerciseInappropriate1”} StepExercise = {StepExercise1”}	The system shall record the data on the database. **Result: Before edit**  **After** 

Table 17: Test Case of **ST14:** Nurse/Doctor can edit activity on the database.

ST15: Nurse/Doctor can delete activity from the database.

<table>
<tr><td>Description</td><td>Test for UC09: Nurse/Doctor can delete activity from the database.</td></tr>
<tr><td>Precondition</td><td>Test data must be prepared.
Symptom = {“symptom”,“ symptom1”}
suggestion= {“suggestion”, “suggestion1”}
ProperExercise ={ “ProperExercise”, “ProperExercise1”}
ExerciseInappropriate = {“ExerciseInappropriate”, “ExerciseInappropriate1”}
StepExercise = {“StepExercise”, “StepExercise1”}</td></tr>
<tr><td>Prerequisites or Test input</td><td>Table: Exercise
<table>
<tr><td>Symptom</td><td>Symptom</td><td>Symptom1</td></tr>
<tr><td>suggestion</td><td>suggestion</td><td>Suggestion1</td></tr>
<tr><td>ProperExercise</td><td>ProperExercise</td><td>ProperExercise1</td></tr>
<tr><td>ExerciseInappropriate</td><td>ExerciseInappropriate</td><td>ExerciseInappropriate1</td></tr>
<tr><td>StepExercise</td><td>StepExercise</td><td>StepExercise1</td></tr>
</table></td></tr>
<tr><td>Test Script</td><td>1 Nurse/Doctor go to activity management page
2 Nurse/Doctor select activity.
3 Nurse/Doctor clicks “Delete”.</td></tr>
</table>

Test Case:

Case	Description	Input	Expect Output
1	Test for the system shall delete exercises from list. When the doctor or nurse click delete.	Symptom = symptom1 suggestion= suggestion1 ProperExercise = "ProperExercise1" ExerciseInappropriate = "ExerciseInappropriate1" StepExercise = "StepExercise1"	The system shall delete the activity from the database.  **When delete.** 

Table 18: Test Case of **ST15:** Nurse/Doctor can delete activity from the database.

3.3.5 Feature#5: Activity plan and recommend system.

ST16: Patient can select the activity from database.

<table>
<tr><td>Description</td><td>Test for UC10: Patient can select the activity from database.</td></tr>
<tr><td>Precondition</td><td>Test data must be prepared.
Symptom = {“symptom”,“ symptom1”}
suggestion= {“suggestion”, “suggestion1”}
ProperExercise ={ “ProperExercise”, “ProperExercise1”}
ExerciseInappropriate = {“ExerciseInappropriate”, “ExerciseInappropriate1”}
StepExercise = {“StepExercise”, “StepExercise1”}</td></tr>
<tr><td>Prerequisites or Test input</td><td>Table: Exercise
<table>
<tr><td>Symptom</td><td>Symptom</td><td>Symptom1</td></tr>
<tr><td>suggestion</td><td>suggestion</td><td>Suggestion1</td></tr>
<tr><td>ProperExercise</td><td>ProperExercise</td><td>ProperExercise1</td></tr>
<tr><td>ExerciseInappropriate</td><td>ExerciseInappropriate</td><td>ExerciseInappropriate1</td></tr>
<tr><td>StepExercise</td><td>StepExercise</td><td>StepExercise1</td></tr>
</table></td></tr>
<tr><td>Test Script</td><td>1 Patients go to health plan page.
2 Patient click “Exercises suggestion” link button.
3 Patient select symptom.
4 Patient must click “select”
5 The system re view food name to patient.</td></tr>
</table>

Test Case:

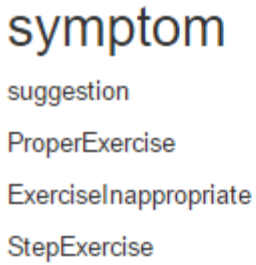
Case	Description	Input	Expect Output
1	Test for the system shall get list of exercises from table "Exercises" on the database. And display list of exercises to the patient.	Symptom = "symptom", "symptom1"	The system shall provide list of symptom to patient. 
2	Test for the system shall get exercises suggestion from table "Exercises" on the database. And display exercises suggestion to the patient.	Symptom = "symptom",	The system shall provide suggestion about precautions exercise. 

Table 19: Test Case of **ST16:** Patient can select the activity from database.

3.3.6 Feature#6: Nutrition plan system.

ST17: Patient can select the foods from database.

<table>
<tr><td>Description</td><td>Test for UC12: Patient can select the foods from database.</td></tr>
<tr><td>Precondition</td><td>Test data must be prepared. Food {“banana1”, “banana2”}.</td></tr>
<tr><td>Prerequisites or Test input</td><td>Table: Food
<table>
<tr><td>food name</td><td>banana1</td><td>banana2</td></tr>
<tr><td>glycemic</td><td>100</td><td>110</td></tr>
<tr><td>calorie</td><td>100</td><td>110</td></tr>
</table></td></tr>
<tr><td>Test Script</td><td>1 Patients go to health plan page.
2 Patient select foods name.
3 Patient must click “add”
4 The system re view food name to patient.</td></tr>
</table>

Test Case:

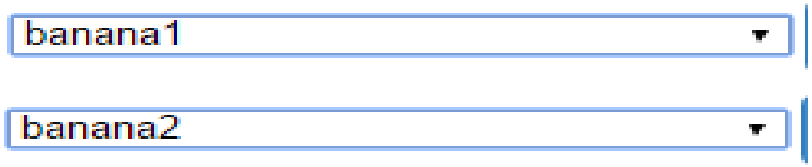
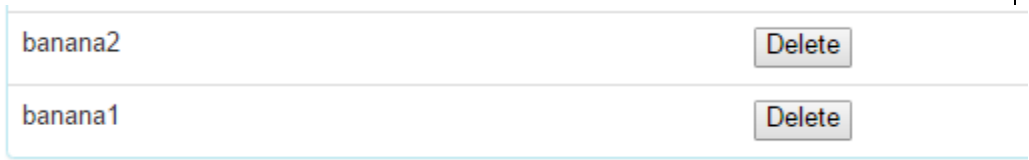
Case	Description	Input	Expect Output
1	Test for the system shall provide list of food from the database table “Food” to the patient	Food = “ banana1” , “banana2”	The system shall provide list of food name to patient. 
2	Test for the system shall add food to table “TodayFood” on the database. And display to the patient.	Food = “ banana1”	The system shall display names of food to patient. 

Table 20: Test Case of **ST17:** Patient can select the foods from database.

3.3.7 Feature#7: Health monitors system.

ST18: Patient can view body mass value that calculated.

<table>
<tr><td>Description</td><td>Test for UC16: Patient can view body mass value that calculated.</td></tr>
<tr><td>Precondition</td><td>Test data must be prepared. Height{123} , Weight{213}, Gender{“male”}</td></tr>
<tr><td>Prerequisites or Test input</td><td>Table: AspUser
<table>
<tr><td>First name</td><td>qwe</td></tr>
<tr><td>Last name</td><td>wqr</td></tr>
<tr><td>Gender</td><td>Male</td></tr>
<tr><td>Age</td><td>123</td></tr>
<tr><td>Height</td><td>123</td></tr>
<tr><td>Weight</td><td>213</td></tr>
</table></td></tr>
<tr><td>Test Script</td><td>1. Patient go to a health plan page</td></tr>
</table>

Test Case:

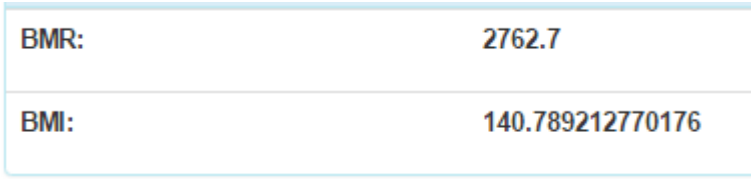
Case	Description	Input	Expect Output
1	Test for the system shall get the data of height, weight and gender to calculate BMR and BMI. And the system shall display to the patient.	Height = 123 Weight =213 Gender = male	The system redirect to health plan page and display BMI. 

Table 21: Test Case of **ST18:** Patient can view body mass value that calculated.

ST19: Patient can record blood sugar.

Description	Test for **UC15:** Patient can record blood sugar.
Precondition	Test data must be prepared. Blood sugar value {120, 110, -100}
Prerequisites or Test input	**Table: UserGlycemic** UserId: 1 Value: 120, 110
Test Script	1. Patient go health page 2. Patient input “blood sugar” 3. Patient clicks “add” button.

Test Case:

Case	Description	Input	Expect Output
1	Test for the system shall record the value of blood sugar to the database	Blood sugar = 120, 110	The system record value of blood sugar to the database
3	Test when input blood sugar value is equal minus	Blood sugar= “-100”	The system display error message “value must not minus”

Table 22: Test Case of **ST19:** Patient can record blood sugar.

ST20: Patient can view statistic graph of blood sugar.

Description	Test for **UC17:** Patient can view statistic graph of blood sugar.
Precondition	Test data must be prepared. Blood sugar value {120, 110}
Prerequisites or Test input	**Table: UserGlycemic** UserId: 1 Value: 120, 110
Test Script	1. Patient go health page

Test Case:

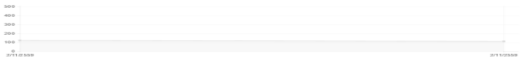
Case	Description	Input	Expect Output
1	Test for the system shall get value of blood sugar to analyze and display graph to the patient.	Blood sugar value = 120 Blood sugar value = 110	The system shall redirect to health plan page and dis play graph to user 

Table 23: Test Case of **ST20:** Patient can view statistic graph of blood sugar.

ST21: Patient can get interpretation about blood sugar from system.

Description	Test for **UC18:** Patient can get interpretation about blood sugar from system.
Precondition	Test data must be prepared. Blood sugar value {120, 110}
Prerequisites or Test input	**Table: UserGlycemic** UserId: 1 Value: 120, 110
Test Script	1. Patient go health page

Test Case:

Case	Description	Input	Expect Output
1	Test for the system shall get value of blood sugar to analyze and display interpretation to the patient.	Blood sugar value = 120 Blood sugar value = 110	The system shall redirect to health plan page. And display interpretation about blood sugar to the patient.

Table 24: Test Case of **ST21:** Patient can get interpretation about blood sugar from system.

3.3.8 Feature#8: Behavior monitors system.

ST22: Patient can select food.

<table>
<tr><td>Description</td><td>Test for UC19: Patient can select food.</td></tr>
<tr><td>Precondition</td><td>Test data must be prepared. Food {“banana1”, “banana2”}.</td></tr>
<tr><td>Prerequisites or Test input</td><td>Table: Food
<table>
<tr><td>food name</td><td>banana1</td><td>banana2</td></tr>
<tr><td>glycemic</td><td>100</td><td>110</td></tr>
<tr><td>calorie</td><td>100</td><td>110</td></tr>
</table></td></tr>
<tr><td>Test Script</td><td>1 Patients go to behavior page.
2 Patient select foods name.
3 Patient must click “select”
4 The system review food name to patient.</td></tr>
</table>

Test Case: Edit

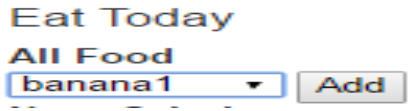
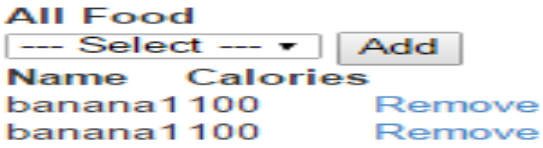
Case	Description	Input	Expect Output
1	Test for the system shall provide list of food from the database table “Food” to the patient	Food = “ banana1” , “banana2”	The system shall provide list of food name to patient. 
2	Test for the system shall add food to table “TodayFood” on the database. And display to the patient.	Food = “ banana1”	The system shall display names of food to patient. 

Table 25: Test Case of **ST22:** Patient can select food.

ST23: Patient can submit and view about analyze result.

<table>
<tr><td>Description</td><td>Test for UC21: Patient can submit and view about analyze result.</td></tr>
<tr><td>Precondition</td><td>Test data must be prepared. Food{“banana1”, “banana2”}, Medicine{“para1”, “para2”}</td></tr>
<tr><td>Prerequisites or Test input</td><td>Table: Food
<table>
<tr><td>food name</td><td>banana1</td><td>banana2</td></tr>
<tr><td>glycemic</td><td>100</td><td>110</td></tr>
<tr><td>calorie</td><td>100</td><td>110</td></tr>
</table>
Table: Medicine
<table>
<tr><td>name</td><td>para</td><td>para2</td></tr>
</table></td></tr>
<tr><td>Test Script</td><td>1 Patients go to behavior page.
2 Patient select data.
3 Patient must click submit.</td></tr>
</table>

Test Case: Edit

Case	Description	Input	Expect Output
1	Test for the system shall display analyze result to the patient. After the patient submit the data.	Food = banana1 Food = banana2 Medicine = para Medicine = para2	The system shall provide analyze result to patient. 

Table 26: Test Case of **ST23:** Patient can submit and view about analyze result.

3.4 Unit Test

Black Box technique (UT01-UT11)

Use Case ID	Use case name
UT01	AccountController: AddTodayFood()
UT02	AccountController: addTodayfood()
UT03	AccountController: RemoveTodayFood()
UT04	AccountController: HealthPlan()
UT05	AccountController: AddGlycemic()
UT06	AccountController: addUserGlycemic()
UT07	AccountController: AddFood()
UT08	AccountController: addUSerFood()
UT09	AccountController: DeleteFood()
UT10	AccountController: AddMedicine()
UT11	AccountController: addMedicineData()
UT12	AccountController: DeleteMedicine()
UT13	AccountController: EditUserProfile()
UT14	AccountController: Register()
UT15	AccountController: Login()
UT16	AccountController:LogOff()
UT17	FoodController:Create()
UT18	FoodController:Edit()
UT19	FoodController:Delete()
UT20	ExercisesController:Create()
UT21	ExercisesController: Edit()
UT22	ExercisesController:Delete()
UT23	ManageController ChangePassword()

Table 26: Black Box technique (UT01-UT11)

3.4.1 Class: AccountController

UT01: public ActionResult AddTodayFood(int value = 0)
Description: Test for the method AddTodayFood has redirected is correct after process success.

Test Case: AddTodayFood()

Case	Description	Input	Expect Output
1	Tests method AddTodayFood, test return type.	value = 1	Object of class ActionResult
2	Tests method redirects result	value = 1	Redirect to Account

Table 27: Test Case of **UT01**

UT02: public virtual TodayFood addTodayfood(UserProfileInfo profile, int value = 0)

Description: Test for insert the data of food to table TodayFood on the database. This AddTodayFood function uses to add food to analyze select food by food id

.

Test Case: addValueTodayfood()

Case	Description	Input	Expected Output
1	Test for insert user_id and food_id to method addTodayfood. (Assert.IsNotNull)	UserProfileInfo {UserId = 1}, value = 1	TodayFood {User Id = 1, Food id = 1}

Table 28: Test Case of **UT02**

Test Case: addNullValueTodayfood()

Case	Description	Input	Expect Output
1	Test for insert user_id and food_id to method addTodayfood. When value equal null. (Assert.IsNull)	null	null

Table 29: Test Case of **UT02**

UT03: public ActionResult RemoveTodayFood(int value = 0)

Description: Test for the method RemoveTodayFood has redirected is correct after delete process success.

Test Case: RemoveTodayFoodTest()

Case	Description	Input	Expect Output
1	Tests method RemoveTodayFood, test return type.	value = 1	Object of class ActionResult
2	Tests method redirects result	value =1	Redirect to Account

Table 30: Test Case of **UT03**

UT04: public ActionResult HealthPlan()

Description : Test for calculate the BMI, BMR, Food calorie, Food glycemic value This HealthPlan function uses to calculate by height, weight, age, food calorie, food glycemic,

Test Case: HealthPlanTest()

Case	Description	Input					Expected Output
		Height	Weight	Age	Food calorie	Food glycemic	
1	Test to calculate BMI by input user height, user weight	1	1	-	-	-	BMI = 10000
2	Test to calculate BMR for male by input user height, user weight, and user age.	1	1	1	-	-	BMRM =77.9
3	Test to calculate BMR for female by input user height, user weight, and user age.	1	1	1	-	-	BMRF =671.7

4	Test to calculate calorie of food	-	-	-	100,100,200	-	Calorie = 400
5	Test to calculate glycemic index of food	-	-	-	-	100,100,200	Glycemic = 400

Table 31: Test Case of **UT04**

UT05: public ActionResult AddGlycemic(int value = 0)

Description: Test for the method AddGlycemic has redirected is correct after process success.

Test Case: AddGlycemic**()**

Case	Description	Input	Expect Output
1	Test returns type of the method AddGlycemic().	value = 0	Object of class ActionResult
3	Tests method redirects result.	value = 0	Redirect to HealthPlan

Table 32: Test Case of **UT05**

UT06: public virtual UserGlycemic addUserGlycemic (UserProfileInfo profile, int value = 0)

Description: Test for insert the value of blood sugar to table UserGlycemic on the database. This AddGlycemic function uses to add blood sugar value to system include data user id, blood value, and date.

Test Case: addValueAddGlycemic**()**

Case	Description	Input	Expect Output
1	Test for insert user_id and value to method addUserGlycemic. (Assert.IsNotNull)	UserProfileInfo{id=1}, value = 100	UserGlycemic{User id =1, Value =100}

Table 33: Test Case of **UT06**

Test Case: addNullValueAddGlycemic**()**

Case	Description	Input	Expect Output
1	Test for insert user_id and value to method addUserGlycemic. When value equal null. (Assert.IsNull)	null	null

Table 34: Test Case of **UT06**

UT07: public ActionResult AddFood(int id = 0)

Description: Test for the method AddFood has redirected is correct after process success.

Test Case: AddFood()

Case	Description	Input	Expect Output
1	Test returns type of the method.	id = 0	Object of class ActionResult
2	Tests method redirects result, when redirects correct url "/Account/HealthPlan"	Id = 0	Redirect to HealthPlan

Table 35: Test Case of **UT07**

UT08: public virtual UserFood addUSerFood(UserProfileInfo profile, int id = 0)

Description: Test for method inserts the data of food to table UserFood on the database. This addUSerFood method uses to select food from database Food to add in database UserFood table.

Test Case: addValue AddFood()

Case	Description	Input	Expect Output
1	Test for insert user_id and food_id to method addUSerFood. (Assert.IsNotNull)	UserProfileInfo{id=1},id = 1	UserFood{User id = 1, Food Id =1}

Table 36: Test Case of **UT08**

Test Case: addNullValue AddFood()

Case	Description	Input	Expect Output
1	Test for insert user_id and food_id to method addUSerFood. When value equal null. (Assert.IsNull)	null	null

Table 37: Test Case of **UT08**

UT09: public ActionResult DeleteFood(int id = 0)

Description: Test for the method DeleteFood has redirected is correct after delete process is success.

Test Case: DeleteFoodTest()

Case	Description	Input	Expect Output
1	Test returns type of the method.	id = 0	Object of class ActionResult
2	Tests method redirects result, when redirects correct url.	id = 0	Redirect to HealthPlan

Table 38: Test Case of **UT09**

UT10: public ActionResult AddMedicine(string name = null)

Description: **Description:** Test for the method AddMedicine has redirected is correct after process success.

Test Case: AddMedicine()

Case	Description	Input	Expect Output
1	Test returns type of the method.	name = “para”	Object of class ActionResult
2	Tests method redirects result, when redirects correct url.	name = “para”	Redirect to DisplayProfile

Table 39: Test Case of **UT10**

UT11: public virtual Medicine addMedicineData (UserProfileInfo profile , string name = null)

Description: Test for method inserts the data of medicine to table Medicine on the database.

Test Case: addValueAddMedicine()

Case	Description	Input	Expect Output
1	Test for insert user_id and name to method addMedicineData. (Assert.IsNotNull)	UserProfileInfo{id=1}, name= “para”	Medicine {User id =1, MedicineName = “para”}

Table 40: Test Case of **UT11**

Test Case: addNullValueAddMedicine()

Case	Description	Input	Expect Output
1	Test for insert user_id and name to method addMedicineData. When value equal null. (Assert.IsNull)	null	null
2	Test for insert user_id and name to method addMedicineData. When name equal null. (Assert.IsNull)	UserProfileInfo{id=1}, name= null	null

Table 41: Test Case of **UT11**

UT12: public ActionResult DeleteMedicine(int id = 0)

Description: Test for the method DeleteMedicine has redirected is correct after delete process is success.

Test Case: DeleteMedicineTest()

Case	Description	Input	Expect Output
1	Test returns type of the method.	id = 0	Object of class ActionResult
2	Tests method redirects result, when redirects correct url.	id = 0	Redirect to DisplayProfile

Table 42: Test Case of **UT12**

UT13: public ActionResult EditUserProfile(UserProfileInfo userprofile)

Description: Test for method EditUserProfile has redirected is correct after delete process method edit/update the information of user in the table AspNetUser on the database is success.

Test Case: EditUserProfileTest(UserProfileInfo userprofile)

Case	Description	Input	Expect Output
1	Test returns type of the method.	UserProfileInfo{id = 1}	Object of class ActionResult
2	Tests method redirects result, when redirects correct url.	UserProfileInfo{id = 1}	Redirect to Index

Table 43: Test Case of **UT13**

UT14: public async Task<ActionResult> Register(RegisterViewModel model)

Description: Test for method register user account to insert data to the database. This Register method uses to register user account to the system for using the system.

Test Case: RegisterTestValid()

Case	Description	Input	Expect Output
1	Test for insert username, email, first name, last name gender, age, height, weight, address, city, zipcode to method Register. (Assert.IsNotNull)	RegisterViewModel{Username = “atom”, Email = “at@om.com”, FirstName = “Jira” , LastName = “yu”, Gender= ”male”, Age = 20 , Height = 170 , Weight = 60 , Address = “139 m.7 ” , City = “Phayao” , Zipcode = 56000 ;}	Register {Username = “atom” Email = “at@om.com” FirstName = “Jira” LastName = “yu” Gender= ”male” Age = 20 Height = 170 Weight = 60 Address = “139 m.7 ” City = “Phayao” Zipcode = 56000}

Table 44: Test Case of **UT14**

Test Case: RegisterTestInValid()

Case	Description	Input	Expect Output
1	Test for insert username, email, first name, last name gender, age, height, weight, address, city, zipcode to method Register. When value equal null. (Assert.IsNull)	null	Null
2	Test for insert username, email, first name, last name gender, age, height, weight, address, city, zipcode to method Register. When some value equal null. (Assert.IsNull)	RegisterViewModel{Username = null, Email = “at@om.com”, FirstName = null, LastName = null, Gender= ”male”, Age = 20 , Height = 170 , Weight = 60 , Address = “139 m.7 ” , City = “Phayao” , Zipcode = 56000 ;}	Null

Table 45: Test Case of **UT14**

UT15: public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)

Description: Test for method login system. This Login method uses to login to using the system.

Test Case: LoginValid()

Case	Description	Input	Expect Output
1	Test for insert username, password method Login. (Assert.IsNotNull)	LoginViewModel {Username = "Admin@Admin.com", Password = "Password#1"}	Login {Username = "Admin@Admin.com" Password = "Password#1"}

Table 46: Test Case of **UT1**5

Test Case: LoginInValid()

Case	Description	Input	Expect Output
1	Tests method Login, when input value is null (Assert.IsNull)	null	null

Table 47: Test Case of **UT15**

UT016: public ActionResult LogOff()

Description: Test for the method Logoff has redirected is correct after user logoff from the system.

Test Case: LogOffTest**()**

Case	Description	Input	Expect Output
1	Tests method redirects result, when redirects correct url.	“ ”	Redirect to Home/Index

Table 48: Test Case of **UT16**

3.4.2 Class: FoodController

UT017: public ActionResult Create([Bind(Include = “id_food,food_name,food_Calories,food_GlycemicIndex”)] Food food)

Description: Test for insert the foods to table Food on the database. This Create function uses to add foods info to the system include data id, name, calories, glycemic index.

Test Case: CreateValid**()**

Case	Description	Input	Expect Output
1	Test for insert food_id, food_name, food_calories and food_Glycemic to method Create. (Assert.IsNotNull)	Bind{ id_food =1, food_name = “banana”, food _calories = 100, food _Glucemic =100}	Create {id_food =1, food_name = “banana”, food _calories = 100, food _Glucemic =100}

Table 49: Test Case of **UT17**

Test Case: CreateInValid**()**

Case	Description	Input	Expect Output
1	Tests input value equal null. (Assert.IsNull)	null	null
2	Tests input some value equal null. (Assert.IsNull)	Bind{ id_food =1, food_name = null food _calories = 100, food _Glucemic =100}	null

Table 50: Test Case of **UT17**

UT018: public ActionResult Edit([Bind(Include = "id_food,food_name,food_Calories,food_GlycemicIndex")] Food food)

Description: Test for edit the foods in table Food on the database. This Edit function uses to edit foods info to the system include data id, name, calories, glycemic index.

Test Case: EditfoodValid **()**

Case	Description	Input	Expect Output
1	Test for insert food_id, food_name, food_calories and food_Glycemic to method Create. (Assert.IsNotNull)	Bind{ id_food =1, food_name = “banana2”, food _calories = 101, food _Glucemic =101}	Edit{id_food =1, food_name = “banana2”, food _calories = 101, food _Glucemic =101}

Table 51: Test Case of **UT18**

Test Case: EditfoodInValid **()**

Case	Description	Input	Expect Output
1	Tests input value equal null. (Assert.IsNull)	null	null

Table 52: Test Case of **UT18**

UT019: public ActionResult DeleteConfirmed(int id)

Description: Test for the method DeleteConfirmed has redirected is correct after delete process is success.

Test Case: DeleteTest**()**

Case	Description	Input	Expect Output
1	Test returns type of the method.	id = 1	Object of class ActionResult
2	Tests method redirects result, when redirects correct url.	id = 1	Redirect to Index

Table 53: Test Case of **UT19**

3.4.3 Class: ExerciseController

UT020: public ActionResult Create([Bind(Include = "id,symptom,suggestion,ProperExercise,ExerciseInappropriate,StepExercise")] Exercises exercises)

Description: Test for insert the exercises to table Exercises on the database. This Create function uses to add exercises info to the system include data id, symptom, suggestion, proper exercise, exercise inappropriate, step exercise**.**

Test Case: CreateExercisesValid**()**

Case	Description	Input	Expect Output
1	Test for insert id, symptom,suggestion,ProperExercise,ExerciseInappropriate,StepExercise to method Create. (Assert.IsNotNull)	Bind {id =1, symptom = “knee pain”, suggestion= “should not take him”, ProperExercise, = “exercise”, ExerciseInappropriate = “high impact events”, StepExercise = “1. Warm 2. Exercise”;}	Create {id =1 Symptom = “knee pain” suggestion= “should not take him” ProperExercise, = “exercise” ExerciseInappropriate = “high impact events” StepExercise = “1. Warm 2. Exercise”}

Table 54: Test Case of **UT20**

Test Case: CreateExercisesInValid**()**

Case	Description	Input	Expect Output
1	Tests input value equal null. (Assert.IsNull)	null	null
2	Tests input some value equal null. (Assert.IsNull)	Bind {id =1, symptom = “knee pain”, suggestion= “should not take him”, ProperExercise, = null, ExerciseInappropriate = null, StepExercise = null;}	null

Table 55: Test Case of **UT20**

UT021: public ActionResult Edit([Bind(Include = "id,symptom,suggestion,ProperExercise,ExerciseInappropriate,StepExercise")] Exercises exercises)

Description: Test for edit the exercises in table Exercises on the database. This Edit function uses to edit exercises info to the system include data id, symptom, suggestion, proper exercise, exercise inappropriate, step exercise**.**

Test Case: EditExercisesValid **()**

Case	Description	Input	Expect Output
1	Test for insert id, symptom,suggestion,ProperExercise,ExerciseInappropriate,StepExercise to method Edit. (Assert.IsNotNull)	Bind {id =1, symptom = “knee pain”, suggestion= “should not take him”, ProperExercise, = “exercise”, ExerciseInappropriate = “high impact events”, StepExercise = “1. Warm 2. Exercise”;}	Edit {id = 1, symptom = “knee pain” suggestion= “should not take him” ProperExercise, = “exercise” ExerciseInappropriate = “high impact events” StepExercise = “1. Warm 2. Exercise”}

Table 56: Test Case of **UT21**

Test Case: EditExercisesInValid **()**

Case	Description	Input	Expect Output
2	Tests input value equal null. (Assert.IsNull)	null	null

Table 57: Test Case of **UT21**

UT022: public ActionResult DeleteConfirmed(int id)

Description: Test for the method DeleteConfirmed has redirected is correct after delete process is success.

Test Case: DeleteExercisesTest**()**

Case	Description	Input	Expect Output
1	Tests method redirects result, when redirects correct url.	id =1	Redirect to Index

Table 58: Test Case of **UT22**

3.4.4 Class: ManageController

UT023: public ActionResult ChangePassword(ChangePasswordViewModel model)

Description: Test for change password of user. This ChangePassword method uses to change password for login to the system include new password, old password, confirm password**.**

Test Case: ChangePasswordTest()

Case	Description	Input	Expect Output
1	Tests method redirects result, when redirects correct url.	ChangePasswordViewModel {NewPassword = “testpassword”, OldPassword = “testpass”, ConfirmPassword = “testpassword”}	Redirect to Index
2	Test returns type of the method.	ChangePasswordViewModel {NewPassword = “testpassword”, OldPassword = “testpass”, ConfirmPassword = “testpassword”}	Object of class ActionResult

Table 59: Test Case of **UT23**

Test Case: ChangePasswordValid **()**

Case	Description	Input	Expect Output
1	Test for insert new password, old password and confirm password to method ChangePassword (Assert.IsNotNull)	ChangePasswordViewModel {NewPassword = “testpassword”, OldPassword = “testpass”, ConfirmPassword = “testpassword”}	ChangePassword {New password = “testpassword” Old password = “testpass” Confirm password = “testpassword”}

Table 60: Test Case of **UT23**

Test Case: ChangePasswordInValid **()**

Case	Description	Input	Expect Output
1	Tests input value equal null. (Assert.IsNull)	null	null

Table 61: Test Case of **UT23**