|  |  |
| --- | --- |
|  | **MINISTRY OF EDUCATION AND TRAINING** |

|  |
| --- |
| **FPT UNIVERSITY** |
| Capstone Project Document |
| CREDIT RANKING SYSTEM |
|  |
| |  |  | | --- | --- | | **FBD GROUP** | | | **Group Members** | Nguyen Thanh Long 00455  Phan Le Quang Chinh  Pham Thi Phuong Thao  Dao Mai Ly  Tran Thi Huong Giang | | **Supervisor** | Mr. QuangLT | | **Ext Supervisor** |  | | **Capstone Project code** | FBD | |
|  |

|  |
| --- |
| - Hanoi, March, 2011 - |

**REPORT 4 DESIGN DESCRIPTION**

# INTRODUCTION

## Design overview

The purpose of the document is to provide a high-level to very detailed description of System Design Specification.

It is mainly aimed for system engineers, system developers and system maintenance.

First, the document will explain architectural design of the system, which is a very high-level of system specification to get best understand of general components that will be building blocks.

Later, it provides detailed specification of each module in each component, which is class-level description to help develop system in ease. Each module will be mapped to each feature in Software Requirements Specification carefully in the form of MVC style. Reader can easily find information about each feature processing in each module description, they are modular and traceable.

Finally, database diagram will be specified as well as table list explanation.

## Requirements traceability matrix

# SYSTEM ARCHITECTURAL DESIGN

## Chosen of System Architecture

### Deployment diagram

The system will be deployed as three-tier architecture.

As the diagram below, the tier 1 is client-side, users connect to Web interface and access the system by requests to Server at tier 2.

The Server then access Database server at tier 3 to handle or retrieve data then send back to client-side.



Figure 2.1.1. System deployment diagram

### Architecture at Application Server

At application Server, the system will be developed using Model-View-Controller architectural style.

The client send a request to Server, Controller will receive request, retrieve or handle information using Data Model, then choose a View to return to Client-side.

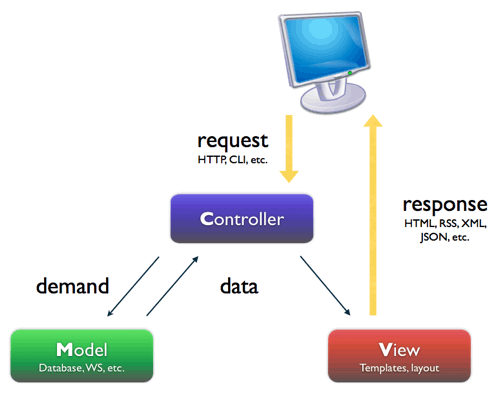


Figure 2.1.2 Application Architecture diagram

## Discussion of alternative Design

An alternative method for this system architecture is three-layers style which is explained in the following figure.

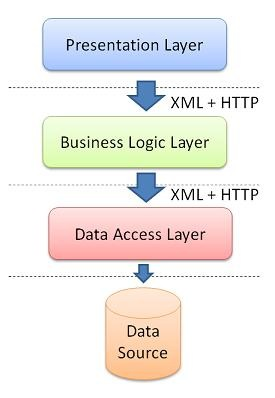


Figure 2.2 Layer architecture diagram

The system could be decomposed into three layers from high-level to low.

Presentation layer: Responsible for displaying User Interface, receive and send requests to Business Logic layer

Business Logic layer: Handle Domain logic and related business utilities

Data Access Layer: In charge of connecting to Database, handle and retrieve data directly

The architecture is so easy to develop, each layer is responsible for a specific job which can make separation of components but to build the system from this style, it can take longer time due to lacking of supportability

We still choose MVC model because of fast development and .NET Framework already provided MVC template which supports developers better in designing and implementation. With MVC, the components of system can be decomposed to specific modules which show clearly each business process and the relations between requirements specification and design.

## System interface description

# GENERAL SYSTEM COMPONENTS

## Components diagram

Diagram below will demonstrate high-level components of Credit Ranking System which will be the building blocks of the system.

For explanation, refer to section 3.2 to read brief explanation

For more details of these components, refer to section 4

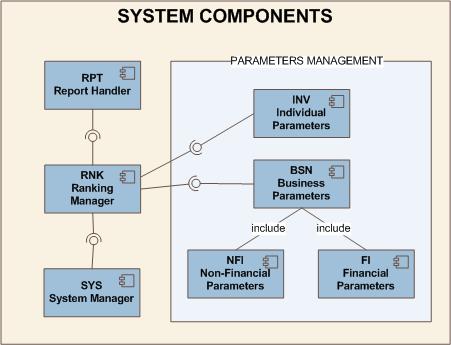


Figure 3.1 General components diagram

## Brief explanation of Components

This is table describing the 7 system’s components in the Figure 3.1. Component BSN, INV, FI and NFI are belong to Parameters management block which can be thought of a subsystem.

|  |  |  |
| --- | --- | --- |
| **ID** | **COMPONENT NAME** | **COMPONENT DESCRIPTION** |
| INV | Individual Parameters | **The component contains modules responsible for managing individual customers parameters such as: Basic index management, collateral index management…** |
| BSN | Business Parameters | **Contains modules of managing business customer parameters such as: business industries, business types, business lines, business scales…** |
| FI | Financial Parameters | **Manages financial parameters including financial index, financial proportion, financial score, which is thought of a part of BSN, but so complex that it is separated as a single component** |
| NFI | Non-Financial Parameters | **Manages non-financial parameters including non-financial index, non-financial proportion, non-financial score, which is thought of a part of BSN, but so complex that it is separated as a single component** |
| SYS | System Manager | **Modules in charge of managing System parameters, login, logout, accounts management, log management…** |
| RNK | Ranking Manager | **Receive services provided by Parameters Management block to rank credit for customers:**  **To rank for individual customers, they need information provided by INV module**  **To rank for business customers, they need information provided by BSN module** |
| **RPT** | **Report Handler** | **Contains modules that handle exporting results of customers’ credit ranking** |

# DETAILED DESCRIPTION OF COMPONENTS

## Section Reading Guide Introduction

## Component BSN: Business Parameters Management

## Component FI: Financial Parameters Management

### MODULE: FINANCIAL INDEX

SRS Mapping: Feature 25

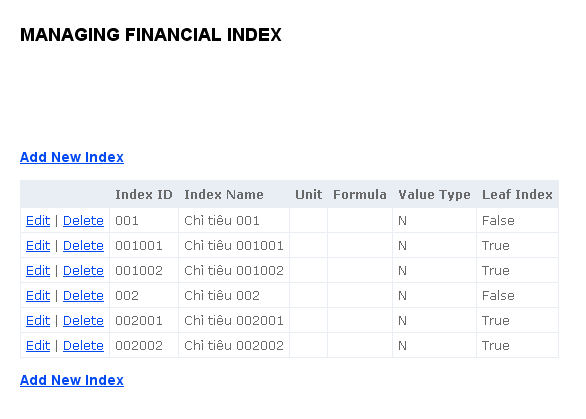
#### Module’s Class diagram



#### Class explanation

##### Class Index: View

###### Screen Image

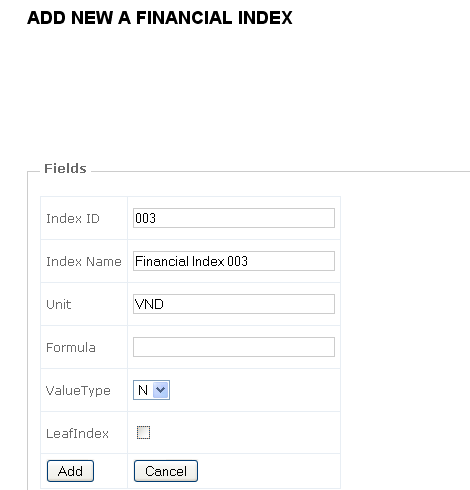


**Screen Description**

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: managing financial indexes |
| Create a Financial Index button x 2 | Button | The button for creating a Financial Index, clicking this will move to Adding page |
| List of financial indexes | Table | List all the financial indexes available in the database, each item will have a button [Edit] and [Delete] Click [Edit] will move to Edit view Click [Delete] will display dialog to verify the deleting the selected a Financial Index |

##### Class Add: View

**Screen Image**

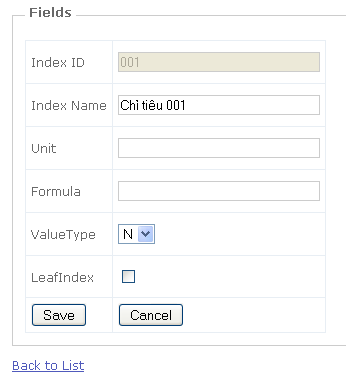
****

**Screen Description**

|  |  |  |  |
| --- | --- | --- | --- |
| No | COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| 1 | Page name | Label | The text describes the responsibility of the page: adding a new financial index |
| 2 | lblIndexID | Label | Display "Index ID" |
| 3 | lblIndexName | Label | Display "Index Name" |
| 4 | lblUnit | Label | Display "Unit" |
| 5 | lblFormula | Label | Display "Formula" |
| 6 | lblValueType | Label | Display "Value Type" |
| 7 | txtIndexID | textbox | user can enter the index id of the financial index |
| 8 | txtIndexName | textbox | text box for entering Index Name |
| 9 | txtUnit | textbox | text box for entering Unit |
| 10 | txtFormula | textbox | text box for entering formular |
| 11 | cmbValueType | combo box | combobox to enter value type (number or text) |
| 12 | chkLeafIndex | checkbox | check box to check whether this is leafindex or not |
| 13 | btnAddIndex | Button | After clicking on the button, add a new FinancialIndex to the system with the entered information then back to the Index View |
| 14 | btnCancel | Button | After clicking on the button, back to the Index View |

##### Class Edit: View

**Screen Image**

****

**Screen Description**

|  |  |  |  |
| --- | --- | --- | --- |
| No | COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| 1 | Page name | Label | The text describes the responsibility of the page: editing a Financial Index, also display the ID that is being edited |
| 2 | lblIndexID | Label | Display "Index ID" |
| 3 | lblIndexName | Label | Display "Index Name" |
| 4 | lblUnit | Label | Display "Unit" |
| 5 | lblFormula | Label | Display "Formula" |
| 6 | lblValueType | Label | Display "Value Type" |
| 7 | txtIndexID | textbox | user can enter the index id of the financial index |
| 8 | txtIndexName | textbox | text box for entering Index Name |
| 9 | txtUnit | textbox | text box for entering Unit |
| 10 | txtFormula | textbox | text box for entering formular |
| 11 | cmbValueType | combo box | combobox to enter value type (number or text) |
| 12 | chkLeafIndex | checkbox | check box to check whether this is leafindex or not |
| 13 | btnSave | Button | After clicking on the button, update the edited index to the DB with entered information, if successful then back to the Index View |
| 14 | btnCancel | Button | After clicking on the button, back to the Index View |

##### Class FIFinancialIndexController: Controller

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| FIFinancialIndexController | The class is responsible for receive requests from View to handle Add, Edit and Delete data | Controller | FI004 FI005 FI006 | Business.FinancialIndex | LyDM |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| METHODS | | | | | |
| NAME | ACCESS LEVEL | INPUT | PROCESS | OUTPUT | EXCEPTION |
| Index | public |  | if this user has sufficient right to use this function: - Use FIFinancialIndexLogic class to select all the financial indexes in the table Business.FinancialIndex then display to the [Index] View | ActionResult (Index View ) | Return Error message |
| Add | public |  | 1.Check User right 2. Forward to [Add] View | ActionResult (Index View ) | Return Error message |
| Add [Http-Post] | public | a new Financial Index information | 1. Receive information from parameter 2. Check user right and input validation. 3. Use Logic class to insert new Financial Index 3. Redirect to [Index] View with label displaying: "A new Financial Index has been added successfully" | ActionResult (Index View ) | Return the Add View Display error |
| Edit | public | ID of the Financial Index editted | 1. Receive ID from parameter 2. Use Logic class to select appropriate Financial Index from Business.FinancialIndex table 3. Display in [Edit] view | ActionResult (Edit View ) | Return Error message |
| Edit [Http-Post] | public | 1. ID of the Financial index input 2. The information of the Financial Index editted | 1. Receive ID from parameter 2. Use Logic class to update appropriate Financial Index with ID selected in DB 3. Display in [Index] view with label displaying: "The Financial Index with ID xyz has been editted successfully" | ActionResult (Index View ) | Return the Edit View Display error |
| Delete [Http-Post] | public | ID of the Financial Index selected | 1. Receive ID from parameter 2. Use Logic class to delete the Financial Index with selected ID from the Business.FinancialIndex table 3. Back to [Index] view with label displaying: "A Financial Index has been deleted successfully" | Index View | Return the Index view with error label. |

##### Class BusinessFinancialIndex

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| BusinessFinancialIndex | Work as DB interface and handle logic process for the controller |  | FI004 FI005 FI006 | Business.FinancialIndex | LyDM |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| METHODS | | | | | |
| NAME | ACCESS LEVEL | INPUT | PROCESS | OUTPUT | EXCEPTION |
| SelectFinancialIndex | public static | FBDEntities FBD | Select all the Financial Index in the table Business.FinancialIndex | List<BusinessFinancialIndex> |  |
| SelectFinancialIndexByID | public static | 1. FBDEntities FBD 2. string ID | Select the Financial Index in the table Business.FinancialIndex with input ID | BusinessFinancialIndex | Return null |
| SelectFinancialLeafIndex | public static | FBDEntities | Select all the Financial Indexes which is leaf index (index used to calculate) in BusinessFinancialIndex table | List<BusinessFinancialIndex> |  |
| AddFinancialIndex | public static | 1. FBDEntities FBD 2. A new FinancialIndex information | 1. Receive information from parameter 2. Insert new index into the Database 3. If successful, return 1 otherwise return 0 | int | Return 0 |
| EditFinancialIndex | public static | 1. FBDEntities FBD 2. ID 3. Information of the FinancialIndex editted | 1. Receive ID from parameter 2. Update appropriate data to table in DB 3. If successful, return 1 otherwise return 0 | int | Return 0 |
| DeleteFinancialIndex | public static | 1. FBDEntities FBD 2 ID of the Financial Index selected | 1. Receive ID from parameter 2. Delete the Financial Index with selected ID from database 3. If successful, return 1 otherwise return 0 | int | Return 0 |

#### Sequence diagrams

##### View Financial Index list



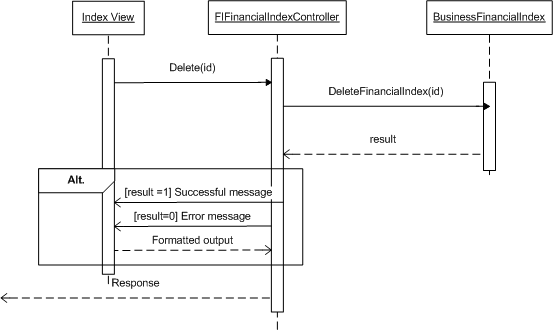
##### Add Financial Index



##### Edit Financial Index



##### Delete Financial Index



### MODULE: FINANCIAL INDEX LEVELS

SRS Mapping: Feature

#### Module’s class diagram

#### Class explanation

#### Sequence diagrams

## Component NFI: Non-Financial Parameters Management

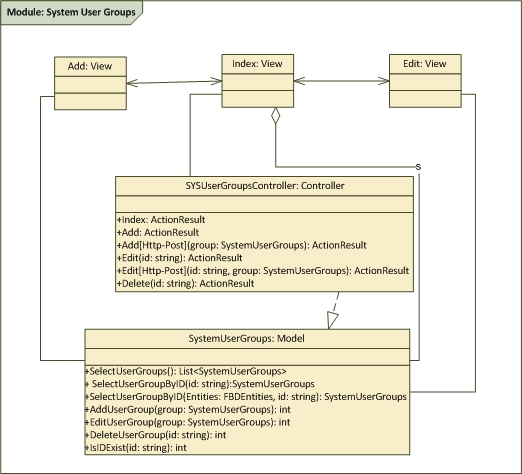
## Component INV : Individual Parameters Management

## Component SYS : System Manager

### MODULE: SYSTEM GROUPS

SRS Mapping: Feature 29

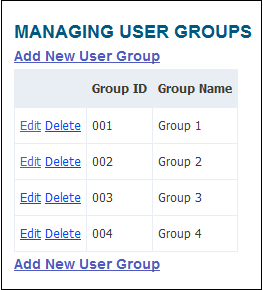
#### Module’s Class diagram



#### Class explanation

##### Class Index: View

###### Screen Image

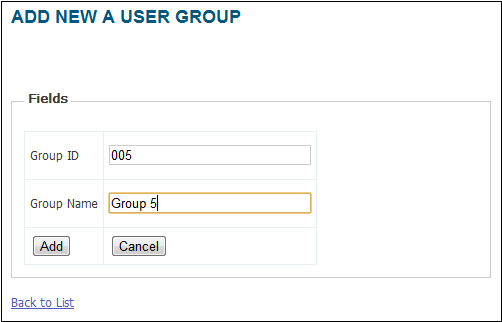


###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: managing user groups |
| Create Group button | Button | The button for creating a user group, clicking this will move to Adding page |
| List of User Groups | Table | List all the groups available in the database, each item will have a button [Edit] and [Delete] Click [Edit] will move to Edit view Click [Delete] will display dialog to verify the deleting the selected group |

##### Class Add: View

###### Screen Image

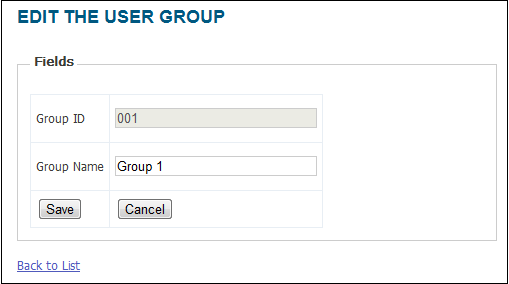


###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: creating user groups |
| lblGroupID | Label | Display "Group ID" |
| lblGroupName | Label | Display "Group Name" |
| txtGroupID | Textbox | Textbox for creating ID of the Group, max length is 10 |
| txtGroupName | Textbox | Textbox for creating Name of the Group, max length is 50 |
| btnAddGroup | Button | After clicking on the button, add a new group to the system with the entered ID and Name then back to the Index View |
| btnCancel | Button | After clicking on the button, back to the Index View |

##### Class Edit: View

###### Screen Image



###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: editing user groups, also display the ID that is being edited |
| lblGroupID | Label | Display "Group ID" |
| lblGroupName | Label | Display "Group Name" |
| txtGroupID | Textbox | Textbox for editting ID of the Group, max length is 10 |
| txtGroupName | Textbox | Textbox for editting Name of the Group, max length is 50 |
| btnSave | Button | After clicking on the button, update the edited group to the DB with ID and Name, if successful then back to the Index View |
| btnCancel | Button | After clicking on the button, back to the Index View |

##### Class SYSUserGroupsController: Controller

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| SYSGroups  Controller | The class is responsible for adding, editing and deleting a group of user | Controller | SYS001 SYS002 SYS003 | System.UserGroups | LongNT |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| METHODS | | | | | |
| NAME | ACCESS LEVEL | INPUT | PROCESS | OUTPUT | EXCEPTION |
| Index | public |  | Use SYSGroupsLogic class to select all the groups (GroupID, GroupName) in the table [System.UserGroups] then display to the [Index] View | ActionResult (moves to [Index] view) |  |
| Add | public |  | Forward to [Add] View | ActionResult (moves to [Add] view) |  |
| Add  [Http-Post] | public | A new Group information | 1. Receive information from parameter 2. Use Logic class to insert new Group 3. Redirect to [Index] View with label displaying: "A new group has been added successfully" | ActionResult (moves to [Index] view) |  |
| Edit | public | ID of the Group editted | 1. Receive ID from parameter 2. Use Logic class to select appropriate Group (GroupID, GroupName) from [System.UserGroups] table 3. Display in [Edit] view | ActionResult (moves to [Edit] view) |  |
| Edit  [Http-Post] | public | ID and the information of the Group editted | 1. Receive ID from parameter2. Use Logic class to update appropriate Group (GroupID, GroupName) with ID selected in [System.UserGroups] table in DB3. Display in [Index] view with label displaying: "The group with ID xyz has been editted successfully" | ActionResult (moves to [Index] view) |  |
| Delete [Http-Post] | public | ID of the Group selected | 1. Receive ID from parameter 2. Use Logic class to delete the Group with selected ID from the [System.UserGroups] table 3. Back to [Index] view with label displaying: "A group has been deleted successfully" | ActionResult (moves to [Index] view) |  |

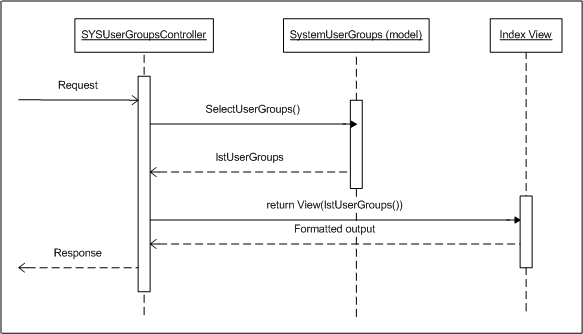
##### Class SystemUserGroups: Model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| SystemUserGroups | Work as DB interface and handle logic process for the controller | Model | SYS001 SYS002 SYS003 | System.UserGroups | LongNT |

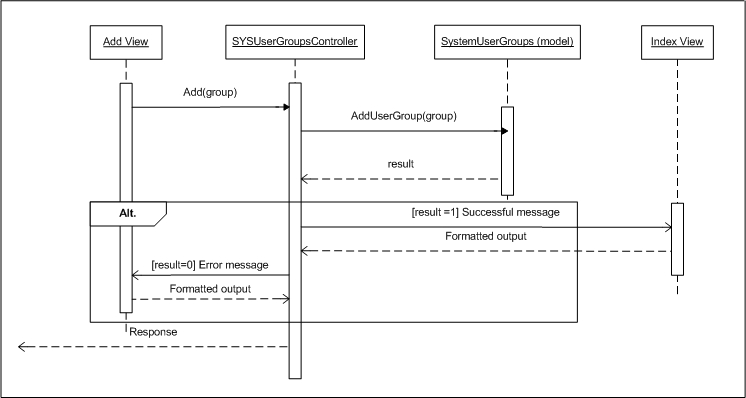
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| METHODS | | | | | |
| NAME | ACCESS LEVEL | INPUT | PROCESS | OUTPUT | EXCEPTION |
| SelectUserGroups | public static |  | Select all the groups (GroupID, GroupName) in the table [System.UserGroups] | List<SystemUserGroups> | Return null |
| SelectUserGroupByID | public static | int ID | Select the group (GroupID, GroupName) in the table [System.UserGroups] with input ID | Model: SystemUserGroups | Return null |
| AddUser  Group | public static | A new Group information | 1. Receive information from parameter 2. Insert new Group into the Database 3. If successful, return 1 otherwise return 0 | int | Return 0 |
| EditUser  Group | public static | ID and the information of the Group editted | 1. Receive ID from parameter 2. Update appropriate Group (GroupID, GroupName) with ID selected in [System.UserGroups] table in DB 3. If successful, return 1 otherwise return 0 | int | Return 0 |
| DeleteUserGroup | public static | ID of the Group selected | 1. Receive ID from parameter 2. Delete the Group with selected ID from the [System.UserGroups] table 3. If successful, return 1 otherwise return 0 | int | Return 0 |
| IsIDExist | public static | string ID | 1. Receive ID from parameter 2. Check for ID dupplication (whether the ID already exists in database or not) 3. IF existed, return 1, otherwise, return 0 |  | Return 2 |

#### Sequence diagrams

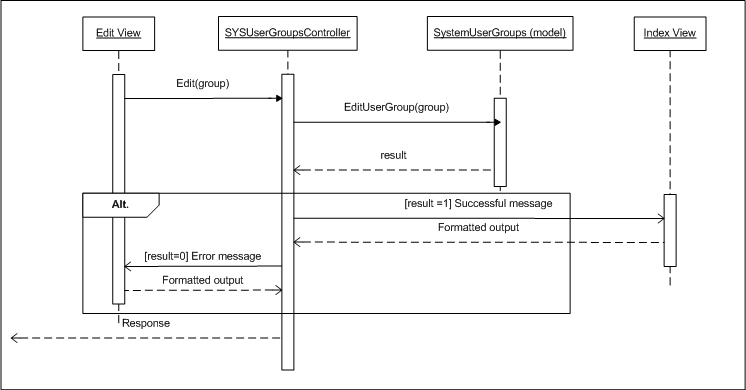
##### View User Group list



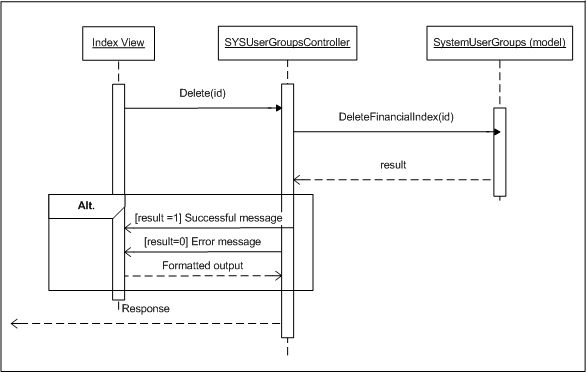
##### Add User Group



##### Edit User Group



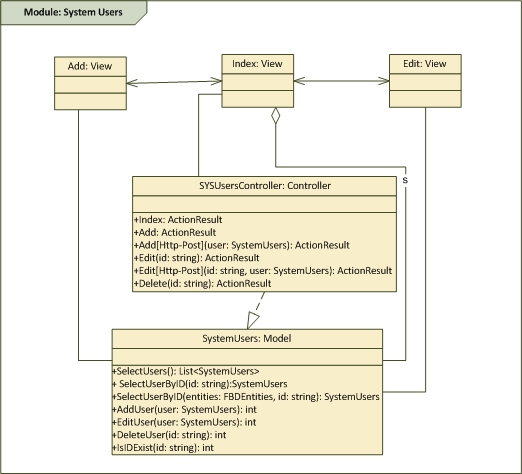
##### Delete UserGroup



### MODULE: SYSTEM USERS

SRS Mapping: Feature 30

#### Module’s Class diagram



#### Class explanation

##### Class Index: View

###### Screen Image

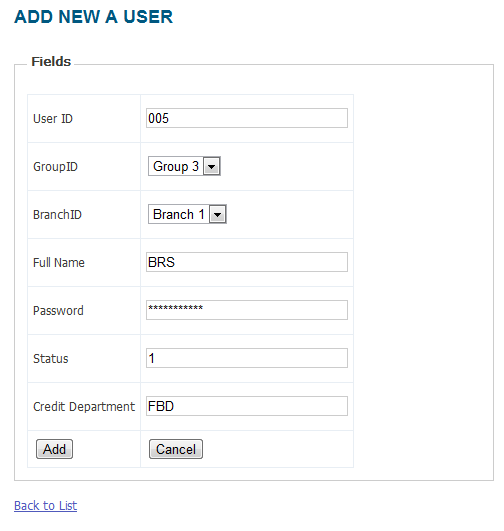


###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: managing users |
| [Add new] Link | Link | The link for creating a user, clicking this will move to Adding page |
| List of Users | Table | List all the users available in the database, each item will have a button [Edit] and [Delete] Click [Edit] will move to Edit view Click [Delete] will display dialog to verify the deleting the selected user |

##### Class Add: View

###### Screen Image

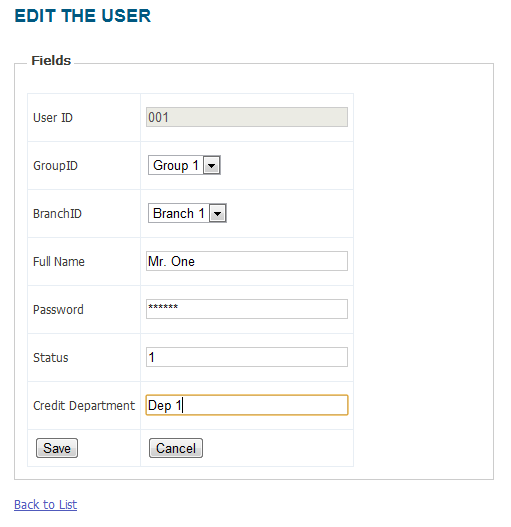


###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: creating user |
| lblUserID | Label | Display "User ID" |
| lblGroup | Label | Display "Group" |
| lblBranch | Label | Display "Branch" |
| lblFullName | Label | Display "Full Name" |
| lblStatus | Label | Display "Status" |
| lblCreditDepartment | Label | Display "Credit Department" |
| txtUserID | Textbox | Textbox for creating User ID, max length is 10 |
| drdGroup | Drop down list | List of Groups available |
| drdBranch | Drop down list | List of Branches available |
| txtFullName | Textbox | Textbox for creating User Name, max length is 50 |
| chkStatus | Checkbox | Enable/Disable user |
| txtCreditDepartment | Textbox | Textbox for creating Credit Department, max length is 50 |
| btnAddUser | Button | After clicking on the button, add a new user to the system with the entered information then back to the Index View |
| btnCancel | Button | After clicking on the button, back to the Index View |

##### Class Edit: View

###### Screen Image



###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: editting user, also displaying user ID of the user editted |
| lblUserID | Label | Display "User ID" |
| lblGroup | Label | Display "Group" |
| lblBranch | Label | Display "Branch" |
| lblFullName | Label | Display "Full Name" |
| lblStatus | Label | Display "Status" |
| lblCreditDepartment | Label | Display "Credit Department" |
| txtUserID | Textbox | Textbox for editting User ID, max length is 10 |
| drdGroup | Drop down list | List of Groups available, value is being displayed is Group of the user |
| drdBranch | Drop down list | List of Branches available, value is being displayed is Branch of the user |
| txtFullName | Textbox | Textbox for editting User Name, max length is 50 |
| chkStatus | Checkbox | Enable/Disable user |
| txtCreditDepartment | Textbox | Textbox for editting Credit Department, max length is 50 |
| btnSave | Button | After clicking on the button, update the user to the system with the entered information then back to the Index View |
| btnCancel | Button | After clicking on the button, back to the Index View |

##### Class SYSUsersController: Controller

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| SYSUsersController | The class is responsible for adding, editing and deleting a User | Controller | SYS004 SYS005 SYS006 | System.Users | LongNT |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| METHODS | | | | | |
| NAME | ACCESS LEVEL | INPUT | PROCESS | OUTPUT | EXCEPTION |
| Index | public |  | Use SYSUsersLogic class to select all the Users (UserID, Group, Branch, FullName, Status, CreditDepartment) in the table [System.Users] then display to the [Index] View | ActionResult (moves to [Index] view) |  |
| Add | public |  | Use Logic class to create ViewModel then Forward to [Add] View | ActionResult (moves to [Add] view) |  |
| Add  [Http-Post] | public | A new User information | 1. Receive information from parameter 2. Use Logic class to insert new User 3. Redirect to [Index] View with label displaying: "A new User has been added successfully" | ActionResult (moves to [Index] view) |  |
| Edit | public | userID of the User editted | 1. Receive ID from parameter 2. Use Logic class to select appropriate User (UserID, Group, Branch, FullName, Status, CreditDepartment) from [System.Users] table and create ViewModel contains list of Groups and Branches 3. Display in [Edit] view | ActionResult (moves to [Edit] view) |  |
| Edit  [Http-Post] | public | userID and the information of the User editted | 1. Receive ID from parameter 2. Use Logic class to update appropriate User (UserID, GroupID, BranchID, FullName, Status, CreditDepartment) with ID selected in [System.Users] table in DB 3. Display in [Index] view with label displaying: "The User with ID xyz has been editted successfully" | ActionResult (moves to [Index] view) |  |
| Delete [Http-Post] | public | userID of the User selected | 1. Receive ID from parameter 2. Use Logic class to delete the User with selected ID from the [System.Users] table 3. Back to [Index] view with label displaying: "A User has been deleted successfully" | ActionResult (moves to [Index] view) |  |

##### Class SystemUsers: Model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| SystemUsers | Work as DB interface and handle logic process for the controller | Model | SYS004 SYS005 SYS006 | System.Users System.UserGroups System.Branches | LongNT |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| METHODS | | | | | |
| NAME | ACCESS LEVEL | INPUT | PROCESS | OUTPUT | EXCEPTION |
| SelectUsers | public static |  | Select all the Users (UserID, Group, Branch, FullName, Status, CreditDepartment) in the table [System.Users] | List<SystemUsers> | Return null |
| SelectUserByID | public static | string userID | 1. Select the User (UserID, Group, Branch, FullName, Status, CreditDepartment) in the table [System.Users] with input ID 2. Use SYSGroupsLogic and SYSBranchesLogic to select all Groups and Branches in DB 3. Create View Model SYSUsersVM from 1 & 2 | ViewModel: SYSUsersVM | Return null |
| AddUser | public static | A new User information | 1. Receive information from parameter 2. Insert new User into the Database and by default, Password field is the same as userID 3. If successful, return 1 otherwise return 0 | int | Return 0 |
| EditUser | public static | userID and the information of the User editted | 1. Receive ID from parameter 2. Update appropriate User (UserID, Group, Branch, FullName, Status, CreditDepartment) with ID selected in [System.Users] table in DB 3. If successful, return 1 otherwise return 0 | int | Return 0 |
| DeleteUser | public static | userID of the User selected | 1. Receive ID from parameter 2. Delete the User with selected ID from the [System.Users] table 3. If successful, return 1 otherwise return 0 | int | Return 0 |
| IsIDExist | public static | string ID | 1. Receive ID from parameter 2. Check for ID dupplication (whether the ID already exists in database or not) 3. IF existed, return 1, otherwise, return 0 |  | Return 2 |

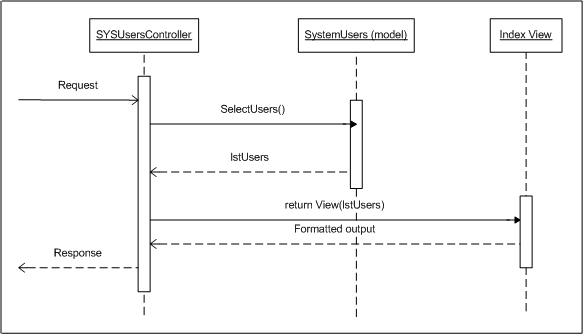
##### Class SystemUsersViewModel: View Model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| SYSUsersVM | View Model used for Add and Edit view | View Model | SYS004 SYS005 SYS006 | System.Users System.UserGroups System.Branches | LongNT |

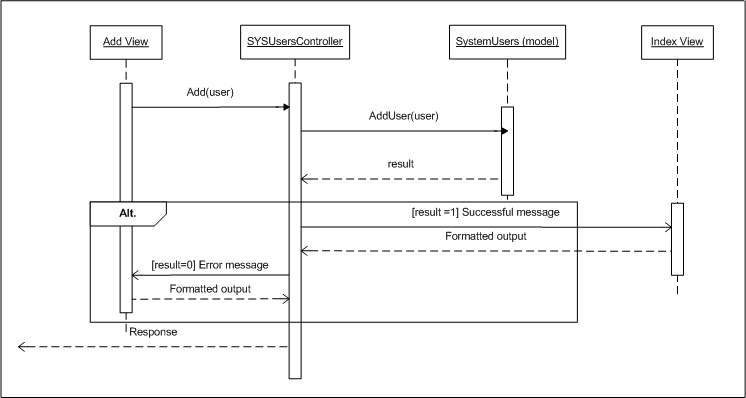
|  |  |  |
| --- | --- | --- |
| PROPERTIES | | |
| NAME | DATA TYPE | INITIAL SETTING |
| SystemUsers | SystemUsers |  |
| SystemUserGroups | List<SystemUserGroups> |  |
| SystemBranches | List<SystemBranches> |  |

#### Sequence diagrams

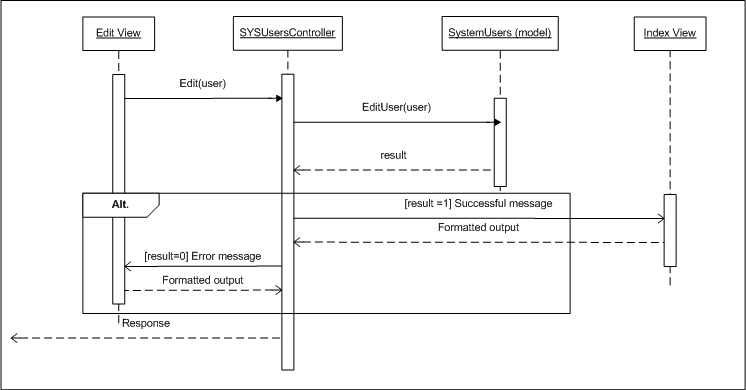
##### View User list



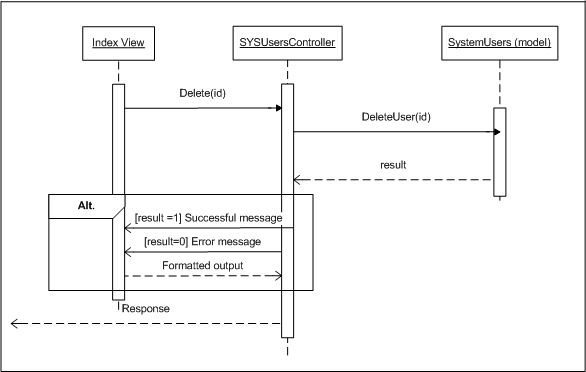
##### Add User



##### Edit User



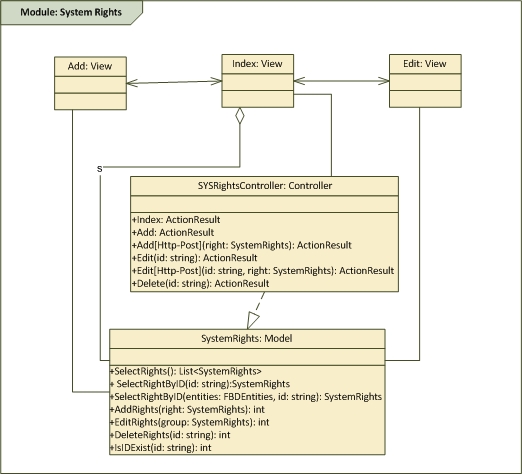
##### Delete User



### MODULE: SYSTEM RIGHTS

SRS Mapping: Feature 31

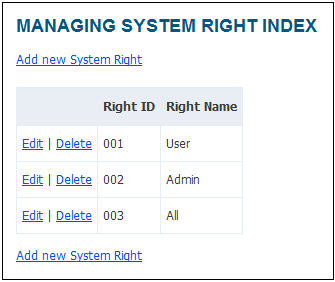
#### Module’s Class diagram



#### Class explanation

##### Class Index: View

###### Screen Image

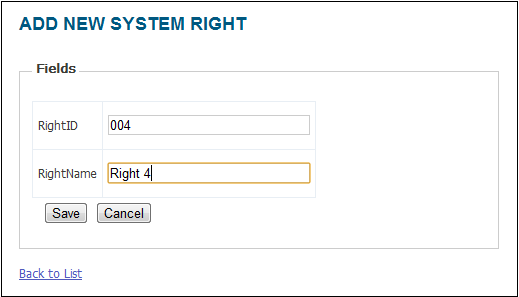


###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: managing rights |
| [Add new] link | link | The link for creating a right, clicking this will move to Adding page |
| List of rights | Table | List all the rights available in the database, each item will have a button [Edit] and [Delete] Click [Edit] will move to Edit view Click [Delete] will display dialog to verify the deleting the selected right |

##### Class Add: View

###### Screen Image

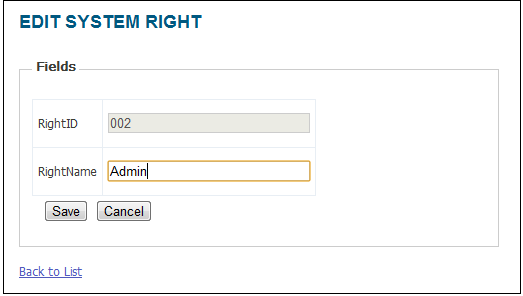


###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: creating Rights |
| lblRightID | Label | Display "Right ID" |
| lblRight | Label | Display "Right" |
| txtRightID | Textbox | Textbox for creating ID of the Right, max length is 20 |
| txtRight | Textbox | Textbox for creating Name of the Right, max length is 255 |
| btnAddRight | Button | After clicking on the button, add a new Right to the system with the entered ID and Name then back to the Index View |
| btnCancel | Button | After clicking on the button, back to the Index View |

##### Class Edit: View

###### Screen Image



###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: editting Rights, also displaying the ID of the Right editted |
| lblRightID | Label | Display "Right ID" |
| lblRight | Label | Display "Right" |
| txtRightID | Textbox | Textbox for creating ID of the Right, max length is 20 |
| txtRight | Textbox | Textbox for creating Name of the Right, max length is 255 |
| btnSave | Button | After clicking on the button, update the Right to the system with the entered ID and Name then back to the Index View |
| btnCancel | Button | After clicking on the button, back to the Index View |

##### Class SYSRightsController: Controller

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| SYSRightsController | The class is responsible for adding, editing and deleting a Right of system | Controller | SYS007 SYS008 SYS009 | System.Rights | LongNT |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| METHODS | | | | | |
| NAME | ACCESS LEVEL | INPUT | PROCESS | OUTPUT | EXCEPTION |
| Index | public |  | Use SYSRightsLogic class to select all the Rights (RightID, Right) in the table [System.Rights] then display to the [Index] View | ActionResult (moves to [Index] view) |  |
| Add | public |  | Forward to [Add] View | ActionResult (moves to [Add] view) |  |
| Add  [Http-Post] | public | A new Right information | 1. Receive information from parameter 2. Use Logic class to insert new Right 3. Redirect to [Index] View with label displaying: "A new Right has been added successfully" | ActionResult (moves to [Index] view) |  |
| Edit | public | ID of the Right editted | 1. Receive ID from parameter 2. Use Logic class to select appropriate Right (RightID, Right) from [System.Rights] table 3. Display in [Edit] view | ActionResult (moves to [Edit] view) |  |
| Edit  [Http-Post] | public | ID and the information of the Right editted | 1. Receive ID from parameter 2. Use Logic class to update appropriate Right (RightID, Right) with ID selected in [System.Rights] table in DB 3. Display in [Index] view with label displaying: "The Right with ID xyz has been editted successfully" | ActionResult (moves to [Index] view) |  |
| Delete [Http-Post] | public | ID of the Right selected | 1. Receive ID from parameter 2. Use Logic class to delete the Right with selected ID from the [System.Rights] table 3. Back to [Index] view with label displaying: "A Right has been deleted successfully" | ActionResult (moves to [Index] view) |  |

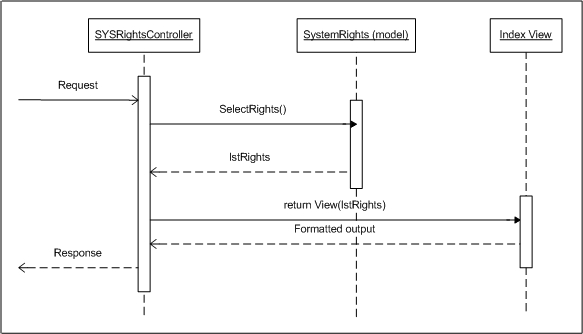
##### Class SystemRights: Model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| SYSRightsLogic | Work as DB interface and handle logic process for the controller | Model | SYS007 SYS008 SYS009 | System.Rights | LongNT |

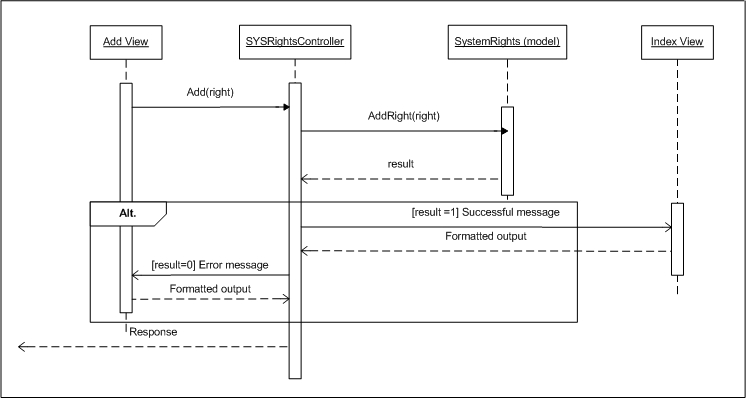
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| METHODS | | | | | |
| NAME | ACCESS LEVEL | INPUT | PROCESS | OUTPUT | EXCEPTION |
| SelectRights | public static |  | Select all the Rights (RightID, Right) in the table [System.Rights] | List<SystemRights> | Return null |
| SelectRightByID | public static | int ID | Select the Right (RightID, Right) in the table [System.Rights] with input ID | Model: SystemRights | Return null |
| AddRight | public static | A new Right information | 1. Receive information from parameter 2. Insert new Right into the Database 3. If successful, return 1 otherwise return 0 | int | Return 0 |
| EditRight | public static | ID and the information of the Right editted | 1. Receive ID from parameter 2. Update appropriate Right (RightID, Right) with ID selected in [System.Rights] table in DB 3. If successful, return 1 otherwise return 0 | int | Return 0 |
| DeleteRight | public static | ID of the Right selected | 1. Receive ID from parameter 2. Delete the Right with selected ID from the [System.Rights] table 3. If successful, return 1 otherwise return 0 | int | Return 0 |
| IsIDExist | public static | string ID | 1. Receive ID from parameter 2. Check for ID dupplication (whether the ID already exists in database or not) 3. IF existed, return 1, otherwise, return 0 |  | Return 2 |

#### Sequence diagrams

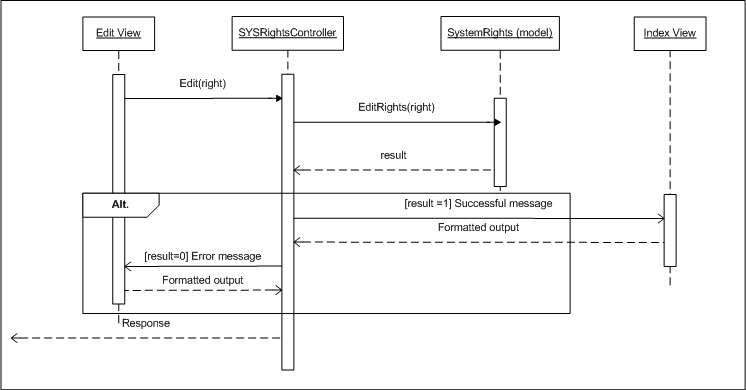
##### View Right list



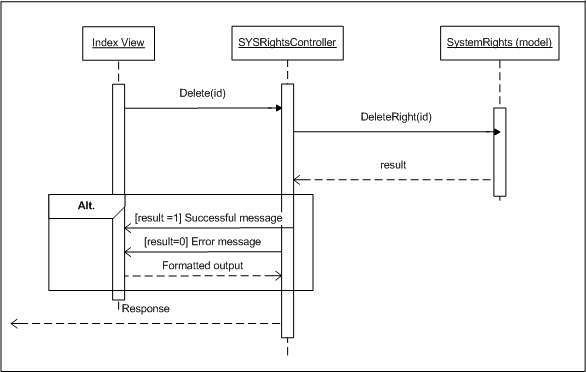
##### Add Right



##### Edit Right



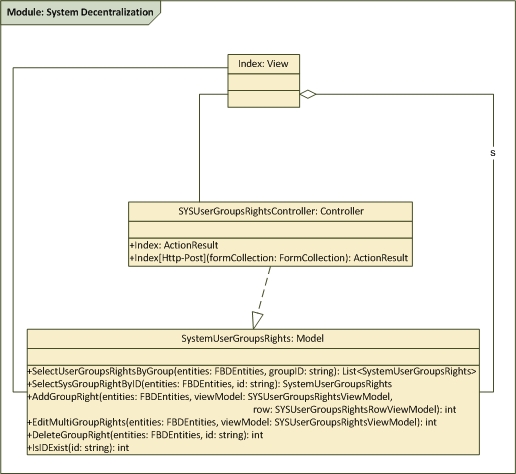
##### Delete Right



### MODULE: SYSTEM DECENTRALIZATION

SRS Mapping: Feature 32

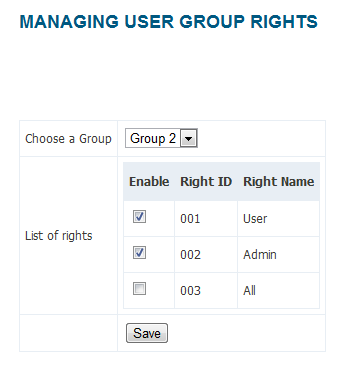
#### Module’s Class diagram



#### Class explanation

##### Class Index: View

###### Screen Image



###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: users decentralization |
| lblGroups | Label | Display "User Groups" |
| lblRights | Label | Display "System's Rights" |
| drdGroups | Drop down list | List all the Groups available in the system |
| list of Rights | Table | List all the Ritghts available with the selected Groups. Each item will have a checkbox to enable/disable the right of the group |
| chkConfirm | Checkbox | Confirm the changes |
| btnSave | Button | After clicking on the button and click checkbox, update the changes to the system |

##### Class SYSUserGroupsRightsController: Controller

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| SYSUserGroupsRightController | The class is responsible for editting user group's rights | Controller | SYS010 | System.UserGroupsRights System.UserGroups System.Rights | LongNT |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| METHODS | | | | | |
| NAME | ACCESS LEVEL | INPUT | PROCESS | OUTPUT | EXCEPTION |
| Edit | public |  | Use SYSRightsLogic class to create a View Model containing the Groups and Rights in DB then display to the [Edit] View | ActionResult (moves to [Edit] view) |  |
| Edit  [Http-Post] | public | groupID and the Rights information of the group | 1. Receive parameter 2. Use Logic class to update appropriate Rights of group selected in [System.UserGroupsRights] table in DB 3. Display in [Index] view with label displaying: "The Rights with group xyz has been editted successfully" | ActionResult (moves to [Edit] view) |  |

##### Class SystemUserGroupsRights: Model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| SystemUserGroupsRights | Work as DB interface and handle logic process for the controller | Model | SYS010 | System.UserGroupsRights System.UserGroups System.Rights | LongNT |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| METHODS | | | | | |
| NAME | ACCESS LEVEL | INPUT | PROCESS | OUTPUT | EXCEPTION |
| SelectGroups  Rights | public static |  | Select all the Groups (GroupID, GroupName) and all Rights (RightID, Right) in DB then store in ViewModel | ViewModel: SYSDecentralizationVM | Return null |
| AddGroupRight | private | GroupID and RightID | 1. Receive information from parameter 2. Insert new GroupRight into the table [System.UserGroupsRights] 3. If successful, return 1 otherwise return 0 | int | Return 0 |
| EditGroupRight | public static | groupID and the Rights information of the group | 1. Receive parameter 2. With the GroupID inputted, iterate the Rights list of the group selected 3. If the right is enable, add this to [System.UserGroupsRights], otherwise delete it 4. If successful, return 1 otherwise return 0 | int | Return 0 |
| DeleteGroupRight | private | GroupID and RightID | 1. Receive ID from parameter 2. Delete the GroupRight with inputted ID from the [System.UserGroupsRights] table 3. If successful, return 1 otherwise return 0 | int | Return 0 |

##### Class SystemUserGroupsRightsViewModel: ViewModel

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| SystemUserGroupsRightsViewModel | View Model used for Index view | View Model | SYS010 | System.UserGroupsRights System.UserGroups System.Rights | LongNT |

|  |  |  |
| --- | --- | --- |
| PROPERTIES | | |
| NAME | DATA TYPE | INITIAL SETTING |
| SystemUserGroups | List<SystemUserGroups> |  |
| SystemRights | List<SystemRights> |  |

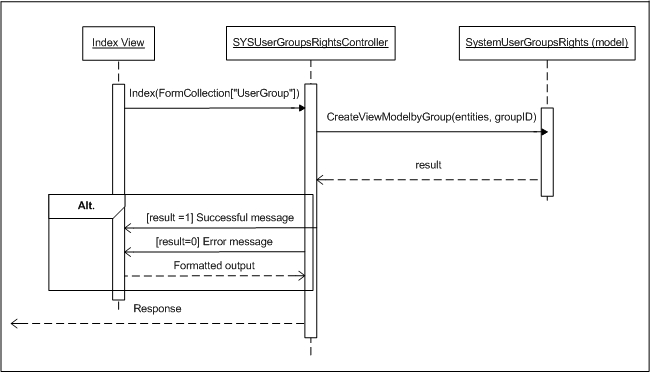
##### Class SystemUserGroupsRightsRowViewModel: ViewModel

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| SystemUserGroupsRightRowViewModel | View Model used for Index view | View Model | SYS010 | System.UserGroupsRights System.UserGroups System.Rights | LongNT |

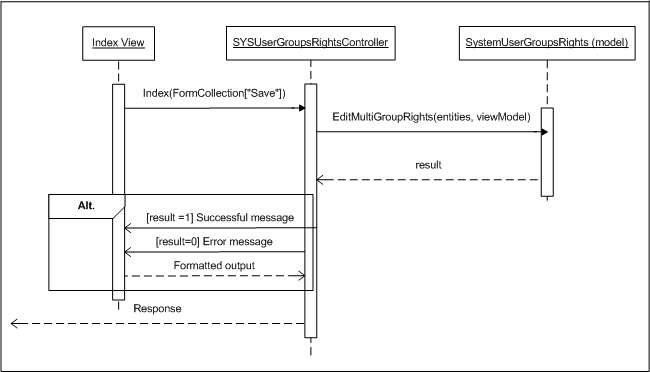
|  |  |  |
| --- | --- | --- |
| PROPERTIES | | |
| NAME | DATA TYPE | INITIAL SETTING |
| Checked | bool | FALSE |
| GroupRightID | int | -1 |
| RIghtID | string |  |
| RightName | String |  |

#### Sequence diagrams

##### View Right list filtered by Group



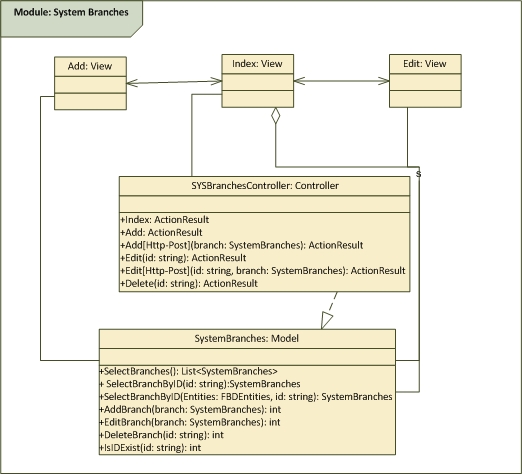
##### Edit Right list



### MODULE: SYSTEM BRANCHES

SRS Mapping: Feature 33

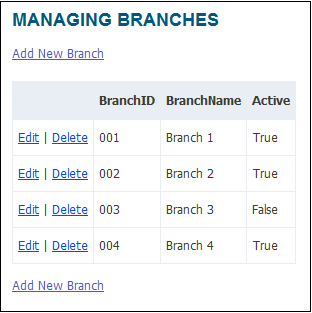
#### Module’s Class diagram



#### Class explanation

##### Class Index: View

###### Screen Image

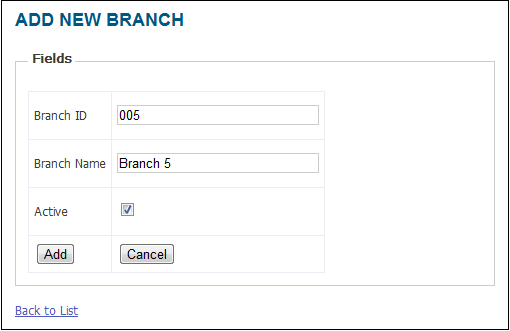


###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: managing branches |
| Create Branch button | Button | The button for creating a branch, clicking this will move to Adding page |
| List of Branches | Table | List all the branches available in the database, each item will have a button [Edit] and [Delete] Click [Edit] will move to Edit view Click [Delete] will display dialog to verify the deleting the selected branch |

##### Class Add: View

###### Screen Image

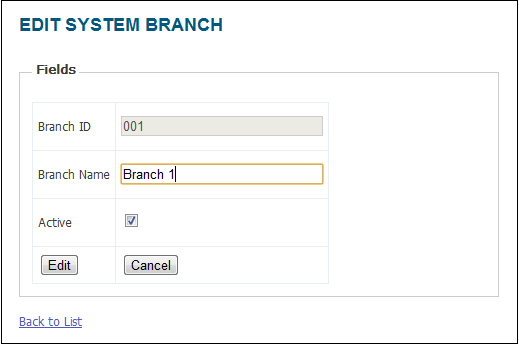


###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: creating branches |
| lblBranchID | Label | Display "Branch ID" |
| lblBranchName | Label | Display "Branch Name" |
| lblActive | Label | Display "Active now" |
| txtBranchID | Textbox | Textbox for creating ID of the Branch, max length is 10 |
| txtBranchName | Textbox | Textbox for creating Name of the Branch, max length is 255 |
| chkActive | Checkbox | Check box for enable/disable the Branch |
| btnAddBranch | Button | After clicking on the button, add a new branch to the system with the entered ID, Name and active status then back to the Index View |
| btnCancel | Button | After clicking on the button, back to the Index View |

##### Class Edit: View

###### Screen Image



###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: editting branch, also display the ID that is being edited |
| lblBranchID | Label | Display "Branch ID" |
| lblBranchName | Label | Display "Branch Name" |
| lblActive | Label | Display "Active now" |
| txtBranchID | Textbox | Textbox for editting ID of the Branch, max length is 10 |
| txtBranchName | Textbox | Textbox for editting Name of the Branch, max length is 255 |
| chkActive | Checkbox | Check box for enable/disable the Branch |
| btnSave | Button | After clicking on the button, the branch is updated to the system with the entered ID, Name and active status then back to the Index View |
| btnCancel | Button | After clicking on the button, back to the Index View |

##### Class SYSBranchesController: Controller

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| SYSBranchesController | The class is responsible for adding, editing and deleting a Branch | Controller | SYS013 SYS014 SYS015 | System.Branches | LongNT |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| METHODS | | | | | |
| NAME | ACCESS LEVEL | INPUT | PROCESS | OUTPUT | EXCEPTION |
| Index | public |  | Use SYSBranchesLogic class to select all the Branches (BranchID, BranchName, Active) in the table [System.Branches] then display to the [Index] View | ActionResult (moves to [Index] view) |  |
| Add | public |  | Forward to [Add] View | ActionResult (moves to [Add] view) |  |
| Add  [Http-Post] | public | A new Branch information | 1. Receive information from parameter 2. Use Logic class to insert new Branch 3. Redirect to [Index] View with label displaying: "A new Branch has been added successfully" | ActionResult (moves to [Index] view) |  |
| Edit | public | ID of the Branch editted | 1. Receive ID from parameter 2. Use Logic class to select appropriate Branch (BranchID, BranchName, Active) from [System.Branches] table 3. Display in [Edit] view | ActionResult (moves to [Edit] view) |  |
| Edit  [Http-Post] | public | ID and the information of the Branch editted | 1. Receive ID from parameter 2. Use Logic class to update appropriate Branch (BranchID, BranchName, Active) with ID selected in [System.Branches] table in DB 3. Display in [Index] view with label displaying: "The Branch with ID xyz has been editted successfully" | ActionResult (moves to [Index] view) |  |
| Delete [Http-Post] | public | ID of the Branch selected | 1. Receive ID from parameter 2. Use Logic class to delete the Branch with selected ID from the [System.Branches] table 3. Back to [Index] view with label displaying: "A Branch has been deleted successfully" | ActionResult (moves to [Index] view) |  |

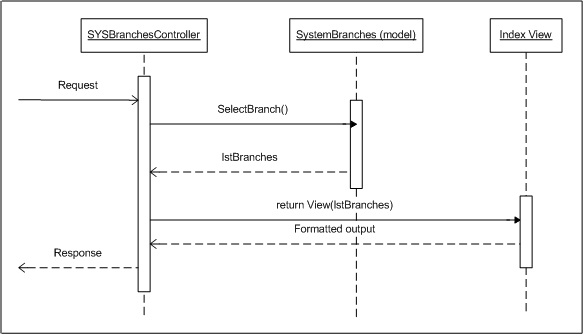
##### Class SystemBranches: Model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| SystemBranches | Work as DB interface and handle logic process for the controller | Model | SYS013 SYS014 SYS015 | System.Branches | LongNT |

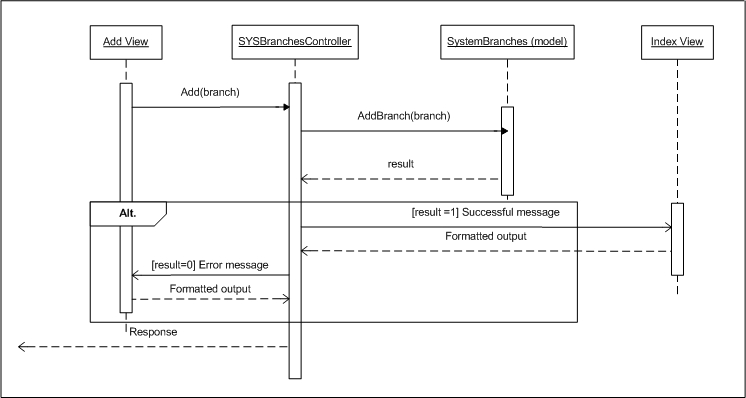
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| METHODS | | | | | |
| NAME | ACCESS LEVEL | INPUT | PROCESS | OUTPUT | EXCEPTION |
| Select  Branches | public static |  | Select all the Branches (BranchID, BranchName, Active) in the table [System.Branches] | List<System  Branches> | Return null |
| SelectBranchByID | public static | int ID | Select the Branch (BranchID, BranchName, Active) in the table [System.Branches] with input ID | Model: System  Branches | Return null |
| AddBranch | public static | A new Branch information | 1. Receive information from parameter 2. Insert new Branch into the Database 3. If successful, return 1 otherwise return 0 | int | Return 0 |
| EditBranch | public static | ID and the information of the Branch editted | 1. Receive ID from parameter 2. Update appropriate Branch (BranchID, BranchName, Active) with ID selected in [System.Branches] table in DB 3. If successful, return 1 otherwise return 0 | int | Return 0 |
| DeleteBranch | public static | ID of the Branch selected | 1. Receive ID from parameter 2. Delete the Branch with selected ID from the [System.Branches] table 3. If successful, return 1 otherwise return 0 | int | Return 0 |
| IsIDExist | public static | string ID | 1. Receive ID from parameter 2. Check for ID dupplication (whether the ID already exists in database or not) 3. IF existed, return 1, otherwise, return 0 |  | Return 2 |

#### Sequence diagrams

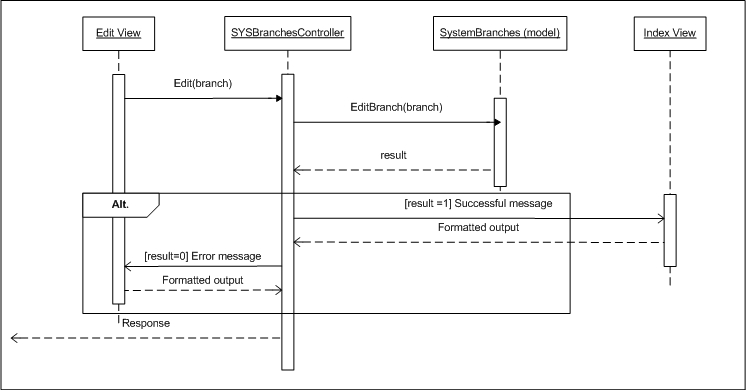
##### View Branch list



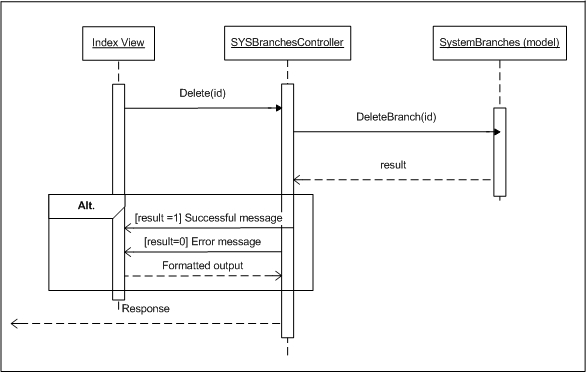
##### Add Branch



##### Edit Branch



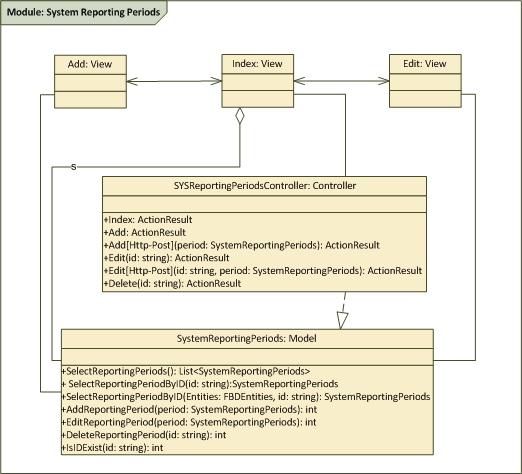
##### Delete Branch



### MODULE: SYSTEM REPORTING PERIODS

SRS Mapping: Feature 34

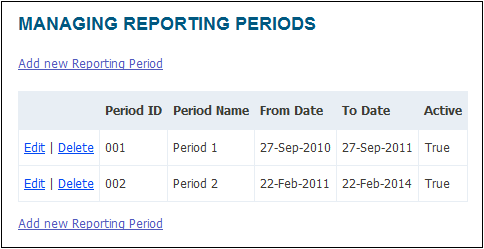
#### Module’s Class diagram



#### Class explanation

##### Class Index: View

###### Screen Image

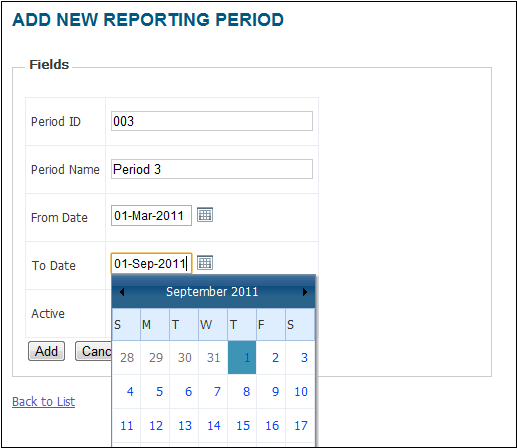


###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: managing reporting periods |
| [Add new] link | Link | The link for creating a reporting period, clicking this will move to Adding page |
| List of reporting periods | Table | List all the periods available in the database, each item will have a button [Edit] and [Delete] Click [Edit] will move to Edit view Click [Delete] will display dialog to verify the deleting the selected period |
| [Add new] link | Link | The link for creating a reporting period, clicking this will move to Adding page |

##### Class Add: View

###### Screen Image

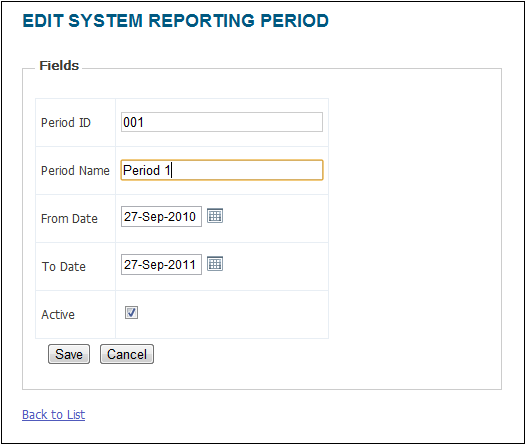


###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: creating reporting period |
| lblPeriodID | Label | Display "Period ID" |
| lblPeriodName | Label | Display "Period Name" |
| lblFromDate | Label | Display "From Date" |
| lblToDate | Label | Display "To Date" |
| lblActive | Label | Display "Active" |
| txtPeriodID | Textbox | Textbox for creating ID of the Period, max length is 10 |
| txtPeriodName | Textbox | Textbox for creating Name of the Period, max length is 50 |
| cldFromDate | Calendar | Choose date |
| cldToDate | Calendar | Choose date, ToDate must be later than FromDate |
| chkActive | Checkbox | Enable/Disable the reporting period |
| btnAdd | Button | After clicking, add a new Period to the system with the entered information then back to the Index View |
| btnCancel | Button | After clicking on the button, back to the Index View |

##### Class Edit: View

###### Screen Image



###### Screen Description

|  |  |  |
| --- | --- | --- |
| COMPONENT NAME | COMPONENT TYPE | DESCRIPTION |
| Page name | Label | The text describes the responsibility of the page: editting reporting period, also displaying the ID of the reporting period editted |
| lblPeriodID | Label | Display "Period ID" |
| lblPeriodName | Label | Display "Period Name" |
| lblFromDate | Label | Display "From Date" |
| lblToDate | Label | Display "To Date" |
| lblActive | Label | Display "Active" |
| txtPeriodID | Textbox | Textbox for editing ID of the Period, max length is 10 |
| txtPeriodName | Textbox | Textbox for editing Name of the Period, max length is 50 |
| cldFromDate | Calendar | Choose date |
| cldToDate | Calendar | Choose date, ToDate must be later than FromDate |
| chkActive | Checkbox | Enable/Disable the reporting period |
| btnSave | Button | After clicking, update the Period to the system with the entered information then back to the Index View |
| btnCancel | Button | After clicking on the button, back to the Index View |

##### Class SYSReportingPeriodController: Controller

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| SYSReportingPeriodsController | The class is responsible for adding, editing and deleting a period | Controller | SYS016 SYS017 SYS018 | System.ReportingPeriods | LongNT |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| METHODS | | | | | |
| NAME | ACCESS LEVEL | INPUT | PROCESS | OUTPUT | EXCEPTION |
| Index | public |  | Use SYSReportingPeriodsLogic class to select all the periods (periodID, periodName, fromDate, toDate, active) in the table [System.ReportingPeriods] then display to the [Index] View | ActionResult (moves to [Index] view) |  |
| Add | public |  | Forward to [Add] View | ActionResult (moves to [Add] view) |  |
| Add  [Http-Post] | public | A new period information | 1. Receive information from parameter 2. Use Logic class to insert new period 3. Redirect to [Index] View with label displaying: "A new period has been added successfully" | ActionResult (moves to [Index] view) |  |
| Edit | public | ID of the period editted | 1. Receive ID from parameter 2. Use Logic class to select appropriate period (periodID, periodName, fromDate, toDate, Active) from [System.ReportingPeriods] table 3. Display in [Edit] view | ActionResult (moves to [Edit] view) |  |
| Edit  [Http-Post] | public | ID and the information of the period editted | 1. Receive ID from parameter 2. Use Logic class to update appropriate period (periodID, periodName, fromDate, toDate, Active) with ID selected in [System.ReportingPeriods] table in DB 3. Display in [Index] view with label displaying: "PeriodID xyz has been editted successfully" | ActionResult (moves to [Index] view) |  |
| Delete [Http-Post] | public | ID of the period selected | 1. Receive ID from parameter 2. Use Logic class to delete the period with selected ID from the [System.ReportingPeriods] table 3. Back to [Index] view with label displaying: "PeriodID xyz has been deleted successfully" | ActionResult (moves to [Index] view) |  |

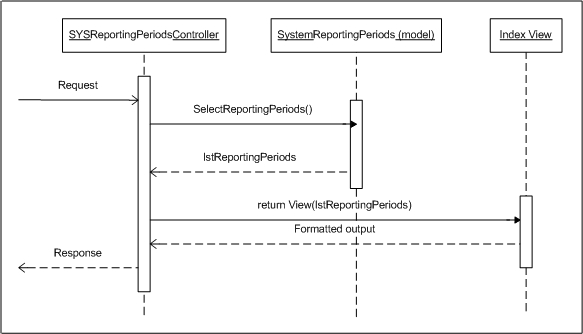
##### Class SystemReportingPeriod: Model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GENERAL INFORMATION | | | | | |
| CLASS NAME | DESCRIPTION | INHERITANCE | REQUIREMENT ID | RELATED DB TABLE | MAKER |
| SYSReportingPeriodsLogic | Work as DB interface and handle logic process for the controller | Model | SYS016 SYS017 SYS018 | System.ReportingPeriods | LongNT |

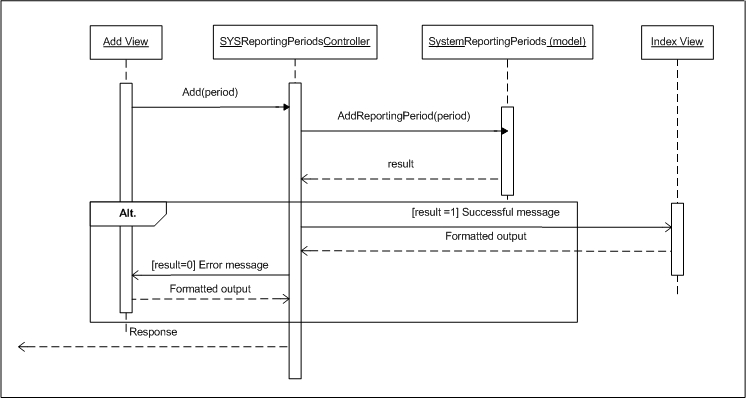
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| METHODS | | | | | |
| NAME | ACCESS LEVEL | INPUT | PROCESS | OUTPUT | EXCEPTION |
| SelectReportingPeriods | public static |  | Select all the periods (periodID, periodName, fromDate, toDate, active) in the table [System.ReportingPeriods] | List<SystemReportingPeriods> | Return null |
| SelectReportingPeriodByID | public static | string ID | Select the period (periodID, periodName, fromDate, toDate, active) in the table [System.ReportingPeriods] with input ID | Model: SystemReportingPeriods | Return null |
| SelectReportingPeriodByID | public static | string ID, FBDEntities | Select the period (periodID, periodName, fromDate, toDate, active) in the table [System.ReportingPeriods] with input ID | Model: SystemReportingPeriods | Return null |
| AddReportingPeriod | public static | A new period information | 1. Receive information from parameter 2. Insert new period into the Database 3. If successful, return 1 otherwise return 0 |  | Return 0 |
| EditReportingPeriod | public static | ID and the information of the period editted | 1. Receive ID from parameter 2. Update appropriate period (periodID, periodName, fromDate, toDate, active) with ID selected in [System.ReportingPeriods] table in DB 3. If successful, return 1 otherwise return 0 |  | Return 0 |
| DeleteReportingPeriod | public static | ID of the period selected | 1. Receive ID from parameter 2. Delete the period with selected ID from the [System.ReportingPeriods] table 3. If successful, return 1 otherwise return 0 |  | Return 0 |
| IsIDExist | public static | string ID | 1. Receive ID from parameter 2. Check for ID dupplication (whether the ID already exists in database or not) 3. IF existed, return 1, otherwise, return 0 |  | Return 2 |

#### Sequence diagrams

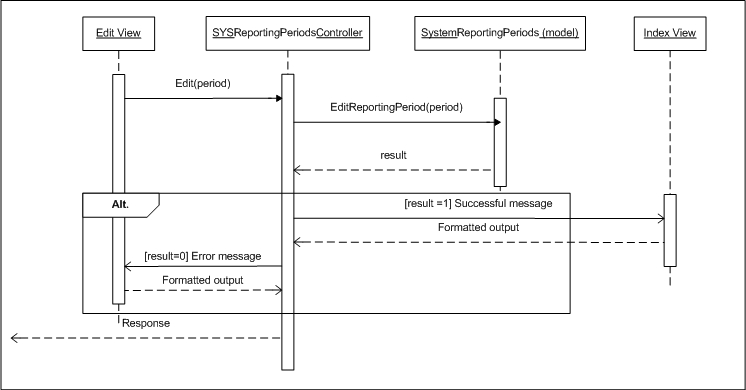
##### View Period list



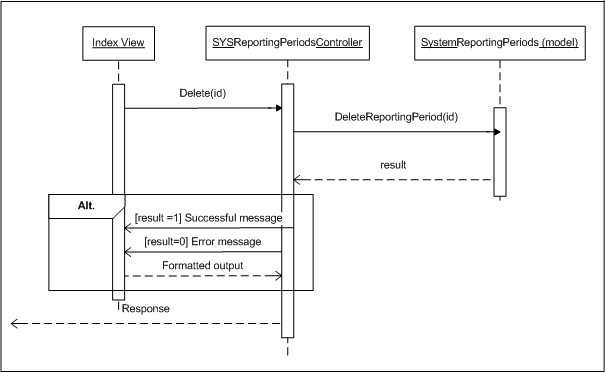
##### Add Period



##### Edit Period



##### Delete Period



## Component RNK : Ranking Manager

## Component RPT : Report Handler

# DATABASE DESIGN