|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | **Logo_FPT_University_doc** | **MINISTRY OF EDUCATION AND TRAINING** |   **FPT UNIVERSITY**   |  | | --- | | Capstone Project | |
|  |
| Smart Menu Solution  Report 4: Software Design Description |
| |  |  | | --- | --- | | **SMS TEAM** | | | **Group Members** |  | | **Supervisor** |  | | **Ext Supervisor** |  | | **Capstone Project code** |  | |
|  |

- Hanoi, 10/2012 -

Contents

[I. Design Overview 4](#_Toc338487079)

[II. System Architectural Design 4](#_Toc338487080)

[1. Overall System Architecture 4](#_Toc338487081)

[2. Web Application Architecture 6](#_Toc338487082)

[2.1 CRS.Common Component 6](#_Toc338487083)

[2.2 CRS.Resources Component 6](#_Toc338487084)

[2.3 CRS.Business Component 7](#_Toc338487085)

[2.4 CRS.Web Component 7](#_Toc338487086)

[III. Component diagram 8](#_Toc338487087)

[IV. Detailed Description of Component 9](#_Toc338487088)

[1. Comment 9](#_Toc338487089)

[1.1 Class Diagram 9](#_Toc338487090)

[1.2 Class Explanations 10](#_Toc338487091)

[1.3 Sequence Diagrams 10](#_Toc338487092)

[2. Search Content 13](#_Toc338487093)

[2.1 Class Diagram 14](#_Toc338487094)

[2.2 Class Explanations 15](#_Toc338487095)

[2.3 Sequence Diagrams 15](#_Toc338487096)

[3. Statistics 16](#_Toc338487097)

[3.1 Class Diagram 16](#_Toc338487098)

[3.2 Class Explanations 17](#_Toc338487099)

[3.3 Sequence Diagrams 17](#_Toc338487100)

[4. Question 19](#_Toc338487101)

[4.1. Class Diagram 20](#_Toc338487102)

[4.2. Class Explanations 21](#_Toc338487103)

[4.3. Sequence Diagrams 21](#_Toc338487104)

[5. Answer 27](#_Toc338487105)

[5.1 Class Diagram 27](#_Toc338487106)

[5.2 Class Explanations 29](#_Toc338487107)

[5.3 Sequence Diagrams 29](#_Toc338487108)

[6. Favorite 33](#_Toc338487109)

[6.1 Class Diagram 33](#_Toc338487110)

[6.2 Class Explanations 34](#_Toc338487111)

[6.3 Sequence Diagrams 34](#_Toc338487112)

[7. Manage Categories 36](#_Toc338487113)

[7.1 Class Diagram 37](#_Toc338487114)

[7.2 Class Explanations 38](#_Toc338487115)

[7.3 Sequence Diagrams 38](#_Toc338487116)

[8. Manage Spam Contents 41](#_Toc338487117)

[8.1 Class Diagram 42](#_Toc338487118)

[8.2 Class Explanations 42](#_Toc338487119)

[8.3 Sequence Diagrams 43](#_Toc338487120)

[9. Manage Spam Comments 45](#_Toc338487121)

[9.1 Class Diagram 46](#_Toc338487122)

[9.2 Class Explanations 47](#_Toc338487123)

[9.3 Sequence Diagrams 47](#_Toc338487124)

[10. Manage Spam Answers 49](#_Toc338487125)

[10.1 Class Diagram 50](#_Toc338487126)

[10.2 Class Explanations 51](#_Toc338487127)

[10.3 Sequence Diagrams 51](#_Toc338487128)

[11. Login/Logout/Register 53](#_Toc338487129)

[11.1 Class Diagram 54](#_Toc338487130)

[11.2 Class Explanations 55](#_Toc338487131)

[11.3 Sequence Diagrams 55](#_Toc338487132)

[12. Forgot Password 57](#_Toc338487133)

[12.1 Class Diagram 57](#_Toc338487134)

[12.2 Class Explanations 58](#_Toc338487135)

[12.3 Sequence Diagrams 58](#_Toc338487136)

[13. Change Password 59](#_Toc338487137)

[13.1 Class Diagram 59](#_Toc338487138)

[13.2 Class Explanations 60](#_Toc338487139)

[13.3 Sequence Diagrams 60](#_Toc338487140)

[14. Update User Profile 60](#_Toc338487141)

[14.1 Class Diagram 61](#_Toc338487142)

[14.2 Class Explanations 62](#_Toc338487143)

[14.3 Sequence Diagrams 62](#_Toc338487144)

[15. View other user's profile 62](#_Toc338487145)

[15.1 Class Diagram 63](#_Toc338487146)

[15.2 Class Explanations 64](#_Toc338487147)

[15.3 Sequence Diagrams 64](#_Toc338487148)

[16. Manager Users 64](#_Toc338487149)

[16.1 Class Diagram 65](#_Toc338487166)

[16.2 Class Explanations 65](#_Toc338487167)

[16.3 Sequence Diagrams 66](#_Toc338487168)

[17. Content 68](#_Toc338487169)

[17.1 Class Diagram 68](#_Toc338487170)

[17.2 Class Explanations 69](#_Toc338487171)

[17.3 Sequence Diagrams 70](#_Toc338487172)

[18. Manage Spam Questions 75](#_Toc338487173)

[18.1 Class Diagram 75](#_Toc338487174)

[18.2 Class Explanations 75](#_Toc338487175)

[18.3 Sequence Diagrams 76](#_Toc338487176)

[V. User Interface Design 78](#_Toc338487177)

[1. Screen Images 78](#_Toc338487178)

[2. Objects and Actions 78](#_Toc338487179)

[VI. Database Design 79](#_Toc338487180)

[1. ERDs 79](#_Toc338487181)

[2. Tables 79](#_Toc338487182)

[VII. References 79](#_Toc338487183)

# Design Overview

***The CMS Elements :***

* **Content Management Application (CMA)**

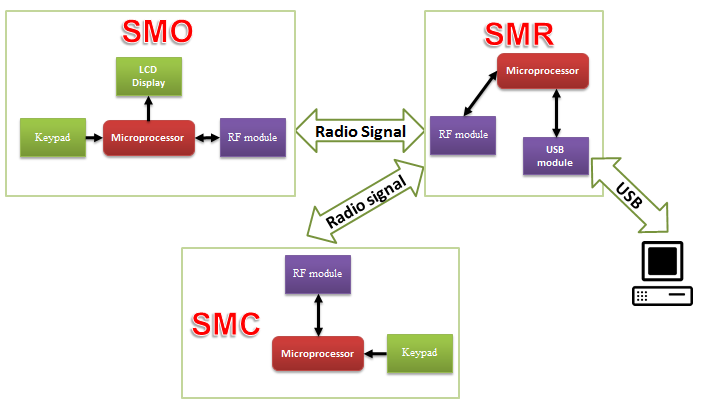
A CMA manages the full life cycle of content components, from inception through removal. A CMS will create, maintain and remove content components to and from a repository. Repository can be a database, a set of file, or a combination of both.

* **Metacontent Management Application (MMA)**

The MMA is an application that manages the full life cycle of metacontent. It looked like information about the content components, in particular how the content components are laid out on a application.

* **Content Delivery Application (CDA)**

The content delivery application’s job is to take the content components out of the CMS repository and display them, using metacontent, to the Web site user. CMS users usually do nothing with the CDA other than install and configure it. The reason for this is that it runs off the data you created with the CMA and the MMA.



# System Architectural Design

## Choice of System Architecture

When developing ASP.NET applications the need of reducing code duplication increases along with their complexity. This is because testing and performing changes become very difficult tasks when having many different sources of code with the same functionality.

Model View Controller architecture (or pattern) allows us to separate different parts of our applications into tiers to fulfill this need.

***MVC Overview:***

Model View Controller architecture aims to separate an application into three parts:

**Model:** It is the business logic of an application. From an object oriented perspective it would consist of a set of classes that implement the critical functionality of an application from a business point of view.

**View:** It can consist of every type of interface given to the user. In ASP.NET the view is the set of web pages presented by a web application.

**Controller:** This part of the architecture is the most difficult to explain, hence the most difficult to implement in many platforms. The controller is the object that allows the manipulation of the view. Usually many applications implement Model-Controller tiers that contain the business logic along with the necessary code to manipulate a user interface. In an ASP.NET application the controller is implicitly represented by the code-behind or the server side code that generates the HTML presented to the user.

## A basic diagram that would help us understand perfectly the specific parts that implement the Model View Controller architecture in an ASP.NET application is presented below:

## 

## Discussion of Alternative Designs

MVC is the separation of **M**odel, **V**iew and **C**ontroller — nothing more, nothing less. It's simply a paradigm; an ideal that you should have in the back of your mind when designing classes. Avoid mixing code from the three categories into one class.

For example, while a table grid view should obviously present data once shown, it should not have code on where to retrieve the data from, or what its native structure (the model) is like. Likewise, while it may have a function to sum up a column, the actual summing is supposed to happen in the controller.

A 'save file' dialog (view) ultimately passes the path, once picked by the user, on to the controller, which then asks the model for the data, and does the actual saving.

This separation of responsibilities allows flexibility down the road. For example, because the view doesn't care about the underlying model, supporting multiple file formats is easier: just add a model subclass for each.

Although originally developed for personal computing, MVC was readily adapted as architecture for World Wide Web applications. Several commercial and noncommercial application frameworks have been created to enforce the design. These frameworks vary in their interpretations, mainly in the way that the MVC responsibilities are divided between the client and server.

Early web MVC frameworks such as Java EE took a thin client approach that placed view and controller logic almost entirely on the server. In this approach, the client sends either hyperlink requests or form input to the controller and then receives a complete and updated web page (or other document) from the view; the model exists entirely on the server. As client technologies have matured, frameworks such as JavaScript MVC and Backbone have been created that allow the MVC components to execute partly on the client.

***Benefits of MVC:***

* **Substitutable user inter face:**

Different views and controllers can be substituted to provide alternate user interfaces for the same model.

* **User interface components:**

Because MVC demands that the user interface of an application be structured in to a hierarchy of objects and defines a standard relationship between these objects, generic versions of these object are possible. They are usually called user interface components and no modern GUI environment is without a full complement of the usually combining view and controller into a single object.

* **Multi simultaneous views of the same model:**

Multiple different views are active at the same time. Each view simultaneously and independently presents the same information from the model.

* **Synchronized views:**

The change propagation mechanism insures that all views simultaneously reflect the current state of the model.

* **Easier user interface changes:**

Changes affecting just the user interface of the application logic become easier to make.

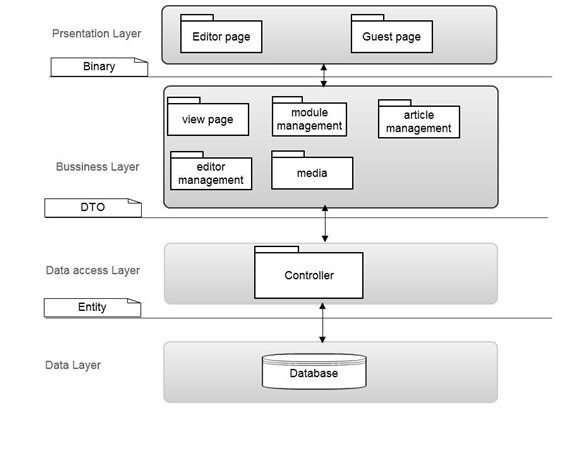
* **Easier testing:**

With MVC it can be easier to test the core of the application, as encapsulated by the model.

***Advantages of using MVC in ASP.NET:***

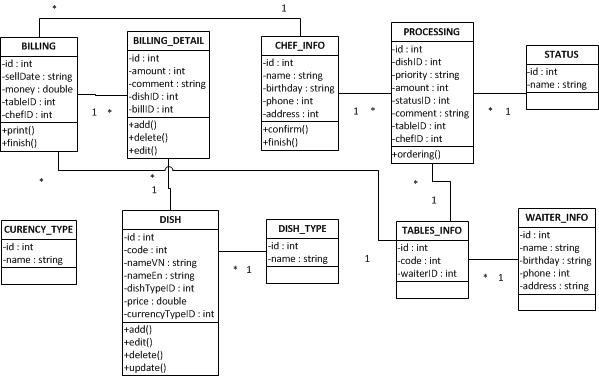
* There's no duplicated code.
* The business logic is encapsulated; hence the controller code is transparent and safer.
* The business logic can be used in several front ends like Web pages, Web services, Windows applications, services, etc.
* Exception handling is well managed showing the user only digested error messages.
* Testing every part of an application is easier as it can be done separately using automated methods.
* Application changes are easier to apply as they are focused in one part of the architecture only.

# Component diagram

****

# Detailed Description of Component

1. **Class Diagram**

****

**Class attribute list**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BILLING** | | | | |
| **No** | **Name** | **Type** | **Constrain** | **Description** |
| **1** | Id | Integer | Unique | ID of Billing |
| **2** | sellDate | String | No | Selling date |
| **3** | Money | Double | No | Total price |
| **4** | tableID | Integer | No | Table ID |
| **5** | chefID | Integer | No | Table ID |
| **6** | Description | String | No | Description of Media |
| **7** | UserID | Integer | Unique | ID of User |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BILLING\_DETAIL** | | | | |
| **No** | **Name** | **Type** | **Constrain** | **Description** |
| **1** | Id | Integer | Unique | ID of Bill |
| **2** | Amount | Integer | No | Amount of bill |
| **3** | Comment | String | No | Comment of dish |
| **4** | dishID | Integer | No | Dish ID |
| **5** | billID | Integer | No | Bill ID |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CHEF\_INFO** | | | | |
| **No** | **Name** | **Type** | **Constrain** | **Description** |
| **1** | Id | Integer | Unique | ID of Bill |
| **2** | Name | String | No | Name of chef |
| **3** | Birthday | String | No | Birthday of chef |
| **4** | Phone | Integer | No | Phone number |
| **5** | Address | String | No | Address of chef |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PROCESSING** | | | | |
| **No** | **Name** | **Type** | **Constrain** | **Description** |
| **1** | Id | Integer | Unique | ID of Bill |
| **2** | dishID | Integer | No | Dish ID |
| **3** | Priority | String | No | Priority |
| **4** | Amount | Integer | No | Amount of dish |
| **5** | statusID | Integer | No | Status of dish |
| **6** | Comment | String | No | Comment |
| **7** | tableID | Integer | No | Table ID |
| **8** | chefID | Integer | No | Chef ID |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STATUS** | | | | |
| **No** | **Name** | **Type** | **Constrain** | **Description** |
| **1** | Id | Integer | Unique | ID of Bill |
| **2** | Name | String | No | Name of status |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CURENCY\_TYPE** | | | | |
| **No** | **Name** | **Type** | **Constrain** | **Description** |
| **1** | Id | Integer | Unique | ID of Bill |
| **2** | Name | String | No | Name of type |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DISH** | | | | |
| **No** | **Name** | **Type** | **Constrain** | **Description** |
| **1** | Id | Integer | Unique | ID of Bill |
| **2** | code | Integer | No | Code |
| **3** | nameVN | String | No | Name of dish vietnamese |
| **4** | nameEN | String | No | Name of dish english |
| **5** | dishTypeID | Integer | No | Type of dish |
| **6** | Price | double | No | Price |
| **7** | currencyTypeID | Integer | No | Currency type |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DISH\_TYPE** | | | | |
| **No** | **Name** | **Type** | **Constrain** | **Description** |
| **1** | Id | Integer | Unique | ID of Bill |
| **2** | Name | String | No | Name of type |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLES\_INFO** | | | | |
| **No** | **Name** | **Type** | **Constrain** | **Description** |
| **1** | Id | Integer | Unique | ID of Bill |
| **2** | code | Integer | No | Code |
| **3** | waiterID | Integer | No | Waiter ID |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **WAITER\_INFO** | | | | |
| **No** | **Name** | **Type** | **Constrain** | **Description** |
| **1** | Id | Integer | Unique | ID of Bill |
| **2** | Name | string | No | Name of waiter |
| **3** | Birthday | String | No | Birthday |
| **4** | Phone | Integer | No | Phone number |
| **5** | Address | String | No | Address |

**Class operation list**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BILLING** | | | | | |
| **No** | **Name** | **Parameter** | **Return type** | **Exception** | **Description** |
|  |  |  |  |  |  |
| **1** | Print | Print | Void | NotFoundException | Print bill |
| **2** | Finish | Finish | Void | NotFoundException | Finish bill |

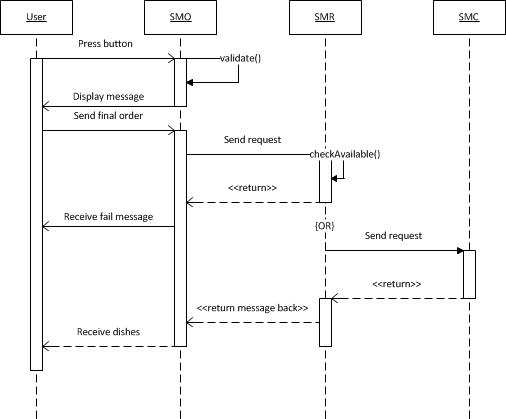
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BILLING\_DETAIL** | | | | | |
| **No** | **Name** | **Parameter** | **Return type** | **Exception** | **Description** |
|  |  |  |  |  |  |
| **1** | Add | Add | Void | NotFoundException | Add bill |
| **2** | Delete | Delete | Void | NotFoundException | Delete bill |
| **3** | Edit | Edit | Void | NotFoundException | Edit bill |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CHEF\_INFO** | | | | | |
| **No** | **Name** | **Parameter** | **Return type** | **Exception** | **Description** |
|  |  |  |  |  |  |
| **1** | Confirm | Confirm | Void | NotFoundException | Chef confirm |
| **2** | Finish | Finish | Void | NotFoundException | Chef finish |

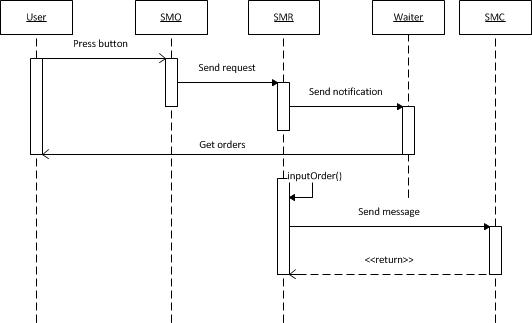
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROCESSING** | | | | | |
| **No** | **Name** | **Parameter** | **Return type** | **Exception** | **Description** |
|  |  |  |  |  |  |
| **1** | Ordering | Order | Void | NotFoundException | Ordering |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **DISH** | | | | | |
| **No** | **Name** | **Parameter** | **Return type** | **Exception** | **Description** |
|  |  |  |  |  |  |
| **1** | Add | Print | Void | NotFoundException | Add dish |
| **2** | Edit | Finish | Void | NotFoundException | Edit dish |
| **3** | Delete | Delete | Void | NotFoundException | Delete dish |
| **4** | Update | Update | Void | NotFoundException | Update dish |

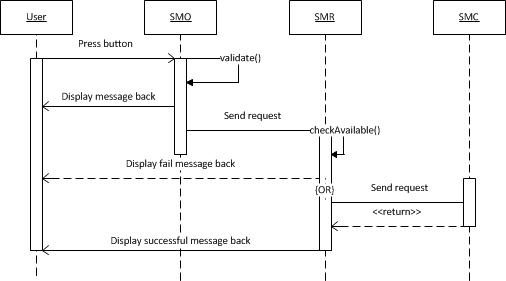
1. **Sequence Diagram**
   1. **Customer Management**
      1. **Order Dishes**

****

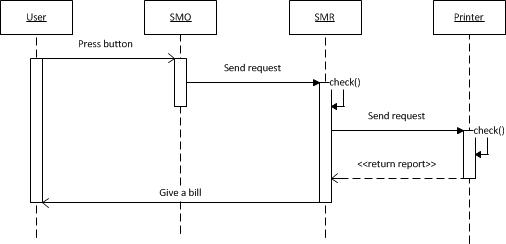
* + 1. **Call Waiter**

****

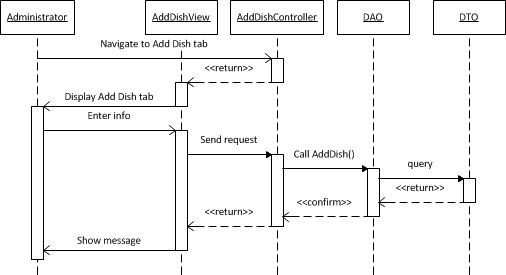
* + 1. **Cancel order**

****

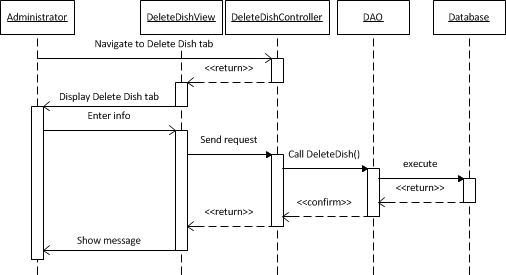
* + 1. **Call for payment**

****

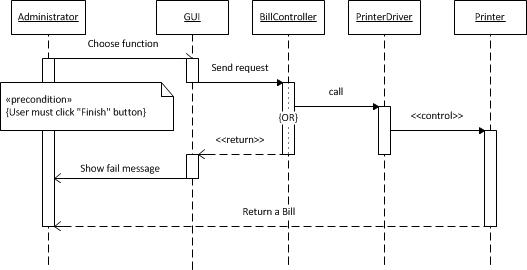
* 1. **Category Management**
     1. **Add dishes**

****

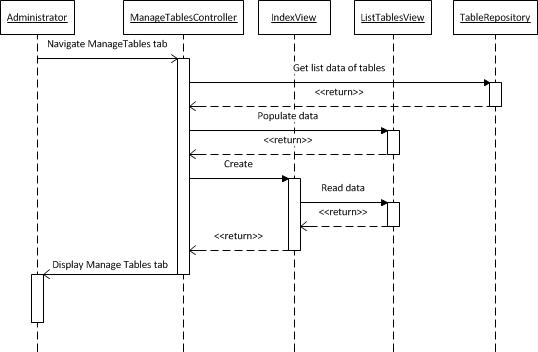
* + 1. **Delete dishes**

****

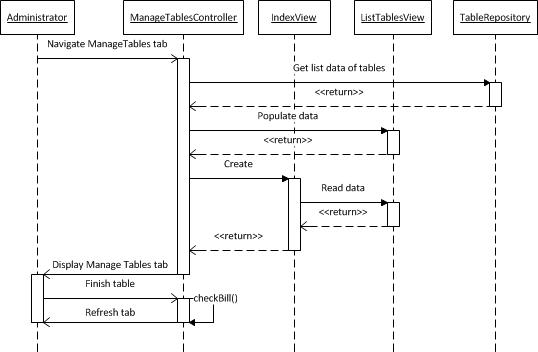
* + 1. **Print bill**

****

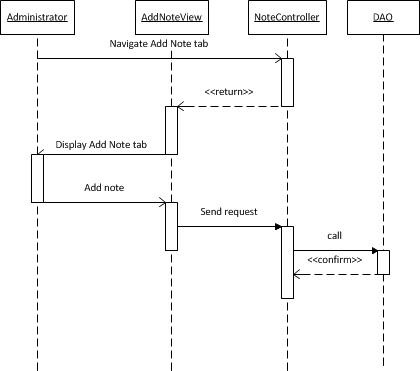
* + 1. **View list of tables**

****

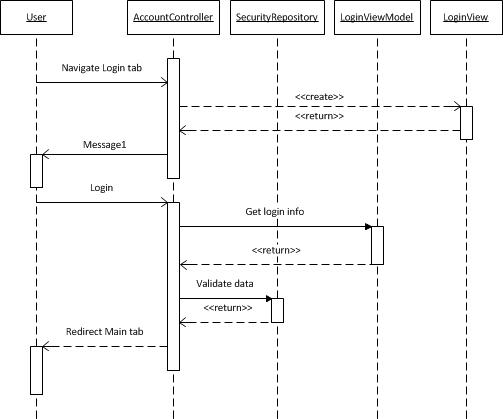
* + 1. **Finish table**

****

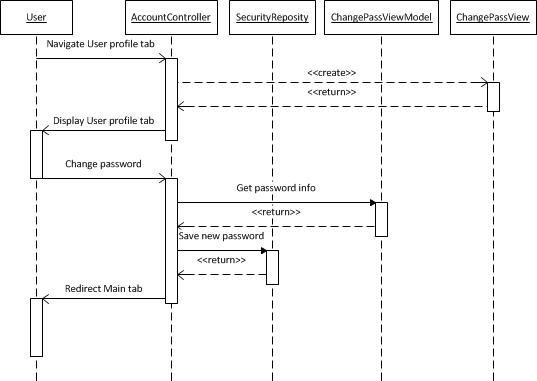
* + 1. **Add note**

****

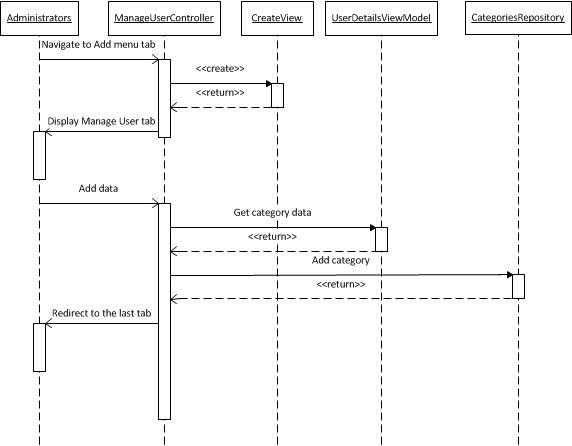
* + 1. **Login**

****

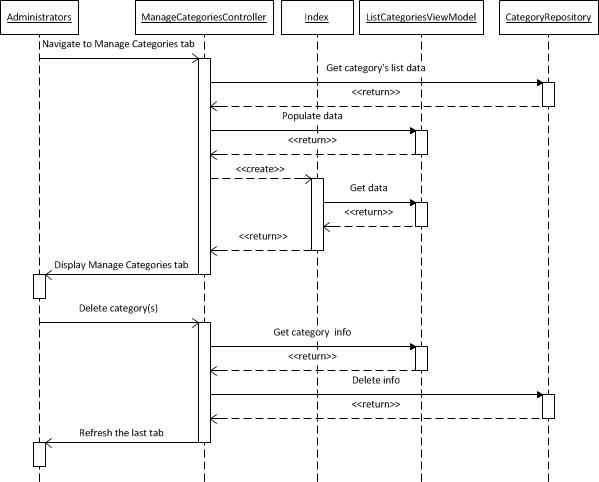
* + 1. **Change password**

****

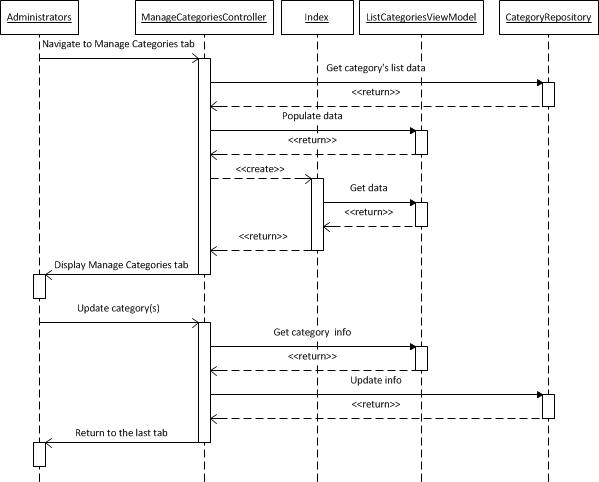
* + 1. **Add menu**

****

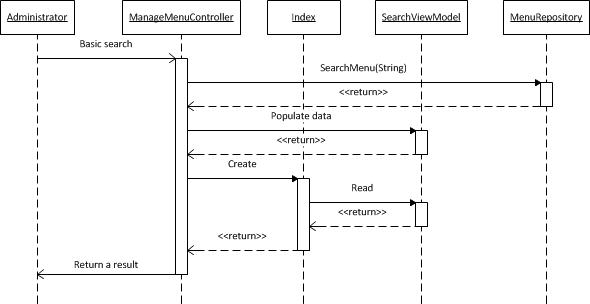
* + 1. **Delete menu**

****

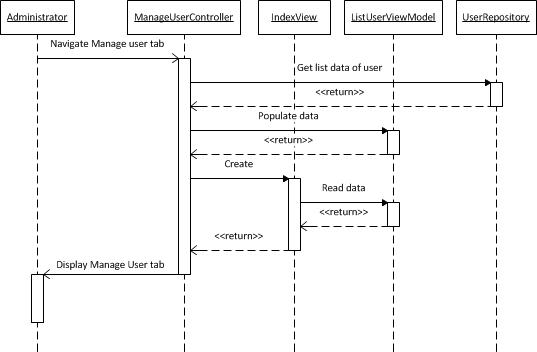
* + 1. **Edit menu**

****

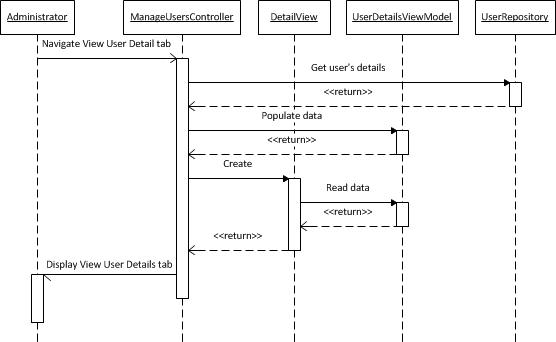
* + 1. **Search menu**

****

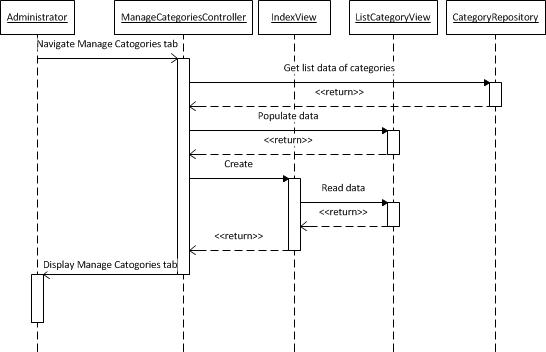
* + 1. **View staff list**

****

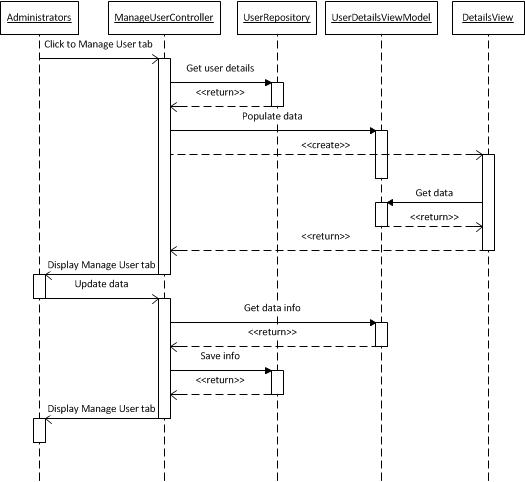
* + 1. **View staff detail**

****

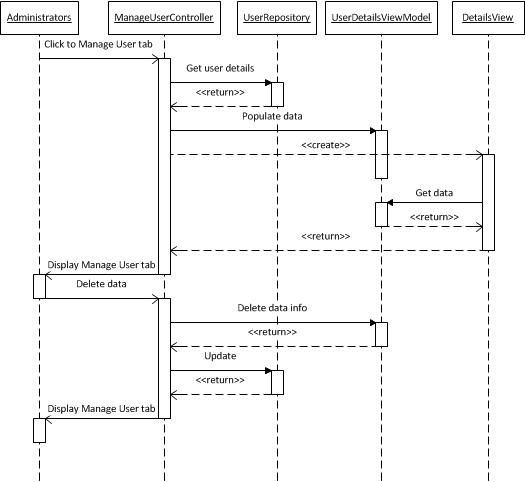
* + 1. **List menu**

****

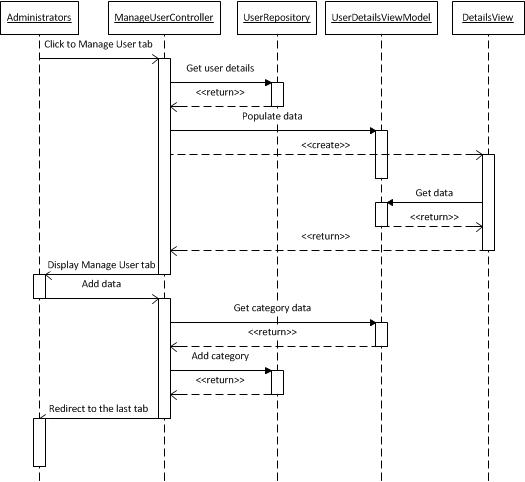
* + 1. **Edit staff detail**

****

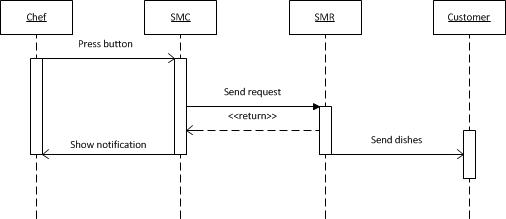
* + 1. **Delete staff**

****

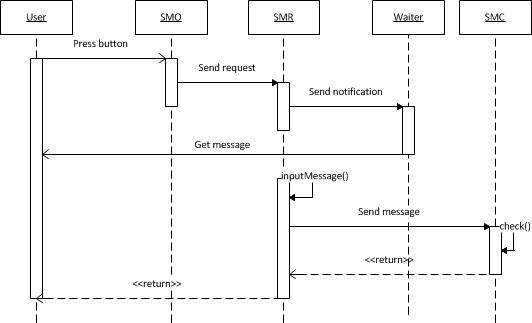
* + 1. **Add staff detail**

****

* 1. **Chef**
     1. **Finish Dishes**

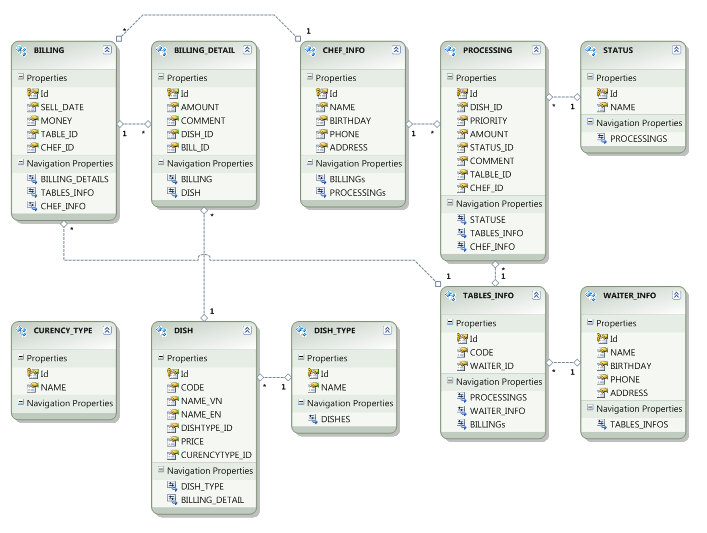
****

* + 1. **Receive Request**

****

# User Interface Design

# Database Design



# References