|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | **Logo_FPT_University_doc** | **MINISTRY OF EDUCATION AND TRAINING** |   **FPT UNIVERSITY**   |  | | --- | | Capstone Project | |
|  |
| Cooking recipes sharing website  Report 4: Software Design Description |
| |  |  | | --- | --- | | **Master Chefs team** | | | **Group Members** | Trần Trung Kiên - 01017  Đặng Xuân Toàn - 01143  Trần Sỹ Tùng - 01302  Bùi Trung Kiên - 01359  Ngô Thị Việt Anh - 01286 | | **Supervisor** | Huỳnh Anh Dũng | | **Ext Supervisor** | Nguyễn Tất Trung | | **Capstone Project code** | CTA\_v1.0 | |
|  |

- Hanoi, 7/2012 –

**Record of Changes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Change Item** | **Description** | **By** | **Version** |
| 18/6/2011 | All | Create the document | ToanDX | 0.1 |
| 19/6/2011 | Detail Design | Edit content | ToanDX | 0.2 |
| 20/6/2011 | Detail Design | Edit content | ToanDX | 0.3 |
| 21/6/2011 | - Design Overview  - System Architectural Design  - References | Create | ToanDX | 0.4 |
| 23/6/2011 | Database Design | Create | ToanDX | 0.5 |
| 24/6/2011 | - UI Design  - References | - Remove content  - Edit content | ToanDX | 0.6 |
| 24/6/2011 | Database design | Some description | ToanDX | 0.7 |
| 5/7/2011 | Change design | Change all cases | ToanDX | 0.8 |
| 8/7/2011 | Change database design | Add new and edit tables in Database design | ToanDX | 0.9 |
| 10/7/2011 | All | Add detail designs for new use cases | ToanDX | 1.0 |

Contents

[I. Design Overview 6](#_Toc333044140)

[II. System Architectural Design 6](#_Toc333044141)

[1. Overall System Architecture 6](#_Toc333044142)

[2. Web Application Architecture 8](#_Toc333044143)

[2.1 CRS.Common Component 8](#_Toc333044144)

[2.2 CRS.Resources Component 9](#_Toc333044145)

[2.3 CRS.Business Component 9](#_Toc333044146)

[2.4 CRS.Web Component 10](#_Toc333044147)

[III. Detailed Description of Component 12](#_Toc333044148)

[1. Comment 12](#_Toc333044149)

[1.1 Class Diagram 13](#_Toc333044150)

[1.2 Class Explanations 14](#_Toc333044151)

[1.3 Sequence Diagrams 15](#_Toc333044152)

[2. Search Content 18](#_Toc333044153)

[2.1 Class Diagram 19](#_Toc333044154)

[2.2 Class Explanations 19](#_Toc333044155)

[2.3 Sequence Diagrams 20](#_Toc333044156)

[3. Statistics 21](#_Toc333044157)

[3.1 Class Diagram 21](#_Toc333044158)

[3.2 Class Explanations 22](#_Toc333044159)

[3.3 Sequence Diagrams 23](#_Toc333044160)

[4. Question 24](#_Toc333044161)

[4.1. Class Diagram 25](#_Toc333044162)

[4.2. Class Explanations 25](#_Toc333044163)

[4.3. Sequence Diagrams 27](#_Toc333044164)

[5. Answer 32](#_Toc333044165)

[5.1 Class Diagram 33](#_Toc333044166)

[5.2 Class Explanations 35](#_Toc333044167)

[5.3 Sequence Diagrams 36](#_Toc333044168)

[6. Manage Categories 39](#_Toc333044169)

[6.1 Class Diagram 40](#_Toc333044170)

[6.2 Class Explanations 40](#_Toc333044171)

[6.3 Sequence Diagrams 42](#_Toc333044172)

[7. Manage Spam Contents 45](#_Toc333044173)

[1.1 Class Diagram 46](#_Toc333044174)

[1.2 Class Explanations 46](#_Toc333044175)

[1.3 Sequence Diagrams 48](#_Toc333044176)

[8. Manage Spam Comments 50](#_Toc333044177)

[8.1 Class Diagram 51](#_Toc333044178)

[8.2 Class Explanations 51](#_Toc333044179)

[8.3 Sequence Diagrams 53](#_Toc333044180)

[9. Manage Spam Answers 55](#_Toc333044181)

[9.1 Class Diagram 56](#_Toc333044182)

[9.2 Class Explanations 56](#_Toc333044183)

[9.3 Sequence Diagrams 58](#_Toc333044184)

[10. Manager Users 60](#_Toc333044185)

[10.1 Class Diagram 61](#_Toc333044202)

[10.2 Class Explanations 62](#_Toc333044203)

[10.3 Sequence Diagrams 63](#_Toc333044204)

[11. Contents 65](#_Toc333044205)

[11.1 Class Diagram 65](#_Toc333044206)

[11.2 Class Explanations 67](#_Toc333044207)

[11.3 Sequence Diagrams 68](#_Toc333044208)

[12. Manage Spam Questions 72](#_Toc333044209)

[12.1 Class Diagram 73](#_Toc333044210)

[12.2 Class Explanations 73](#_Toc333044211)

[12.3 Sequence Diagrams 75](#_Toc333044212)

[IV. Database Design 77](#_Toc333044213)

[1. Database Design 77](#_Toc333044214)

[2. Tables 77](#_Toc333044215)

[a. Answers table 78](#_Toc333044216)

[2.2 FavoriteRecipes table 79](#_Toc333044217)

[2.3 Locations table 79](#_Toc333044218)

[2.4 News table 79](#_Toc333044219)

[2.5 NewsComments table 80](#_Toc333044220)

[2.6 Permissions table 80](#_Toc333044221)

[2.7 PointConfigs table 81](#_Toc333044222)

[2.8 Questions table 81](#_Toc333044223)

[2.9 RatedRecipes table 81](#_Toc333044224)

[2.10 RecipeCategories table 82](#_Toc333044225)

[2.11 RecipeCategoryMapping table 82](#_Toc333044226)

[2.12 RecipeComments table 82](#_Toc333044227)

[2.13 Recipes table 83](#_Toc333044228)

[2.14 RecipeSmallCategories table 83](#_Toc333044229)

[2.15 ReportedAnswers table 84](#_Toc333044230)

[2.16 ReportedNews table 84](#_Toc333044231)

[2.17 ReportedNewsComments table 84](#_Toc333044232)

[2.18 ReportedQuestions table 85](#_Toc333044233)

[2.19 ReportedRecipeComments table 85](#_Toc333044234)

[2.20 ReportedRecipes table 85](#_Toc333044235)

[2.21 ReportedTipComments table 86](#_Toc333044236)

[2.22 ReportedTips table 86](#_Toc333044237)

[2.23 RolePermissions table 86](#_Toc333044238)

[2.24 Roles table 87](#_Toc333044239)

[2.25 TipCategories table 87](#_Toc333044240)

[2.26 TipComments table 87](#_Toc333044241)

[2.27 Tips table 87](#_Toc333044242)

[2.28 Titles table 88](#_Toc333044243)

[2.29 UpdateWords table 88](#_Toc333044244)

[2.30 UserRoles table 89](#_Toc333044245)

[2.31 Users table 89](#_Toc333044246)

[2.32 UserStates table 90](#_Toc333044247)

[2.33 VisitorNumbers table 90](#_Toc333044248)

[2.34 VotedAnswers table 90](#_Toc333044249)

[2.35 VotedNewsComments table 90](#_Toc333044250)

[2.36 VotedRecipeComments table 91](#_Toc333044251)

[2.37 VotedTipComments table 91](#_Toc333044252)

[2.38 VotedTips table 91](#_Toc333044253)

[2.39 Words table 91](#_Toc333044254)

[V. References 92](#_Toc333044255)

# 

# Design Overview

This document describes the technical and UI design of the system. It includes the architectural design and the detailed design of common functions and business functions. It also includes the design of database model, and overall design of user interface.

The architectural design describes the overall architecture of the system, and the architecture of each main component and subsystem. It will describe the patterns being used, the role of each component and the role of the system in the working environment.

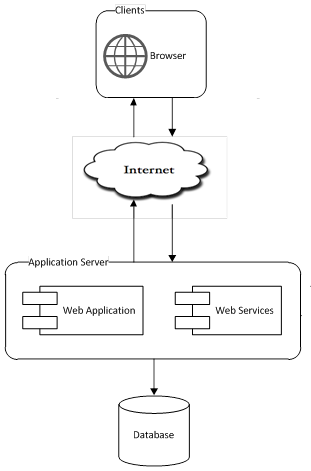
The detailed design describes static and dynamic structure for each component and function. It includes class diagrams, class explanations, and sequence diagrams of the main use cases. The detailed design uses notations of .NET framework 4.0 and C# 4.0 as they are the framework and language for developing the system.

The database design describes the relationship between entities, and details of each entity. It uses notations of SQL Server 2008 as it is the database server for developing the system.

The user interface design describes the layout of the system, and some design for the screens.

# System Architectural Design

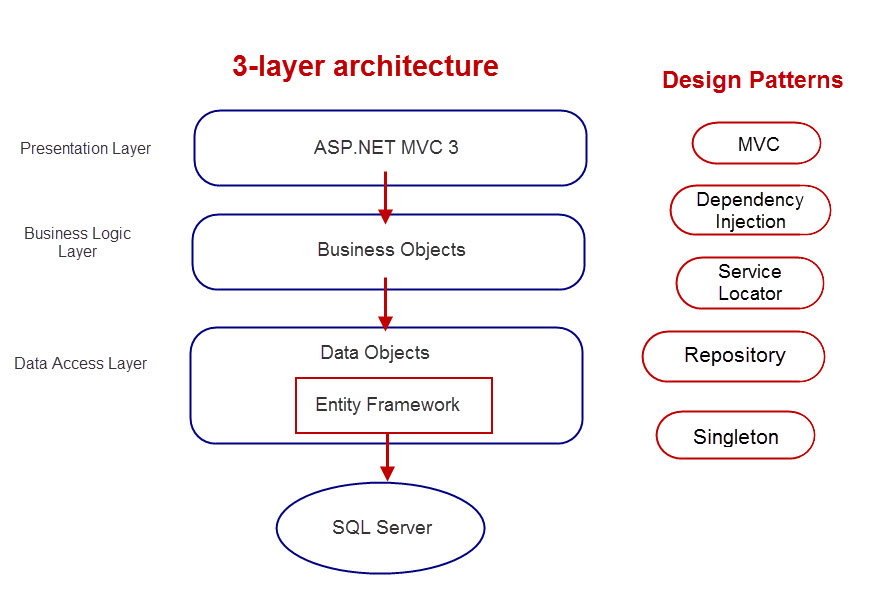
## Overall System Architecture



The system will use the Client-Server architectural patterns as the main architecture. It makes use of the Internet and common web protocols to deliver its services to the users.

The main entities of the above diagram will be described as following:

* **Database store:** this is the central data storage for the system. It may be running on a separate server to the Application Server, or in the same machine.
* **Application Server:** this is the server of the system running on IIS web server. It includes two main components: Web Application and Web Services. Web Application is responsible for creating web pages. It is developed using **ASP.NET MVC 3** framework. Web Services provides the system’s API to the world and some parts of the application.
* **Clients:** these include internet browsers running on personal computers, on mobile devices or on tablet. The web browsers will interact with the web application via normal protocols (HTTP, HTTPS…).



Patterns will be used throughout the project:

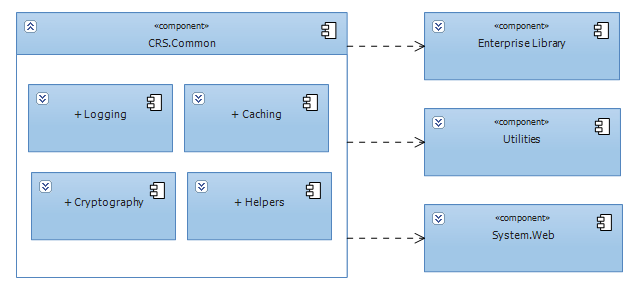
MVC, Dependency Injection, Service Locator, Repository, Singleton

## Web Application Architecture

The architecture for the CRS web application will be depicted as the following diagram:

### CRS.Common Component

This component contains common logic that can be used throughout the application. It includes for example mechanism for logging, caching, cryptography, and other utilities.



### CRS.Resources Component

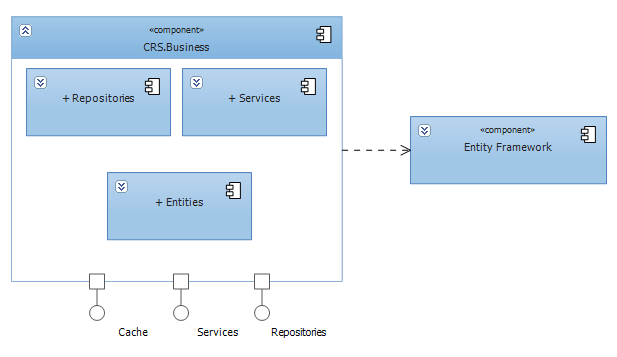
This component contains static resources for the system. It includes localized strings and messages.

### CRS.Business Component

This component contains all business logic for the system. It includes object-relational mapping classes, business classes, data access layer, business interface…

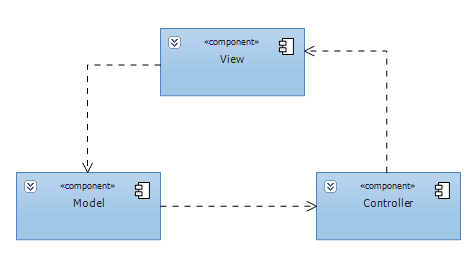
The aim and challenge for the design is that this component will be application type independent. That is this component is not tied only to web application but can be used for other .NET application types if necessary.

All the services of this component will be used by dependent components via its interfaces.



### CRS.Web Component

This component is responsible for delivering the front end to the users and accepting their inputs. It will be developed using ASP.NET MVC 3 framework. An MVC Application is designed and implemented using the following three attributes:

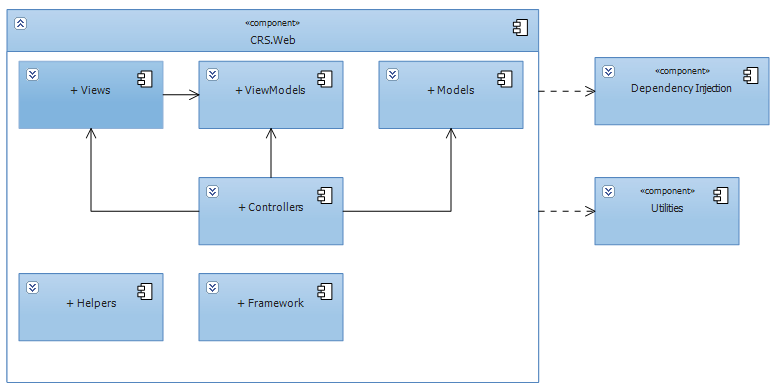


* **Model:** The model contains the core information for an application. This includes the data and validation rules as well as data access and aggregation logic.
* **View:** The view encapsulates the presentation of the application, and in ASP.NET this is typically the HTML markup.
* **Controller:** The controller contains the control-flow logic. It interacts with the Model and Views to control the flow of information and execution of the application.

The **CRS** application will make use another attribute: **View Model**. The view model encapsulates all the data properties that the view needs or provides. It is the view model that the view depends directly on, not the model. The model will be implemented by **CRS.Business** components, except those not related to business logics.



The overall architecture of CRS.Web component will be depicted as the following diagram:



* **Helpers:** These are utilities classes specific for this component
* **Framework:** Contain the core framework to build web application

# Detailed Description of Component

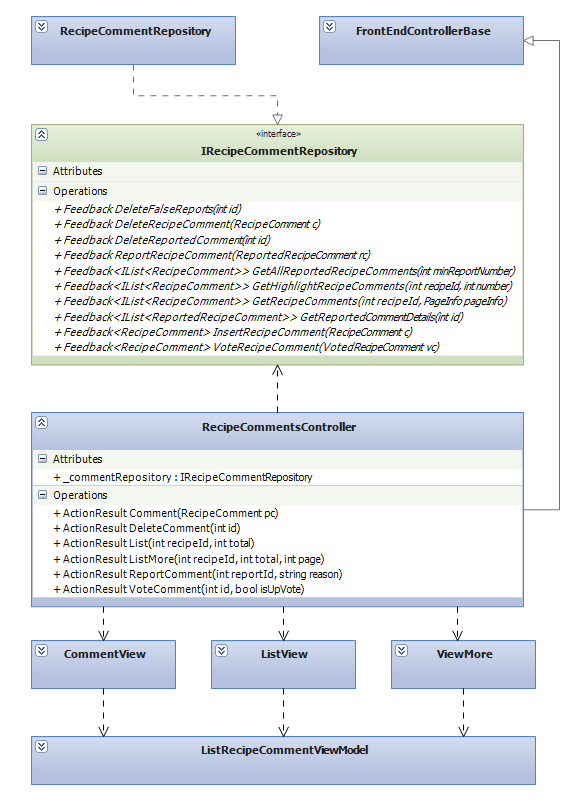
## Comment

This part is the detailed design for the following use cases:

* Post a comment
* Edit a comment
* Delete a comment
* Report a comment
* Vote a comment

All contents (news, tips, and recipes) contain their own comment but the structure is the same. The following diagrams are drawn follow recipe comment.

### Class Diagram



### Class Explanations

#### IRecipeCommentRepository interface

Outlines the methods that the data repository class (RecipeCommentRepository) will be able to perform. The data repository class will expand thisrepository pattern interface to include a full set of data access functions such as Select, Insert, Update, Delete.

Responsibilities:

* Delete comment
* Report comment
* Update vote status of a comment
* Get comments
* Get highlight comments
* Insert a new comment

#### RecipeCommentRepository class

Default implementation of IRecipeCommentRepository interface, contains the required code for accessing the MSSQL database

#### RecipeCommentsController class

The basic of responsibilities of RecipeCommentsController is to carry out the various use cases

Responsibilities:

* Allow all users to perform View Undeleted Comments Use Case
* Allow registered users to perform Insert A New Comment Use Case
* Allow registered users to perform Vote A Comment Use Case
* Allow registered users to Report A Comment Use Case
* Allow registered users users to perform Delete Their Own Comments Use Case

#### ListView class

Contain HTML markup and .NET code to render List Comments View screen

#### CommentView class

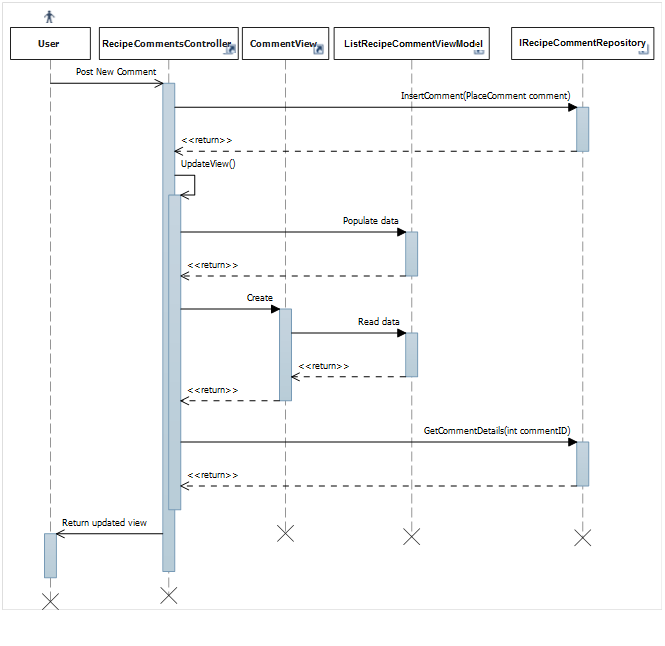
Contain HTML markup and .NET code to render Comment View screen

#### ListRecipeCommentViewModel class

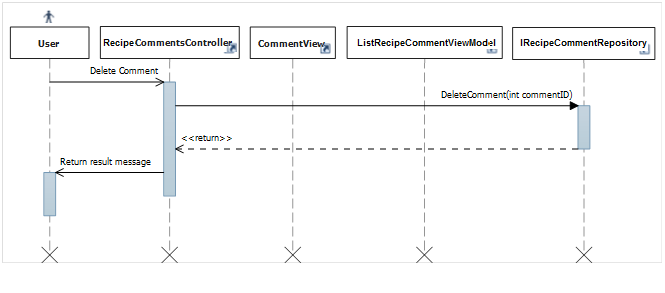
Model type of ListView and CommentView

### Sequence Diagrams

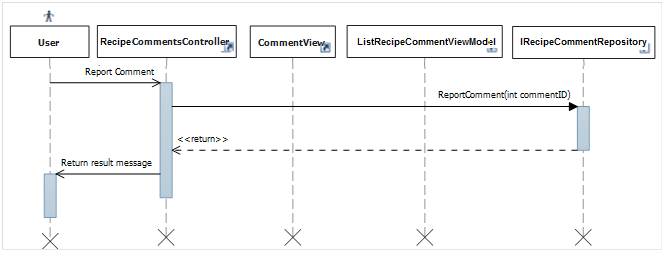
#### 1.3.1 Create Comment



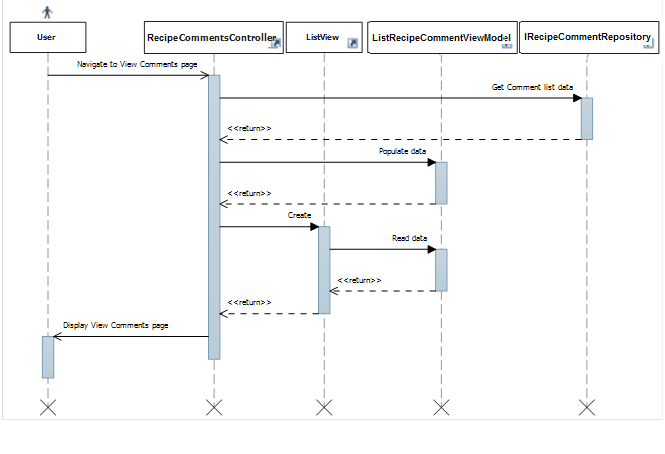
#### 1.3.2 Delete Comment



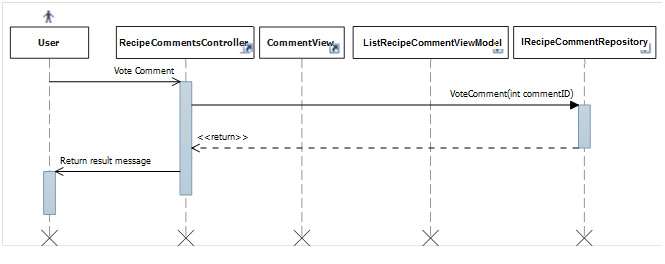
#### 1.3.3 Report Comment



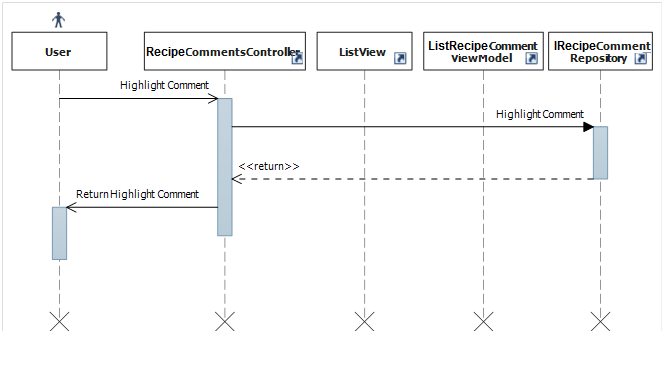
#### 1.3.4 View Comment List



#### 1.3.5 Vote Comment



#### 1.3.6 Highlight Best Comment

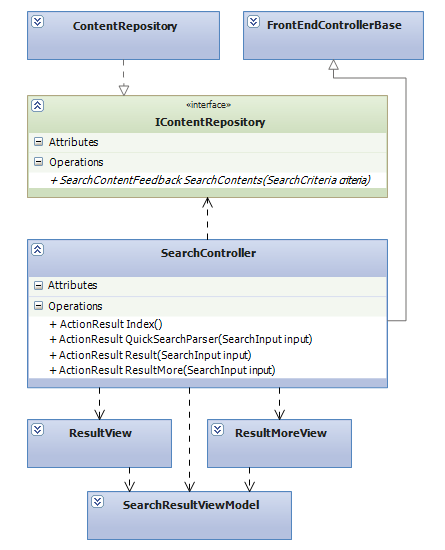


## Search Content

This part is the detailed design for the following use cases:

* Search Content (includes news, tips, recipes and all the contents)

### Class Diagram



### Class Explanations

#### IContentRepository interface

Outlines the methods that the data repository class (ContentRepository) will be able to perform. The data repository class will expand this repository pattern interface to include Select data access functions

Responsibilities:

* Search contents (includes search on each type of content (news, tips, recipes) and search all contents)

#### ContentRepository class

Default implementation of IContentRepository interface, contains the required code for accessing the MSSQL database

#### SearchController class

The basic of responsibilities of RecipeCommentsController is to carry out the various use cases

Responsibilities:

* Allow all users to perform Search Content Use Case

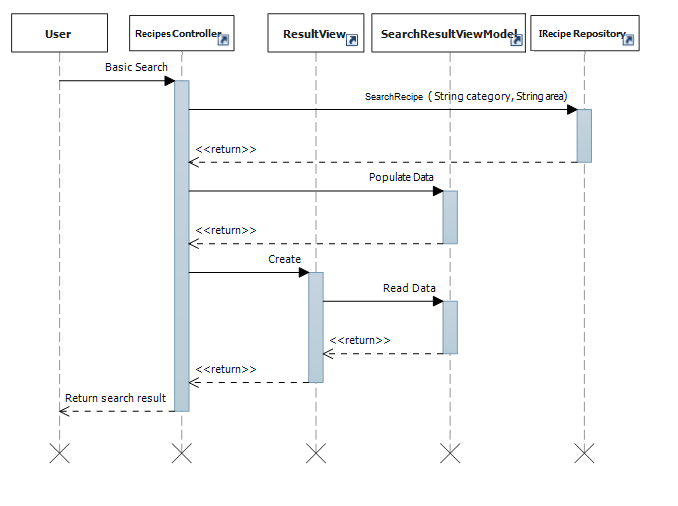
#### ResultView class

Contain HTML markup and .NET code to render Search Result View screen

#### SearchResultViewModel class

Model type of ResultView

### Sequence Diagrams

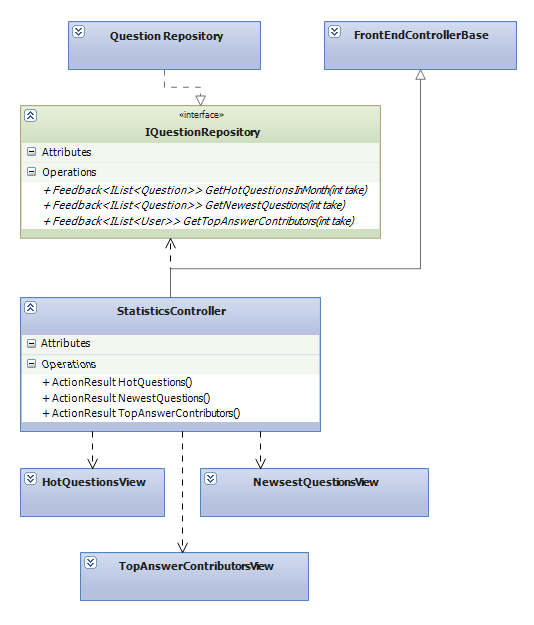


## Statistics

This part is the detailed design for the following use cases:

* Hot questions in month
* Top answer contributors
* Newest questions

### Class Diagram



### Class Explanations

#### IQuestionRepository interface

Outlines the methods that the data repository class (QuestionRepository) will be able to perform. For statistical purposes, the data repository class will expand this repository pattern interface to include Select data access functions.

Responsibilities:

* Get list hot questions
* Get list top answer contributors
* Get list newest questions

#### QuestionRepository class

Default implementation of IQuestionRepository interface, contains the required code for accessing the MSSQL database

#### StatisticsController class

The basic of responsibilities of StatisticsControlleris to carry out the various use cases

Responsibilities:

* Allow all users to perform View Hot Questions In Month Use Case
* Allow all users to perform View Newest Questions Use Case
* Allow all users to perform View Top Answer Contributors Use Case

#### HotQuestionsView class

Contain HTML markup and .NET code to render Hot Questions View screen

#### NewestQuestionsView class

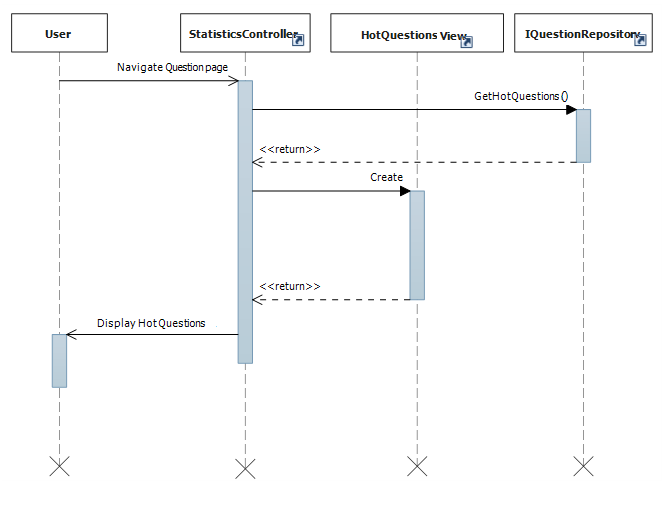
Contain HTML markup and .NET code to render Newest Questions View screen

#### TopAnswerContributorsView class

Contain HTML markup and .NET code to render Top Answer Contributors

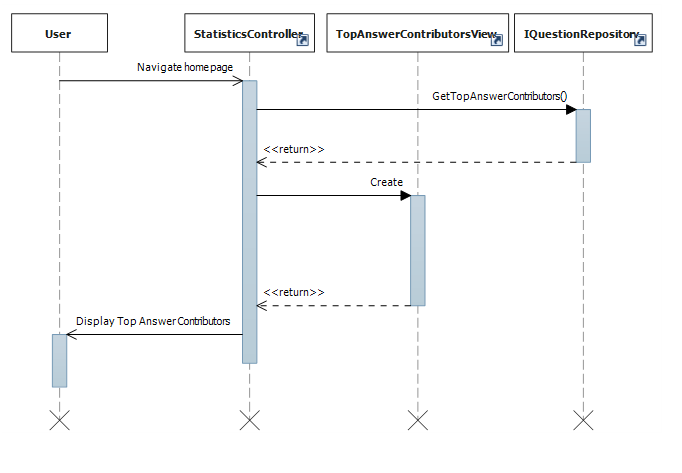
### Sequence Diagrams

#### Hot Questions



#### Newest Questions

#### Top Answer Contributors

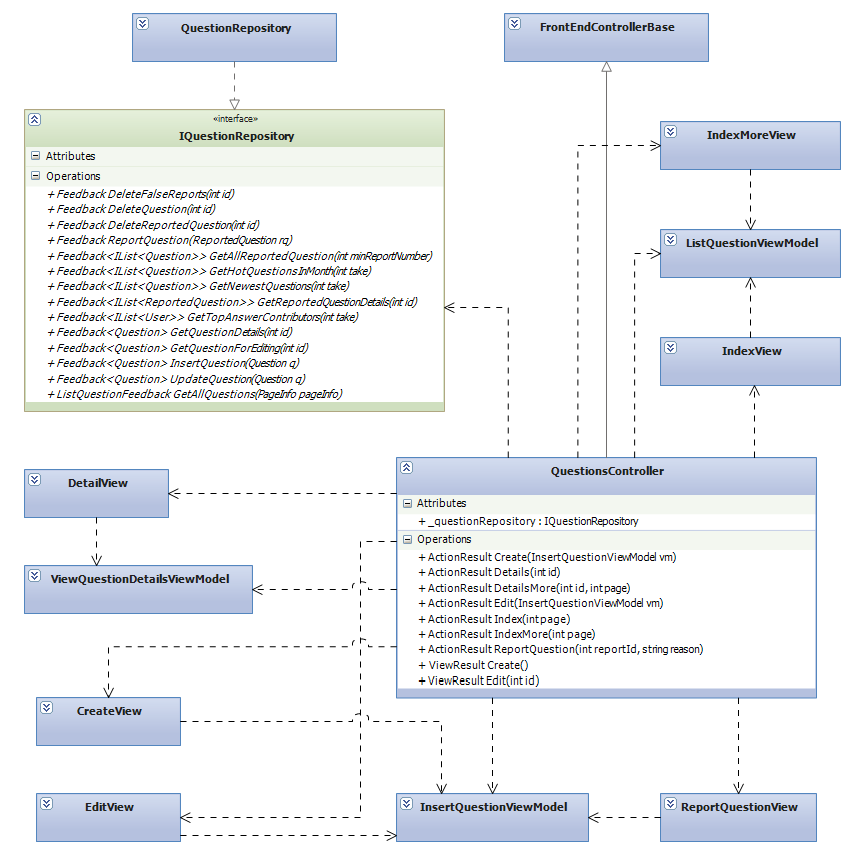


## Question

This part is the detailed design for the following use cases:

* Get Question List
* Get question’s details
* Insert a new question
* Update questions
* Report questions

### Class Diagram



### Class Explanations

#### IQuestionRepository interface

Outlines the methods that the data repository class (QuestionRepository) will be able to perform. The data repository class will expand this repository pattern interface to include a full set of data access functions such as Select, Insert, Update, Delete.

Responsibilities:

* Get Question List
* Get question’s details
* Insert a new question
* Update questions
* Report questions

#### QuestionRepository class

Default implementation of IQuestionRepository interface, contains the required code for accessing the MSSQL database

#### QuestionsController class

The basic of responsibilities of QuestionsControlleris to carry out the various use cases

Responsibilities:

* Allow all users to performView Question List Use Case
* Allow all users to perform View Question’s Details Use Case
* Allow registered users to perform Insert A New Question Use Case
* Allow registered users to perform Update Questions Use Case
* Allow registered users to perform Report Questions Use Case

#### IndexView class

Contain HTML markup and .NET code to render Index View screen

#### DetailViewclass

Contain HTML markup and .NET code to render Detail View screen

#### CreateViewclass

Contain HTML markup and .NET code to render Create View screen

#### EditViewclass

Contain HTML markup and .NET code to render Edit View screen

#### ReportQuestionViewclass

Contain HTML markup and .NET code to render Report Question View screen

#### ListQuestionViewModelclass

Model type of IndexView

#### InsertQuestionViewModelclass

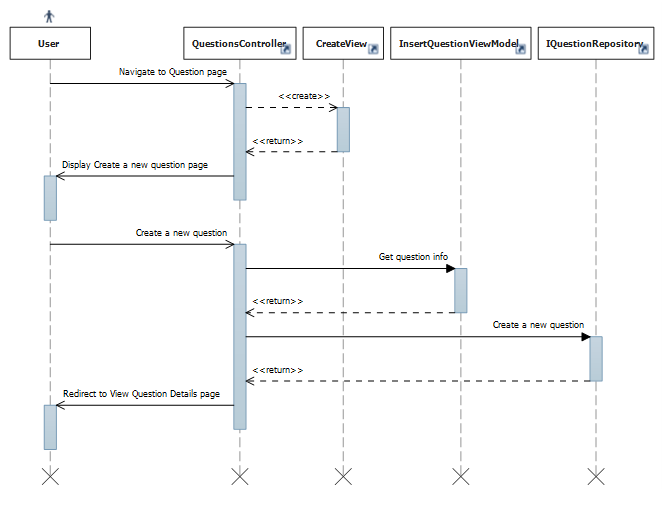
Model type of CreateView,EditView and ReportQuestionView

#### ViewQuestionDetailsViewModelclass

Model type of DetailView

### Sequence Diagrams

#### Create Question



#### Edit Question



#### Get All Questions



#### Get Question Details



#### Report Question



#### Vote Question

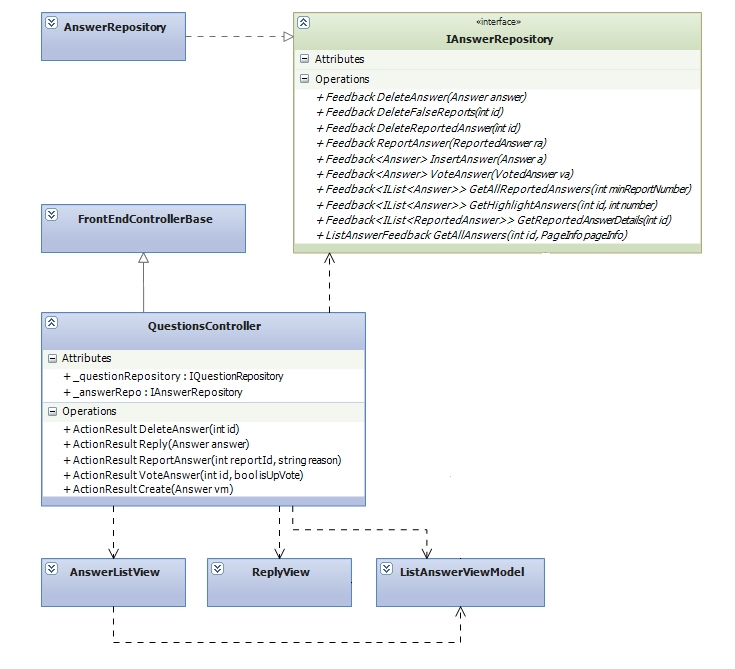


## Answer

This part is the detailed design for the following use cases:

* Get answer list
* Reply question

### Class Diagram





### Class Explanations

#### IAnswerRepository interface

Outlines the methods that the data repository class (AnswerRepository) will be able to perform. The data repository class will expand this repository pattern interface to include a set of data access functions such as Select, Insert, Delete.

Responsibilities:

* Get Answer list
* Vote answers
* Insert a newanswer
* Delete answers
* Get highlight answers

#### AnswerRepository class

Default implementation of IAnswerRepository interface, containsthe required code for accessing the MSSQL database

#### QuestionsController class

The basic of responsibilities of QuestionsControlleris to carry out the various use cases

Responsibilities:

* Allow all users to perform View Answer List Use Case
* Allow all users to perform View Highlight Answer Use Case
* Allow registered users to perform Insert A New Answer Use Case
* Allow registered users to perform Vote Answers Use Case
* Allow registered users to perform Delete Their OwnAnswer Use Case

#### AnswerListView class

Contain HTML markup and .NET code to render Answer List View screen

#### ReplyViewclass

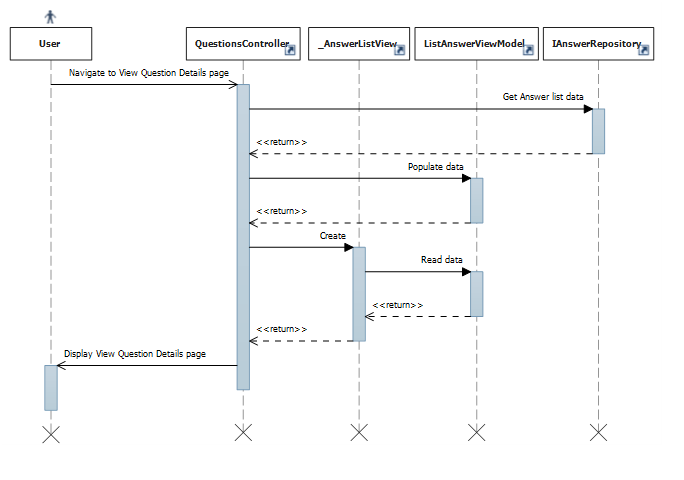
Contain HTML markup and .NET code to render Reply View screen

#### ListAnswerViewModelclass

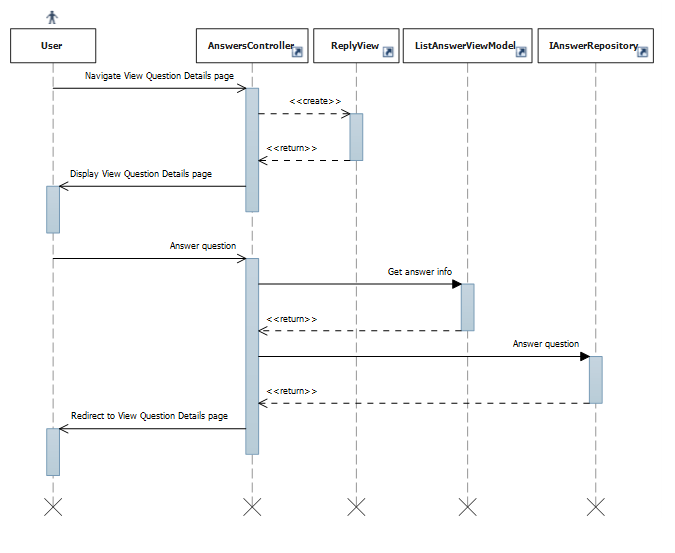
Model type of AnswerListView and ReplyView

### Sequence Diagrams

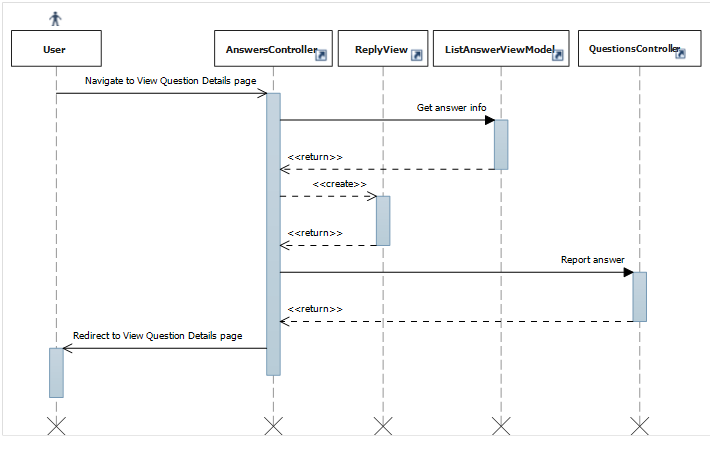
#### Get Answer List



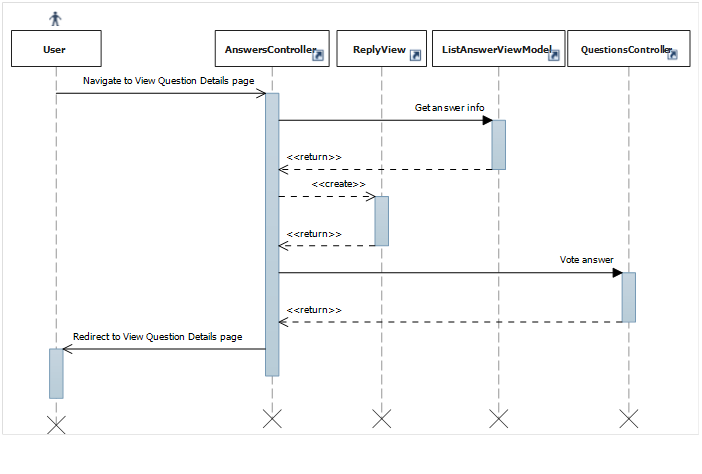
#### Reply



#### Report Answer



#### Vote Answer



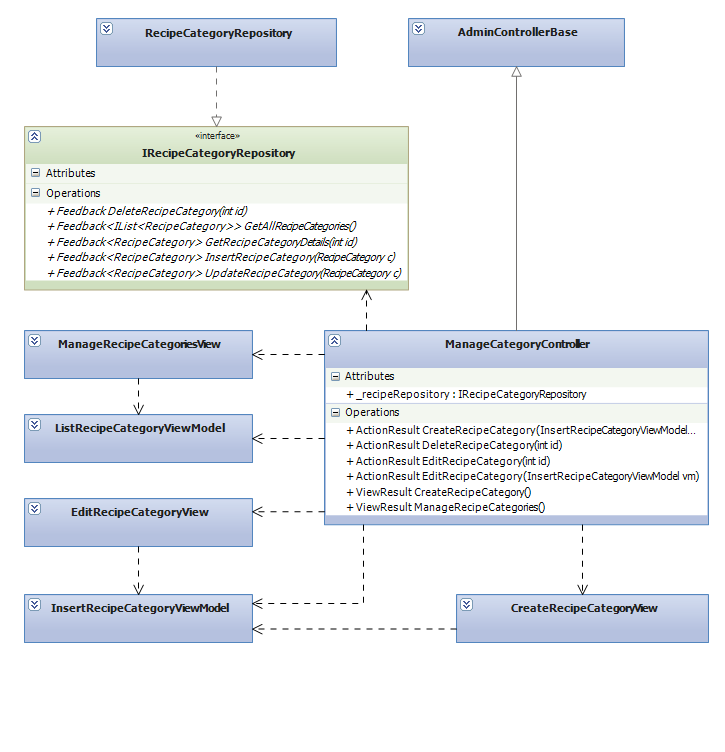
## Manage Categories

This part is the detailed design for the following use cases:

* View Categories list
* View Category ‘s details
* Create a new category
* Edit category
* Delete category

The following detailed design is follow Recipe categories but can be used for Tip categories as well

### Class Diagram



### Class Explanations

#### IRecipeCategoryRepository interface

Outlines the methods that the data repository class (RecipeCategoryRepository) will be able to perform. The data repository class will expand this repository pattern interface to include a full set of data access functions such as Select, Delete, Update, Insert.

Responsibilities:

* Insert a new category
* Update categories
* Get category’s detail
* Get all categories
* Delete categories

#### RecipeCategoryRepository class

Default implementation of IRecipeCategoryRepository interface, contains the required code for accessing the MSSQL database

#### ManageCategoriesController class

The basic of responsibilities of ManageCategoriesControlleris to carry out the various use cases

Responsibilities:

* Allow administrator and power users to perform Insert A New Category Use Case
* Allow administrator and power users to perform Update Categories Use Case
* Allow administrator and power users to perform View Category’s Detail Use Case
* Allow administrator and power users to perform View All Categories Use Case
* Allow administrator and power users to perform Delete Categories Use Case

#### ManageRecipeCategoriesView class

Contain HTML markup and .NET code to render Manage Recipe Categories View screen

#### CreateRecipeCategoryView class

Contain HTML markup and .NET code to render Create Recipe Category View screen

#### EditRecipeCategoryView class

Contain HTML markup and .NET code to render Edit Recipe Category View screen

#### InsertRecipeCategoryViewModel class

Model type of EditRecipeCategoryView and CreateRecipeCategoryView

#### ListRecipeCategoryViewModel class

Model type of ManageRecipeCategoryView

### Sequence Diagrams

#### Delete Category



#### Get All Categories



#### Insert Category



#### Edit Category

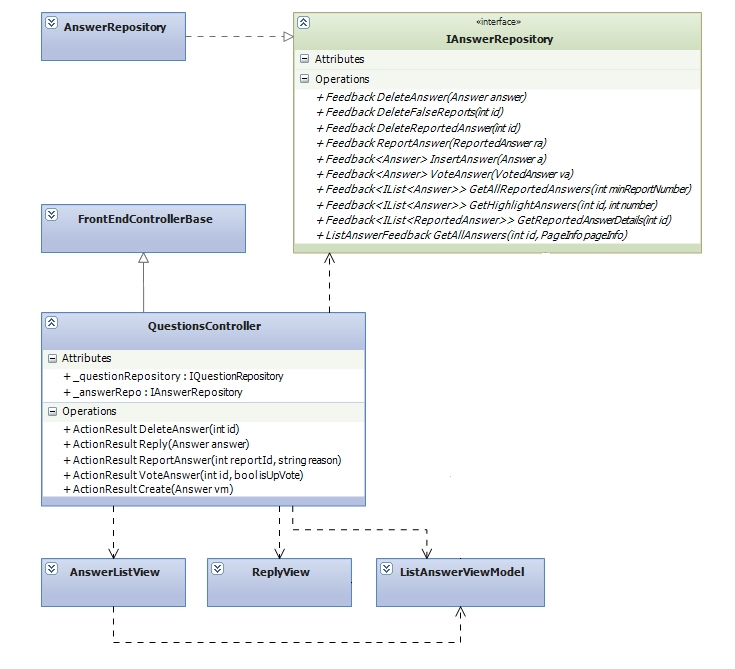


## Manage Spam Contents

This part is the detailed design for the following use cases:

* Delete False Content Reports
* Delete Reported Content
* Manage Reported Content

### Class Diagram



### Class Explanations

#### IContentRepository interface

Outlines the methods that the data repository class (ContentRepository) will be able to perform. The data repository class will expand this repository pattern interface to include a set of data access functions such as Select, Delete.

Responsibilities:

* Delete false content reports
* Delete reported content
* Get reported content

#### ContentRepository class

Default implementation of IContentRepository interface, contains the required code for accessing the MSSQL database

#### ManageContentsController class

The basic of responsibilities of ManageContentsControlleris to carry out the various use cases

Responsibilities:

* Allow administrator and power users to perform DeleteFalse Content Reports Use Case
* Allow administrator and power users to perform Delete Reported Content Use Case
* Allow administrator and power users to perform View Reported Content Use Case

#### ManageReportedContentView class

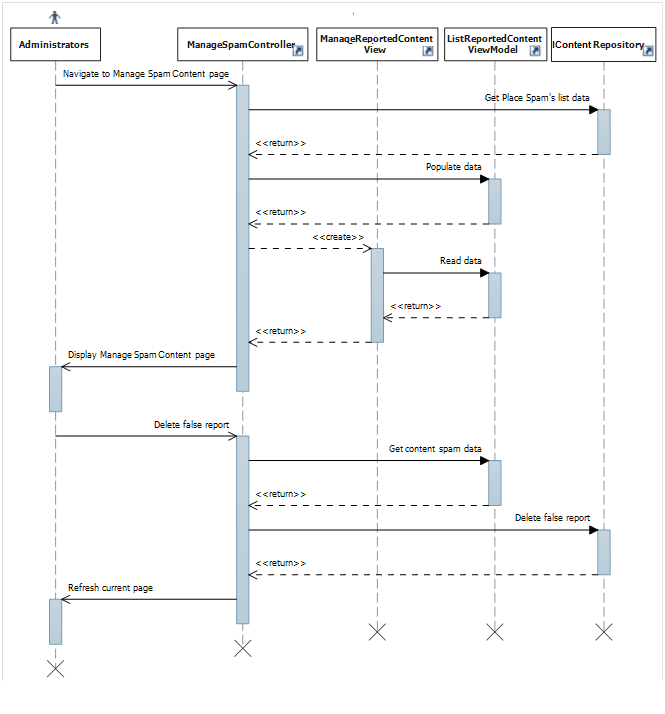
Contain HTML markup and .NET code to render Manage ReportedContent View screen

#### ListReportedContentViewModel class

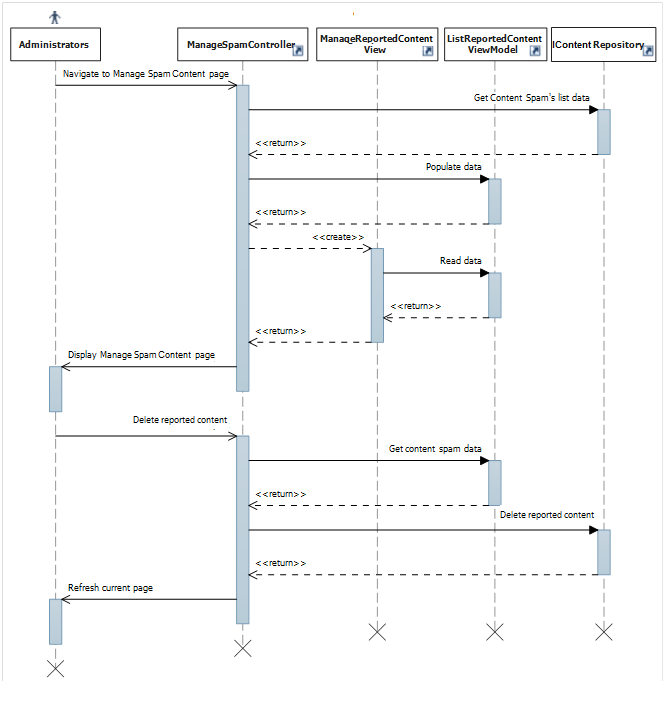
Model type of ManageReportedContentView

### Sequence Diagrams

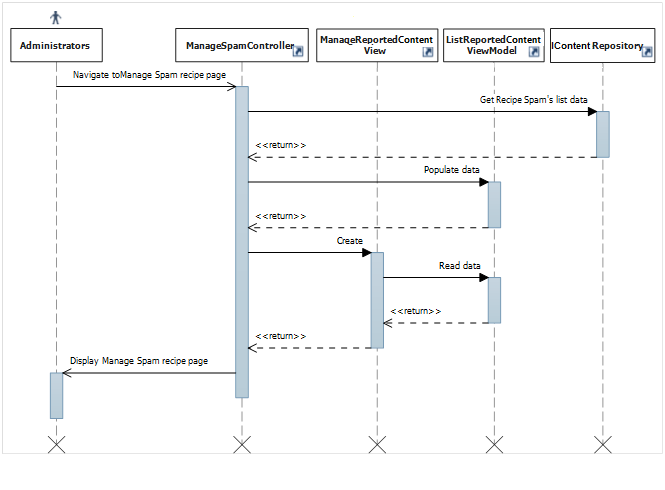
#### Delete False ContentReports



#### Delete Reported Content



#### Manage Reported Content

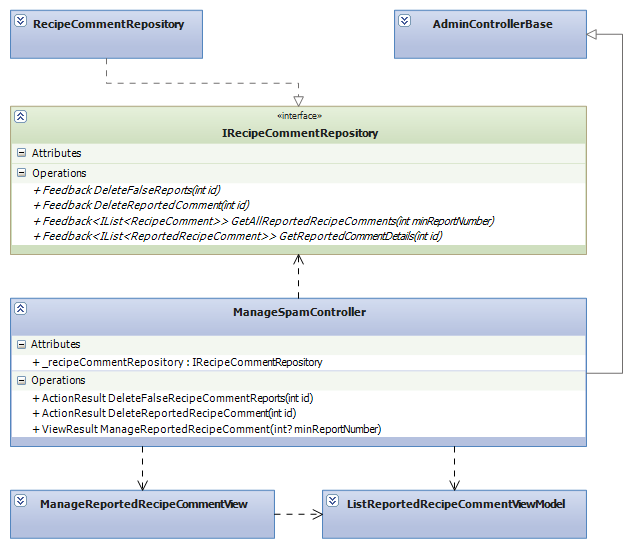


## Manage Spam Comments

This part is the detailed design for the following use cases:

* Delete False Comment Reports
* Delete Reported Comment
* Manage Reported Comment

### Class Diagram



### Class Explanations

#### IRecipeCommentRepository interface

Outlines the methods that the data repository class (RecipeCommentRepository) will be able to perform. The data repository class will expand this repository pattern interface to include a set of data access functions such as Select, Delete.

Responsibilities:

* Delete false comment reports
* Delete reported comment
* Get reported comment

#### RecipeCommentRepository class

Default implementation of IRecipeCommentRepository interface, contains the required code for accessing the MSSQL database

#### ManagSpamController class

The basic of responsibilities of ManageSpamControlleris to carry out the various use cases

Responsibilities:

* Allow administrator and power users to perform Delete False Comment Reports Use Case
* Allow administrator and power users to perform Delete Reported Comment Use Case
* Allow administrator and power users to perform View Reported Comment Use Case

#### ManageReportedRecipeCommentView class

Contain HTML markup and .NET code to render Manage Reported Recipe Comment View screen

#### ListReportedRecipeCommentViewModel class

Model type of ManageReportedRecipeCommentView

### Sequence Diagrams

#### Delete False Comment Reports



#### Delete Reported Comment



#### Manage Reported Comment

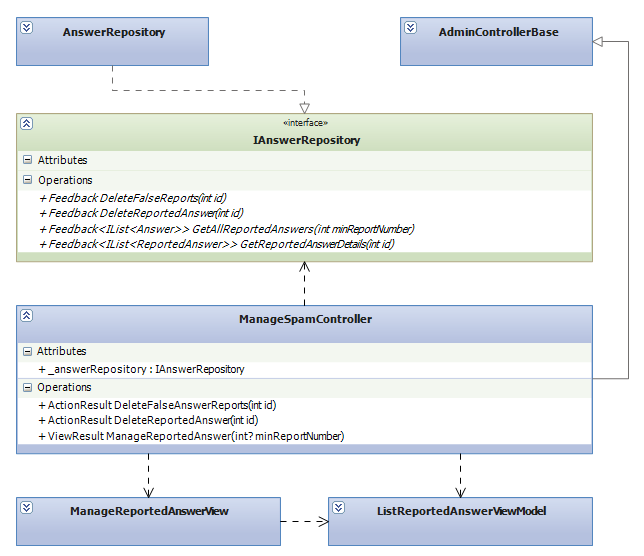


## Manage Spam Answers

This part is the detailed design for the following use cases:

* DeleteFalseAnswerReports
* DeleteReportedAnswer
* ManageReportedAnswer

### Class Diagram



### Class Explanations

#### IAnswerRepository interface

Outlines the methods that the data repository class (AnswerRepository) will be able to perform. The data repository class will expand this repository pattern interface to include a set of data access functions such as Select, Delete.

Responsibilities:

* Delete false answer reports
* Delete reported answer
* Get reported answer

#### AnswerRepository class

Default implementation of IAnswerRepository interface, contains the required code for accessing the MSSQL database

#### ManagSpamController class

The basic of responsibilities of ManageSpamControlleris to carry out the various use cases

Responsibilities:

* Allow administrator and power users to perform Delete False Answer Reports Use Case
* Allow administrator and power users to perform Delete Reported Answer Use Case
* Allow administrator and power users to perform View Reported Answer Use Case

#### ManageReportedAnswerView class

Contain HTML markup and .NET code to render Manage Answer Comment View screen

#### ListReportedAnswerCommentViewModel class

Model type of ManageReportedRecipeCommentView

### Sequence Diagrams

#### DeleteFalseAnswerReports



#### DeleteReportedAnswer



#### ManageReportedAnswer



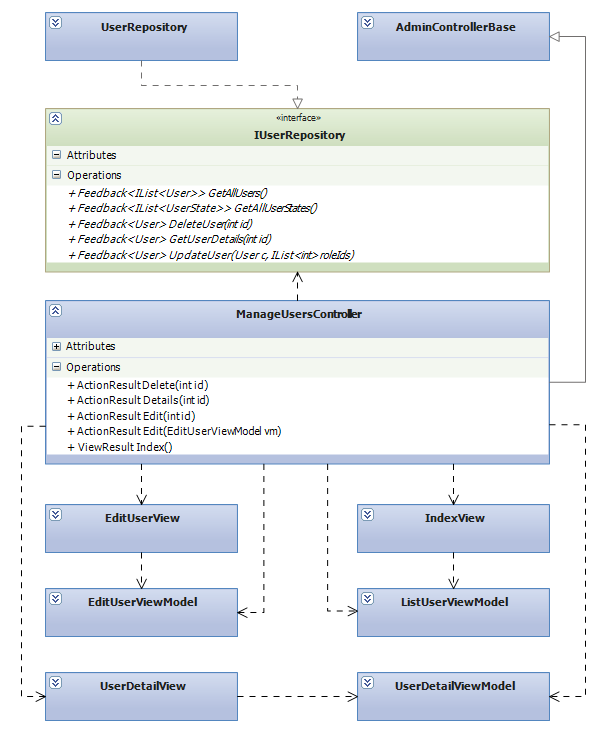
## Manager Users

This part is the detailed design for the following use cases:

* Edit user
* Ban user
* View user list



### Class Diagram



### Class Explanations

#### IUserRepository interface

Outlines the methods that the data repository class (UserRepository) will be able to perform. The data repository class will expand this repository pattern interface to include a set of data access functions such as Select, Update, Delete.

Responsibilities:

* Edit user
* Get user list

#### UserRepository class

Default implementation of IUserRepository interface, contains the required code for accessing the MSSQL database

#### ManageUsersController class

The basic of responsibilities of ManageUsersControlleris to carry out the various use cases

Responsibilities:

* Allow administrator and power users to perform View User List Use Case
* Allow administrator and power users to perform Edit User Use Case

#### EditUserView class

Contain HTML markup and .NET code to render Edit User View screen

#### IndexView class

Contain HTML markup and .NET code to render Index View screen

#### UserDetailsView class

Contain HTML markup and .NET code to render User Details View screen

#### ListUserViewModel class

Model type of IndexView

#### UserDetailsViewModel class

Model type of UserDetailsView

#### EditUserViewModel class

Model type of EditView

### Sequence Diagrams

#### EditUser



#### ViewUserDetails



#### ViewUserList

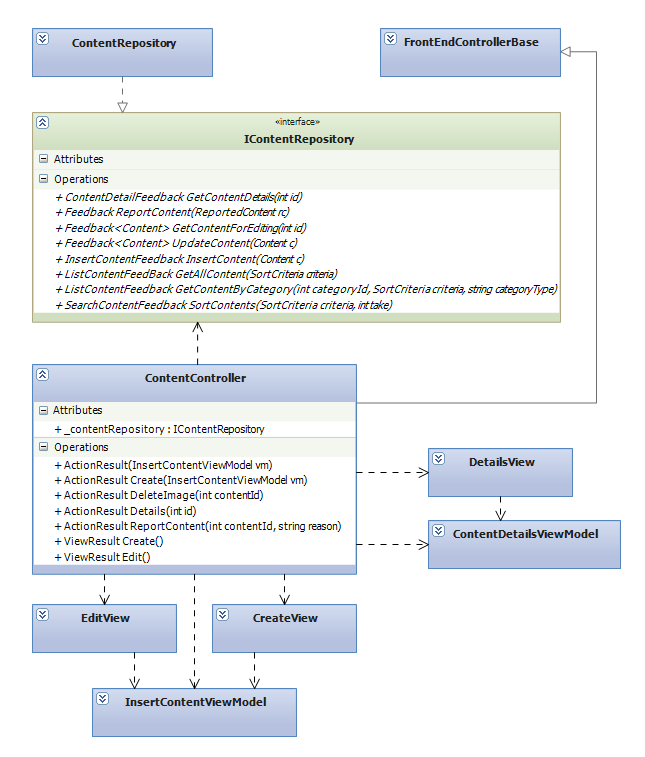


## Contents

### Class Diagram

This part is the detailed design for the following use cases:

* Get all contents
* Create a new content
* Edit contents
* Get contents by category
* Report contents
* Sort contents



### Class Explanations

#### IContentRepository interface

Outlines the methods that the data repository class (ContentRepository) will be able to perform. The data repository class will expand this repository pattern interface to include a full set of data access functions such as Select, Update, Delete, Insert.

Responsibilities:

* Get all contents
* Create a new content
* Edit contents
* Get contents by category
* Report contents
* Sort contents

#### ContentRepository class

Default implementation of IContentRepository interface, contains the required code for accessing the MSSQL database

#### ContentController class

The basic of responsibilities of ContentControlleris to carry out the various use cases

Responsibilities:

* Allow all users to perform View Contents Use Case
* Allow registered users to perform View Contents by Category Use Case
* Allow registered users to perform Create A New Content Use Case
* Allow registered users to perform Edit Contents Use Case
* Allow registered users to perform Report Contents Use Case

#### EditView class

Contain HTML markup and .NET code to render Edit View screen

#### CreateView class

Contain HTML markup and .NET code to render Create View screen

#### DetailsView class

Contain HTML markup and .NET code to render Details View screen

#### InsertViewModel class

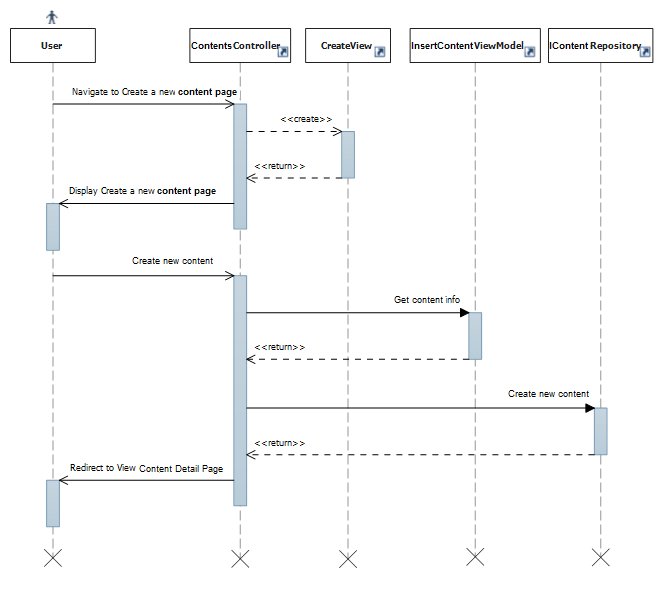
Model type of EditView and CreateView

#### ContentDetailsViewModel class

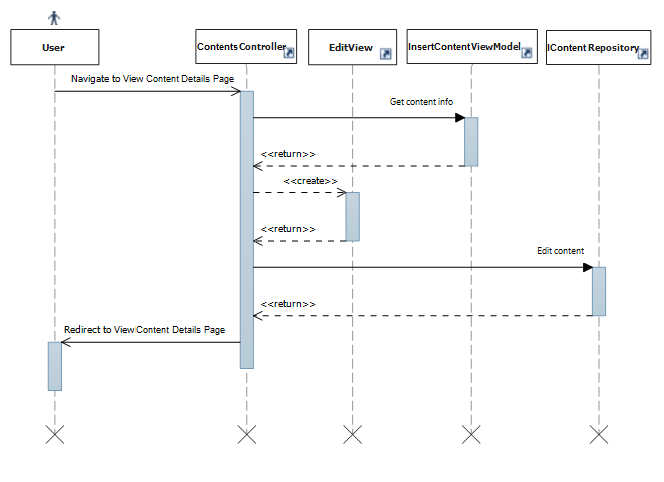
Model type of DetailsView

### Sequence Diagrams

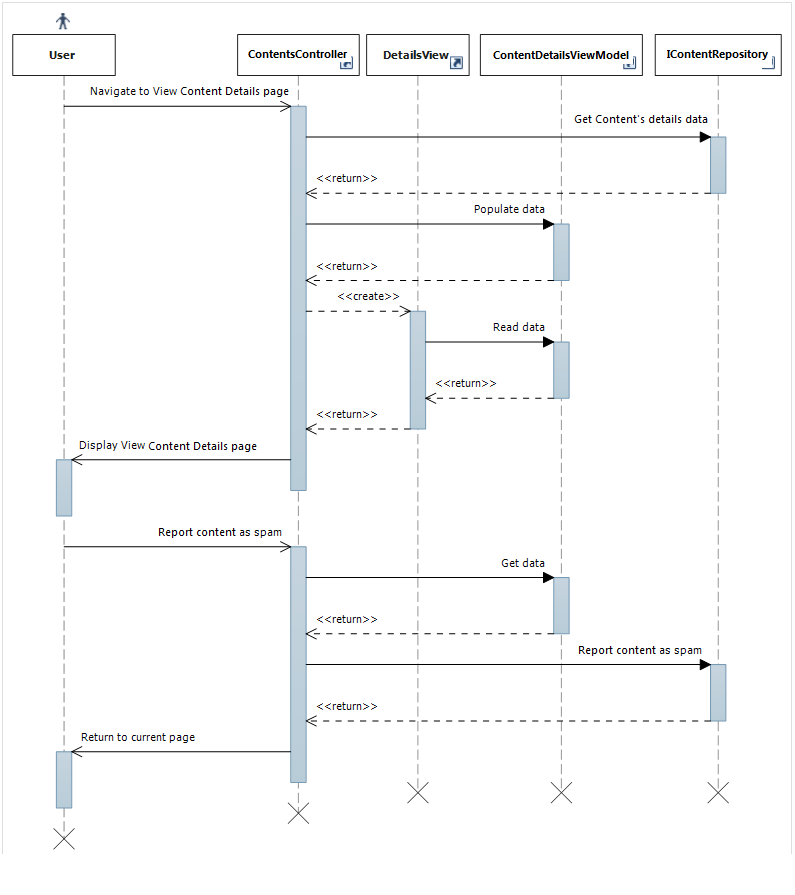
#### Create Content



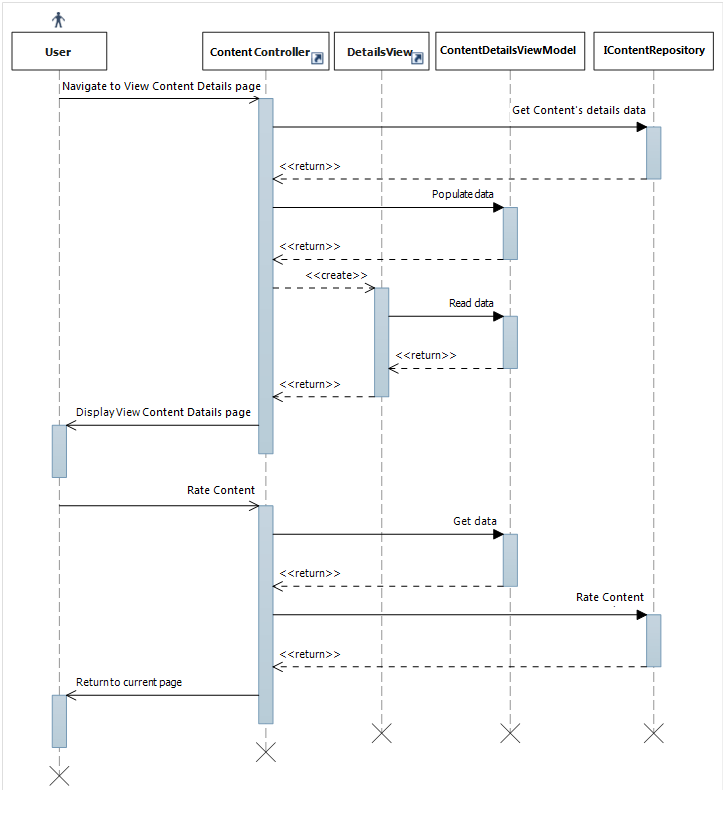
#### Edit Content



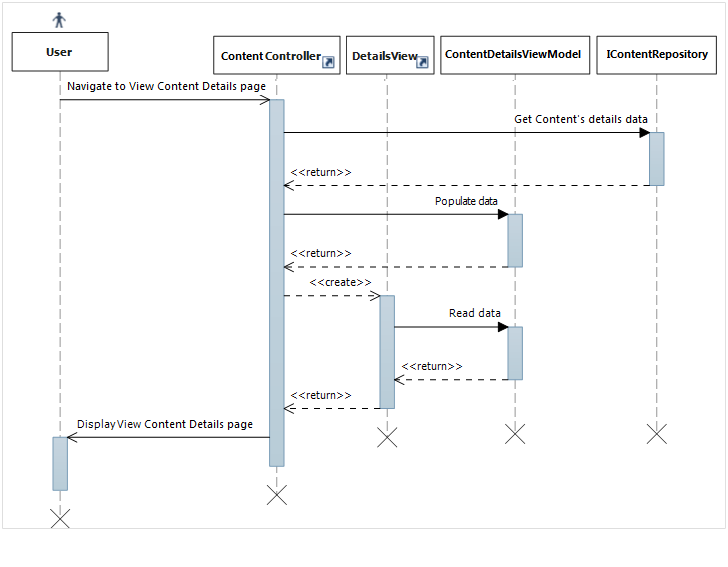
#### ReportContent



#### Rate Content



#### ViewContentDetails

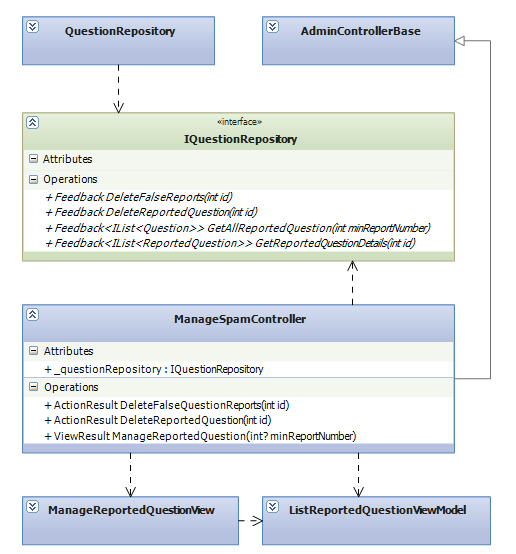


## Manage Spam Questions

This part is the detailed design for the following use cases:

* Delete False Question Reports
* Delete Reported Question
* Manage Reported Question

### Class Diagram



### Class Explanations

#### IQuestionRepository interface

Outlines the methods that the data repository class (QuestionRepository) will be able to perform. The data repository class will expand this repository pattern interface to include a set of data access functions such as Select, Delete.

Responsibilities:

* Delete false questionreports
* Delete reported question
* Get reported question

#### QuestionRepository class

Default implementation of IQuestionRepository interface, contains the required code for accessing the MSSQL database

#### ManageSpamController class

The basic of responsibilities of ManageSpamControlleris to carry out the various use cases

Responsibilities:

* Allow administrator and power users to perform Delete False Question Reports Use Case
* Allow administrator and power users to perform Delete Reported Question Use Case
* Allow administrator and power users to perform View Reported Question Use Case

#### ManageReportedQuestionView class

Contain HTML markup and .NET code to render Manage Reported Question View screen

#### ListReportedQuestionViewModel class

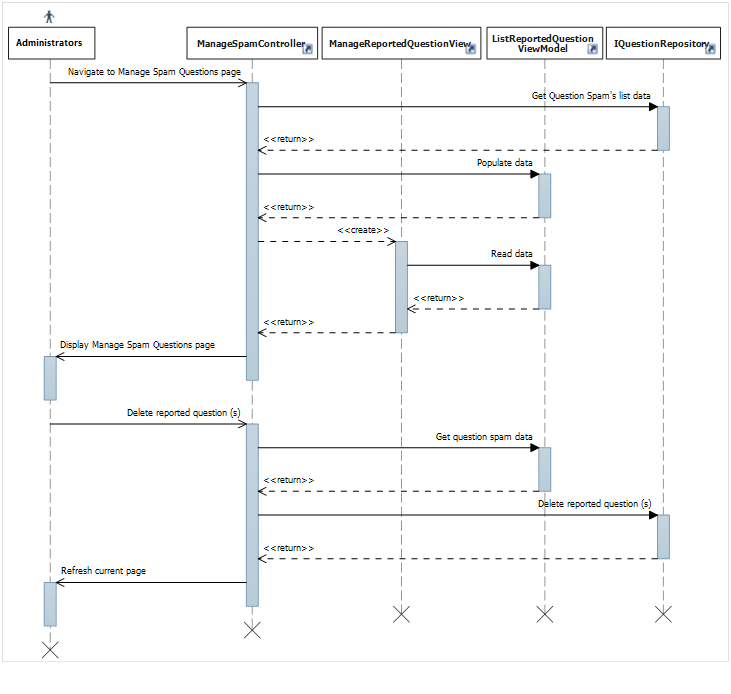
Model type of ManageReportedQuestionView

### Sequence Diagrams

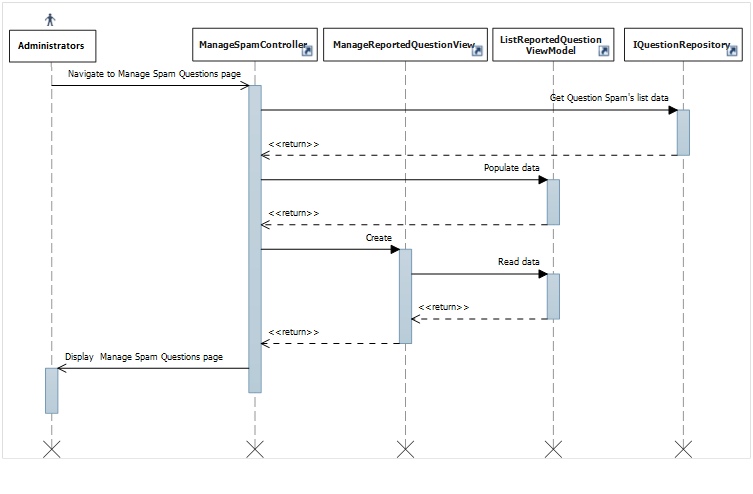
#### DeleteFalse Question Reports



#### DeleteReported Question



#### ManageReported Question



# Database Design

## Database Design

Refer to attached Database Design

## Tables

|  |  |
| --- | --- |
| Table Name | Description |
| Answers | Stores answers of questions |
| FavoriteRecipes | Stores favorite recipe of users |
| Locations | Stores geological information of a location |
| [News](#Tables/Categories) | Stores news |
| [News](#Tables/CategoryAlternatives)Comments | Stores comments of news |
| Permissions | Stores permission in the system |
| [Point](#Tables/FavoritePlaces)Configs | Stores how much point user can earn when contributed |
| [Questions](#Tables/FavoritePlaces) | Stores users questions |
| RatedRecipes | Stores rated recipes and by who |
| [RecipeCategories](#Tables/Permissions) | Stores recipe’s categories |
| [RecipeCategoryMapping](#Tables/PlaceComments) | Stores mapping of recipe’s categories and small recipe’s categories |
| [RecipeComments](#Tables/PlaceComments) | Stores comments of recipes |
| [Recipes](#Tables/PlaceResponses) | Stores recipes |
| [ReportedAnswers](#Tables/PlaceRevisions) | Stores reported answers |
| [ReportedNews](#Tables/Places) | Stores reported news |
| ReportedNewsComments | Stores reported news comments |
| RepotedQuestions | Stores reported questions |
| ReportedRecipeComments | Stores reported recipe’s comments |
| Reported­Recipes | Stores reported recipes |
| ReportedTipComments | Stores reported tip’s comments |
| ReportedTip | Stores reported tips |
| RolePermissions | Stores role’s permissions of the system |
| Roles | Stores role in the system |
| TipCategories | Stores tip’s categories |
| TipComments | Stores tips comments |
| Tips | Stores tips |
| Titles | Stores titles in the system |
| UpdateWords | Stores updated word from users |
| UserRoles | Stores users roles |
| Users | Stores information of users |
| User­States | Stores user account states |
| VisitorNumbers | Stores visitor numbers of contents |
| VisitorNumbers | Stores the number of visitors for each day |
| Voted­Answers | Stores voted answera and who has voted |
| Voted­NewsComments | Stores voted news comments and who has voted |
| Voted­RecipeComments | Stores voted recipes comments and who has voted |
| Voted­TipComments | Stores voted tips comments and who has voted |
| Voted­Tips | Stores voted tips and who has voted |
| Words | Stores keyword and meaning |

### Answers table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of Answers  Identity 1 -1 |
| QuestionId | int |  |  | FK |  | Id of Question FK to Questions.id |
| ContentText | nvarchar |  |  |  |  | Answers content in text mode |
| UpVotes | int |  |  |  |  | Thumb up vote times |
| DownVotes | int |  |  |  |  | Thumb down vote times |
| Reports | int |  |  |  |  | Reported times |
| PostedBy | int |  |  | FK |  | Id of user posted the answer FK to Users.id |
| PostedDate | datetime |  |  |  |  | The posted date |
| UpdatedBy | int | x |  | FK |  | Id of user updated the answer FK to Users.id |
| LastUpdate | Datetime | X |  |  |  | The last updated date |
| isDeleted | bit |  |  |  |  | Flag to delete answer |

### 2.2 FavoriteRecipes table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of Favorite recipe record  Identity 1 -1 |
| UserId | int |  |  | FK |  | User id who add a recipe to favorite list FK to Users.id |
| RecipeId | int |  |  | FK |  | Recipe id is added FK to Recipes.id |
| AddedDate | datetime |  |  |  |  | The added date |

### 2.3 Locations table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of Locations  Identity 1 -1 |
| Name | Nvarchar(50) |  |  |  |  | Locations content |

### 2.4 News table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of News  Identity 1 -1 |
| Title | nvarchar(100) |  |  |  |  | Title of news |
| TitleSearch | nvarchar(100) |  |  |  |  | Converted string of title to search function |
| TitleUrl | nvarchar(100) |  |  |  |  | Title’s url of news |
| ImageUrl | nvarchar(1000) |  |  |  |  | Images url |
| ContentText | nvarchar(255) |  |  |  |  | Content of news in text mode |
| ContentHtml | ntext |  |  |  |  | Content of news in html mode |
| Views | int |  |  |  |  | View times |
| Comments | int |  |  |  |  | Comment times |
| Reports | int |  |  |  |  | Reported times |
| PostedBy | int |  |  | FK |  | Id of user posted the news FK to Users.id |
| PostedDate | datetime |  |  |  |  | The posted date |
| UpdatedBy | int | x |  | FK |  | Id of user updated the news FK to Users.id |
| LastUpdate | Datetime | X |  |  |  | The last updated date |
| isDeleted | bit |  |  |  |  | Flag to delete news |

### 2.5 NewsComments table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of news comment  Identity 1 -1 |
| NewsID | int |  |  | FK |  | News id is commented FK to News.id |
| ContentText | nvarchar(255) |  |  |  |  | Content of news comment in text mode |
| UpVotes | int |  |  |  |  | Thumb up vote times |
| DownVotes | int |  |  |  |  | Thumb down vote times |
| Reports | int |  |  |  |  | Reported times |
| PostedBy | int |  |  | FK |  | Id of user posted the news comment FK to Users.id |
| PostedDate | datetime |  |  |  |  | The posted date |
| UpdatedBy | int | x |  | FK |  | Id of user updated the news comment FK to Users.id |
| LastUpdate | Datetime | X |  |  |  | The last updated date |
| isDeleted | bit |  |  |  |  | Flag to delete news comment |

### 2.6 Permissions table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of Permissions  Identity 1 -1 |
| Name | nvarchar(100) |  |  |  |  | Permissions name |
| Description | nvarchar(1000) | x |  |  |  | Description of permissions |

### 2.7 PointConfigs table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of configuration Identity 1 -1 |
| [Key] | nvarchar(100) |  |  |  |  | Configuration content |
| Value | int |  |  |  |  | Configuration point value |
| Description | nvarchar(1000) | x |  |  |  | Description of configuration |

### 2.8 Questions table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of questions  Identity 1 -1 |
| Title | nvarchar(100) |  |  |  |  | Title of questions |
| TitleUrl | nvarchar(100) |  |  |  |  | Title’s url of questions |
| ContentText | nvarchar(255) |  |  |  |  | Content of question in text mode |
| Views | int |  |  |  |  | Views times |
| Answers | int |  |  |  |  | Answer times of questions |
| Reports | int |  |  |  |  | Reported times |
| PostedBy | int |  |  | FK |  | Id of user posted the question FK to Users.id |
| PostedDate | datetime |  |  |  |  | The posted date |
| UpdatedBy | int | x |  | FK |  | Id of user updated the question FK to Users.id |
| LastUpdate | Datetime | X |  |  |  | The last update date |
| isDeleted | bit |  |  |  |  | Flag to delete question |

### 2.9 RatedRecipes table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of rated recipes  Identity 1 -1 |
| RecipeID | int |  |  | FK |  | Recipe’s id is rated FK to Recipes.id |
| RatedBy | int |  |  | FK |  | Id of user rated the question FK to Users.id |
| RatedDate | datetime |  |  |  |  | The rated date |

### 2.10 RecipeCategories table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of recipe’s categories  Identity 1 -1 |
| Name | nvarchar(50) |  |  |  |  | Name of recipe’s categories |
| NameUrl | nvarchar(50) |  |  |  |  | Name’s url of recipe’s categories |
| Description | nvarchar(100) | x |  |  |  | Description of recipe’s categories |
| isDeleted | bit |  |  |  |  | Flag to delete categories |

### 2.11 RecipeCategoryMapping table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of mapping  Identity 1 -1 |
| RecipeCategoryId | int |  |  | FK |  | Recipe categories be mapped FK to RecipeCategories.id |
| RecipeSmallCategoryId | int |  |  | FK |  | Recipe’s small categories be mapped FK to RecipeSmallCategories.id |

### 2.12 RecipeComments table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of recipes comment  Identity 1 -1 |
| RecipeId | int |  |  | FK |  | Recipe’s id is commented FK to Recipes.id |
| ContentText | nvarchar(4000) |  |  |  |  | Content of recipe’s comment |
| UpVotes | int |  |  |  |  | Thumb up vote times |
| DownVotes | int |  |  |  |  | Thumb down vote times |
| Reports | int |  |  |  |  | Reported times |
| PostedBy | int |  |  | FK |  | Id of user posted the recipes comment FK to Users.id |
| PostedDate | datetime |  |  |  |  | The posted date |
| UpdatedBy | int | x |  | FK |  | Id of user updated the recipes comment FK to Users.id |
| LastUpdate | Datetime | X |  |  |  | The last update date |
| isDeleted | bit |  |  |  |  | Flag to delete recipes comment |

### 2.13 Recipes table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of recipes  Identity 1 -1 |
| Title | nvarchar(100) |  |  |  |  | Title of recipes |
| TitleSearch | nvarchar(100) |  |  |  |  | Converted string of title to search function |
| TitleUrl | nvarchar(100) |  |  |  |  | Title’s url of news |
| ImageUrl | nvarchar(1000) |  |  |  |  | Images url |
| Ingredients | nvarchar(1000) |  |  |  |  | Recipes ingredients |
| ContentText | nvarchar(255) |  |  |  |  | Content of recipes in text mode |
| ContentHtml | ntext |  |  |  |  | Content of recipes in html mode |
| MappingCategories | int |  |  | FK |  | Mapping categories id FK to RecipeCategoryMappings.id |
| RateTimes | int |  |  |  |  | Rate times of recipes |
| TotalRates |  |  |  |  |  | Total rate point of recipes |
| Views | int |  |  |  |  | View times |
| Comments | int |  |  |  |  | Comment times |
| Reports | int |  |  |  |  | Reported times |
| PostedBy | int |  |  | FK |  | Id of user posted the recipe FK to Users.id |
| PostedDate | datetime |  |  |  |  | The posted date |
| UpdatedBy | int | x |  | FK |  | Id of user updated the recipe FK to Users.id |
| LastUpdate | Datetime | X |  |  |  | The last updated date |
| IsApproved | bit |  |  |  |  | Flag to approve news |
| ApprovedBy | int |  |  | FK |  | Id of user approved the recipe FK to Users.id |
| isDeleted | bit |  |  |  |  | Flag to delete recipe |

### 2.14 [RecipeSmallCategories](#Tables/PlaceRevisions) table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of [recipe’s small categories](#Tables/PlaceRevisions)  Identity 1 -1 |
| Name | nvarchar(50) |  |  |  |  | Name of [recipe’s small categories](#Tables/PlaceRevisions) |
| NameUrl | nvarchar(50) |  |  |  |  | Name’s url of [recipe’s small categories](#Tables/PlaceRevisions) |
| Description | nvarchar(100) | x |  |  |  | Description of [recipe’s small categories](#Tables/PlaceRevisions) |
| TipMappingId | int |  |  | FK |  | Mapped tip’s id FK to Tips.id |
| isDeleted | bit |  |  |  |  | Flag to delete [recipe’s small categories](#Tables/PlaceRevisions) |

### 2.15 [Reported](#Tables/Places)Answers table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of [reported](#Tables/Places) answers  Identity 1 -1 |
| AnswerId | int |  |  | FK |  | Answer is reported FK to Answers.id |
| ReportedBy | int |  |  | FK |  | Reported users id FK to Users.id |
| ReportedDate | datetime |  |  |  |  | The reported time |
| Reason | nvarchar(100) | x |  |  |  | Report’s reason |
| IsIgnored | bit |  |  |  |  | Flag to ignore report |

### 2.16 ReportedNews table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of [reported](#Tables/Places) news  Identity 1 -1 |
| NewsId | int |  |  | FK |  | News is reported FK to News.id |
| ReportedBy | int |  |  | FK |  | Reported users id FK to Users.id |
| ReportedDate | datetime |  |  |  |  | The reported time |
| Reason | nvarchar(100) | x |  |  |  | Report’s reason |
| IsIgnored | bit |  |  |  |  | Flag to ignore report |

### 2.17 ReportedNewsComments table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of [reported](#Tables/Places) news comments  Identity 1 -1 |
| CommentId | int |  |  | FK |  | News comment is reported FK to NewsCommets.id |
| ReportedBy | int |  |  | FK |  | Reported users id FK to Users.id |
| ReportedDate | datetime |  |  |  |  | The reported time |
| Reason | nvarchar(100) | x |  |  |  | Report’s reason |
| IsIgnored | bit |  |  |  |  | Flag to ignore report |

### 2.18 ReportedQuestions table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of [reported](#Tables/Places) questions  Identity 1 -1 |
| QuestionId | int |  |  | FK |  | Question is reported FK to Questions.id |
| ReportedBy | int |  |  | FK |  | Reported users id FK to Users.id |
| ReportedDate | datetime |  |  |  |  | The reported time |
| Reason | nvarchar(100) | x |  |  |  | Report’s reason |
| IsIgnored | bit |  |  |  |  | Flag to ignore report |

### 2.19 Reported­RecipeComments table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of reported recipe’s comment  Identity 1 -1 |
| CommentId | int |  |  | FK |  | Recipe’s comment is reported  FK to RecipeComments.Id |
| ReportedBy | int |  |  | FK |  | User reported  FK to Users.Id |
| ReportedDate | datetime |  |  |  |  | The reported date |
| Reason | nvarchar(100) | x |  |  |  | Report’s reason |
| IsIgnored | bit |  |  |  |  | Flag to ignore report |

### 2.20 ReportedRecipes table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of reported recipes  Identity 1 -1 |
| RecipeId | int |  |  | FK |  | Recipe is reported  FK to Recipes.Id |
| ReportedBy | int |  |  | FK |  | User reported  FK to Users.Id |
| ReportedDate | datetime |  |  |  |  | The reported date |
| Reason | nvarchar(100) | x |  |  |  | The reason why user reported |
| IsIgnored | bit |  |  |  |  | Flag to ignore report |

### 2.21 Reported­TipComments table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of reported tip’s comment  Identity 1 -1 |
| CommentId | int |  |  | FK |  | Tip’s comment is reported  FK to TipComments.Id |
| ReportedBy | int |  |  | FK |  | User reported  FK to Users.Id |
| ReportedDate | datetime |  |  |  |  | The reported date |
| Reason | nvarchar(100) | x |  |  |  | Report’s reason |
| IsIgnored | bit |  |  |  |  | Flag to ignore report |

### 2.22 Reported­Tips table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of reported tips  Identity 1 -1 |
| CommentId | int |  |  | FK |  | Tip is reported  FK to Tips.Id |
| ReportedBy | int |  |  | FK |  | User reported  FK to Users.Id |
| ReportedDate | datetime |  |  |  |  | The reported date |
| Reason | nvarchar(100) | x |  |  |  | Report’s reason |
| IsIgnored | bit |  |  |  |  | Flag to ignore report |

### 2.23 RolePermissions table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of Role­Permissions  Identity 1 -1 |
| PermissionId | int |  |  | FK |  | Permission matches with role  FK to Permissions.Id |
| RoleId | int |  |  | FK |  | Role matches with permission  FK to Roles.Id |

### 2.24 Roles table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of Roles  Identity 1 -1 |
| Name | nvarchar(100) |  |  |  |  | Name of Role |
| Description | nvarchar(1000) | x |  |  |  | Description of Role |

### 2.25 TipCategories table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of [tip’s categories](#Tables/PlaceRevisions)  Identity 1 -1 |
| Name | nvarchar(50) |  |  |  |  | Name of [tip’s categories](#Tables/PlaceRevisions) |
| NameUrl | nvarchar(50) |  |  |  |  | Name’s url of [tip’s categories](#Tables/PlaceRevisions) |
| Description | nvarchar(100) | x |  |  |  | Description of [tip’s categories](#Tables/PlaceRevisions) |
| isDeleted | bit |  |  |  |  | Flag to delete [tip’s categories](#Tables/PlaceRevisions) |

### 2.26 TipComments table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of tips comment  Identity 1 -1 |
| TipId | int |  |  | FK |  | Tip’s id is commented FK to Recipes.id |
| ContentText | nvarchar(4000) |  |  |  |  | Content of tip comment |
| UpVotes | int |  |  |  |  | Thumb up vote times |
| DownVotes | int |  |  |  |  | Thumb down vote times |
| Reports | int |  |  |  |  | Reported times |
| PostedBy | int |  |  | FK |  | Id of user posted the tips comment FK to Users.id |
| PostedDate | datetime |  |  |  |  | The posted date |
| UpdatedBy | int | x |  | FK |  | Id of user updated the recipes comment FK to Users.id |
| LastUpdate | Datetime | X |  |  |  | The last update date |
| isDeleted | bit |  |  |  |  | Flag to delete tips comment |

### 2.27 Tips table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of tips  Identity 1 -1 |
| Title | nvarchar(100) |  |  |  |  | Title of news |
| TitleSearch | nvarchar(100) |  |  |  |  | Converted string of title to search function |
| TitleUrl | nvarchar(100) |  |  |  |  | Title’s url of tips |
| ImageUrl | nvarchar(1000) |  |  |  |  | Images url |
| ContentText | nvarchar(255) |  |  |  |  | Content of news in text mode |
| ContentHtml | ntext |  |  |  |  | Content of news in html mode |
| TipCategories | int |  |  | FK |  | Tip’s category FK to TipCategories.id |
| UpVotes | int |  |  |  |  | Thumb up vote times |
| DownVotes | int |  |  |  |  | Thumb down vote times |
| Views | int |  |  |  |  | View times |
| Comments | int |  |  |  |  | Comment times |
| Reports | int |  |  |  |  | Reported times |
| PostedBy | int |  |  | FK |  | Id of user posted the tips FK to Users.id |
| PostedDate | datetime |  |  |  |  | The posted date |
| UpdatedBy | int | x |  | FK |  | Id of user updated the tips FK to Users.id |
| LastUpdate | Datetime | X |  |  |  | The last updated date |
| isDeleted | bit |  |  |  |  | Flag to delete tips |

### 2.28 Titles table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of title Identity 1 -1 |
| [Key] | nvarchar(100) |  |  |  |  | Title content |
| Value | int |  |  |  |  | Title point value |
| Description | nvarchar(1000) | x |  |  |  | Description of title |

### 2.29 UpdateWords table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of Update words Identity 1 -1 |
| WordId | int |  |  | FK |  | Word’s id FK to Words.id |
| NewValue | nvarchar(100) |  |  |  |  | Word’s meaning content |
| UpdatedBy | int |  |  | FK |  | Id of user updated keyword FK to Users.id |
| UpdatedDate | datetime |  |  |  |  | The updated date |
| IsApproved | bit | x |  |  |  | Flag to approve updated word |
| ApprovedBy | int | x |  | FK |  | Id of user approved FK to Users.id |
| isDeleted | bit |  |  |  |  | Flag to delete updated word |

### 2.30 User­Roles table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of User­Roles  Identity 1 -1 |
| UserId | int |  |  | FK |  | User matches with role  FK to Users.Id |
| RoleId | int |  |  | FK |  | Role matches with user  FK to Roles.Id |

### 2.31 Users table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of Users  Identity 1 -1 |
| Username | nvarchar(30) |  |  |  |  | User’s name |
| Password | nvarchar(MAX) |  |  |  |  | Password of User |
| Ponit | Int |  |  |  |  | Point of User |
| [Level] | nvarchar(30) |  |  |  |  | User’s title |
| Email | nvarchar(100) |  |  |  |  | Email of User |
| AvatarUrl | nvarchar(1000) | x |  |  |  | User’s avatar image’s url |
| Birthday | Datetime | x |  |  |  | Birthday of user |
| Sex | nvarchar(20) | x |  |  |  | Sex of user |
| Hobbies | nvarchar(1000) | x |  |  |  | Hobbies of user |
| CookingExp | nvarchar(20) | x |  |  |  | User’s cooking experience |
| SocialNetworkUrl | nvarchar(1000) | x |  |  |  | Social network’s url of user |
| LocationId | int | x |  | FK |  | Location of User FK to Locations.Id |
| UserStateId | int |  |  | FK |  | State of User  FK to UserStates.Id |
| ResetPasswordKey | nvarchar(100) | x |  |  |  | Reset password key of User |
| LastLogin | datetime | x |  |  |  | Last login of User |
| FailedLoginAttempts | int |  |  |  |  | Failed Login times of User |
| LastLoginAttempted | datetime | x |  |  |  | Last Login Attempted of User |
| CreatedDate | datetime | x |  |  |  | The created date of User |
| LastUpdate | datetime | x |  |  |  | The last update date of User |
| IsDeleted | bit |  |  |  |  | Flag to delete User |
| TodayDate | datetime | x |  |  |  |  |
| TodayPoint | int | x |  |  |  |  |

### 2.32 User­States table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of User­States  Identity 1 -1 |
| Name | nvarchar(50) |  |  |  |  | Name of User­State |
| Description | nvarchar(1000) | x |  |  |  | Description of User­State |

### 2.33 VisitorNumbers table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of Visited­Places  Identity 1 -1 |
| Date | datetime |  |  |  |  |  |
| VisitorNumber | int |  |  |  |  |  |

### 2.34 Voted­Answers table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of Voted­Answers  Identity 1 -1 |
| AnswerId | int |  |  | FK |  | Answer is voted  FK to Answers.Id |
| VotedBy | int |  |  | FK |  | User voted  FK to Users.Id |
| VotedDate | datetime |  |  |  |  | The voted date |
| IsUpVote | bit |  |  |  |  | Flag to vote up or vote down |

### 2.35 VotedNews­Comments table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of Voted­Comments  Identity 1 -1 |
| CommentId | int |  |  | FK |  | News comment is voted  FK to NewsComments.Id |
| VotedBy | int |  |  | FK |  | User voted  FK to Users.Id |
| VotedDate | datetime |  |  |  |  | The voted date |
| IsUpVote | bit |  |  |  |  | Flag to vote up or vote down |

### 2.36 Voted­RecipeComments table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of Voted­Comments  Identity 1 -1 |
| CommentId | int |  |  | FK |  | Recipe’s comment is voted  FK to RecipeComments.Id |
| VotedBy | int |  |  | FK |  | User voted  FK to Users.Id |
| VotedDate | datetime |  |  |  |  | The voted date |
| IsUpVote | bit |  |  |  |  | Flag to vote up or vote down |

### 2.37 VotedTipComments table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of Voted­Comments  Identity 1 -1 |
| CommentId | int |  |  | FK |  | Tip’s comment is voted  FK to TipsComments.Id |
| VotedBy | int |  |  | FK |  | User voted  FK to Users.Id |
| VotedDate | datetime |  |  |  |  | The voted date |
| IsUpVote | bit |  |  |  |  | Flag to vote up or vote down |

### 2.38 VotedTips table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of Voted­Tips  Identity 1 -1 |
| TipId | int |  |  | FK |  | Tip’s comment is voted  FK to Tips.Id |
| VotedBy | int |  |  | FK |  | User voted  FK to Users.Id |
| VotedDate | datetime |  |  |  |  | The voted date |
| IsUpVote | bit |  |  |  |  | Flag to vote up or vote down |

### 2.39 Words table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Type | Null | Unique | P/F Key | Default | Description |
| Id | int |  |  | PK |  | Id of word Identity 1 -1 |
| [Key] | nvarchar(20) |  |  |  |  | Key content |
| Value | nvarchar(100) |  |  |  |  | Key’s meaning content |
| CreatedBy | int |  |  | FK |  | Id of user created keyword FK to Users.id |
| CreatedDate | datetime |  |  |  |  | The created date |
| IsApproved | bit | x |  |  |  | Flag to approve word |
| ApprovedBy | int | x |  | FK |  | Id of user approved FK to Users.id |
| isDeleted | bit |  |  |  |  | Flag to delete word |

# References

[1] Microsoft, ' ASP.NET MVC Framework '

<http://www.asp.net/mvc/whatisaspmvc>

[2] MSDN, ' UML Class Diagrams: Guidelines '

<http://msdn.microsoft.com/en-us/library/dd409416.aspx>

[3] MSDN, ' UML Sequence Diagrams: Guidelines '

<http://msdn.microsoft.com/en-us/library/dd409389.aspx>

[4] Wikipedia, ' Dependency Injection '

<http://en.wikipedia.org/wiki/Dependency_injection>