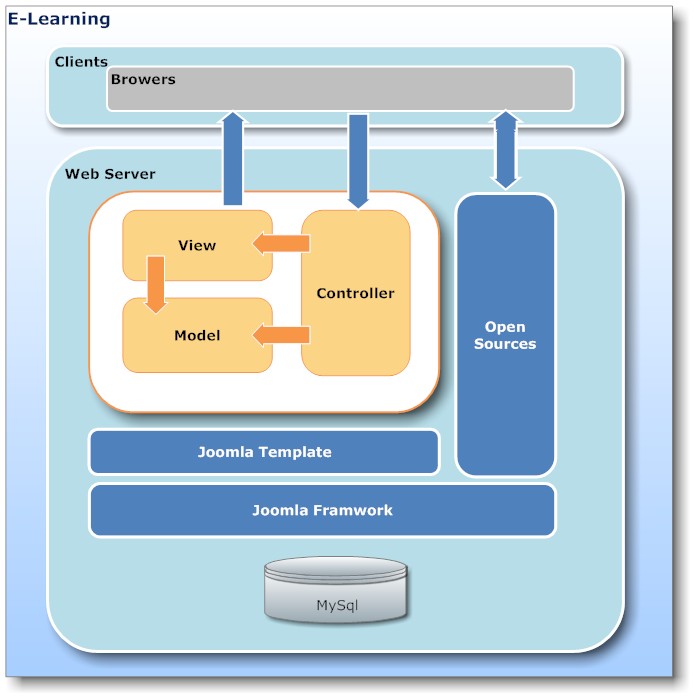
**Report No.4: Software Design Description (SDD)**

# 4.1. Design Overview

# 4.2. System Architectural Design

## 4.2.1 Choice of System Architecture

To approach the project, we choose the system architecture which is built as the diagram as below. This diagram will help us visually abstract the system and understand the key modules with their interaction in our E-Learning system.



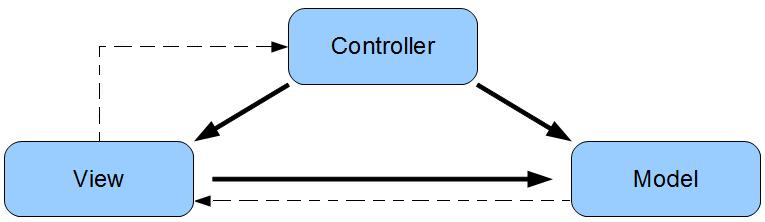
**Figure 4.1:** E-Learning system architecture design

**Client:** the client layer is user, user will access to web server by browser. The layer will send requests on the server. Then, the server will receive and return request for user.

**Web server:** the layer is where process and returns request to the client. After receiving request, the controller on web server will control the model to get data from MySQL server. Then, the controller sends methods of model to update the view and return the browser on client. The layer includes 5 modules:

**+ Model – View – Controller (MVC):**

The module includes common functions of the system: theories, question bank. We will develop modules of the system based on MVC model. The following diagram represents the Model-View-controller pattern:



**Figure 4.2:** MVC diagram

Model-View-Controller is a software design pattern that can be used to organize code in such a way that the business logic and data presentation are separate. The premise behind this approach is that if the business logic is grouped into one section, then the interface and user interaction that surrounds the data can be revised and customized without having to reprogram the business logic. MVC was originally developed to map the traditional input, processing, output roles into a logical GUI architecture.

### Model

The model is the part of the component that encapsulates the application's data. It will often provide routines to manage and manipulate this data in a meaningful way in addition to routines that retrieve the data from the model. In our case, the model will contain methods to add, remove and update information about the greetings in the database. It will also contain methods to retrieve the list of greetings from the database. In general, the underlying data access technique should be encapsulated in the model. In this way, if an application is to be moved from a system that utilizes a flat file to store its information to a system that uses a database, the model is the only element that needs to be changed, not the view or the controller.

### View

The view is the part of the component that is used to render the data from the model in a manner that is suitable for interaction. For a web-based application, the view would generally be an HTML page that is returned to the user. The view pulls data from the model (which is passed to it from the controller) and feeds the data into a template which is populated and presented to the user. The view does not cause the data to be modified in any way; it only displays data retrieved from the model.

### Controller

The controller is responsible for responding to user actions. In the case of a web application, a user action is (generally) a page request. The controller will determine what request is being made by the user and respond appropriately by triggering the model to manipulate the data appropriately and passing the model into the view. The controller does not display the data in the model, it only triggers methods in the model which modify the data, and then pass the model into the view which displays the data.

+ **Joomla Template:**

The module controls the overall look and layout of a site. It provides the framework that brings together common elements, modules and components as well as providing the cascading style sheet for the site. Templates are managed with the Template Manager, which is located on the Extensions menu in the back-end (administrator) area of our site. The way Joomla! is designed separates out the key tasks involved in producing a website for efficient maintenance of the software. One of these tasks is to create the aesthetic (the look, feel and layout) of the site. This includes making decisions such as which content elements (components, modules and plugins) you may want to place in any given page.

A template is used to manipulate the way content is delivered to a web browser or screen reader. Here are some ways you can employ this on your Joomla!-powered site.

### Layout

The template is the place where the design of the main layout is set for your site. This includes where you place different elements (components, modules, and plugins), which are responsible for different types of content. For example: The various menus (you can chose from existing options and create your own) Advertising banners Polls The main body of the page (you can select from different styles such as typical blog layout, a news article, etc.)

If the template is designed to provide choices, you can also "dynamically" alter the content placement on your site, perhaps putting the main menu on the right or left side of the screen.

### Color Scheme:

Using CSS within the template design, you can change the colors of your backgrounds, text, links or just about anything that you could within your ordinary (X)HTML code.

### Images and Effects:

You can also control the way images are displayed on the page, and even create flash-like effects or include [AJAX](http://en.wikipedia.org/wiki/AJAX) applications such as drop-down menus.

### Fonts:

The same applies to fonts. The designs for these are all set within the template's CSS file(s) to create a uniform look across your entire site, which makes it fantastically easy to change the whole look just by altering one or two files rather than every single page.

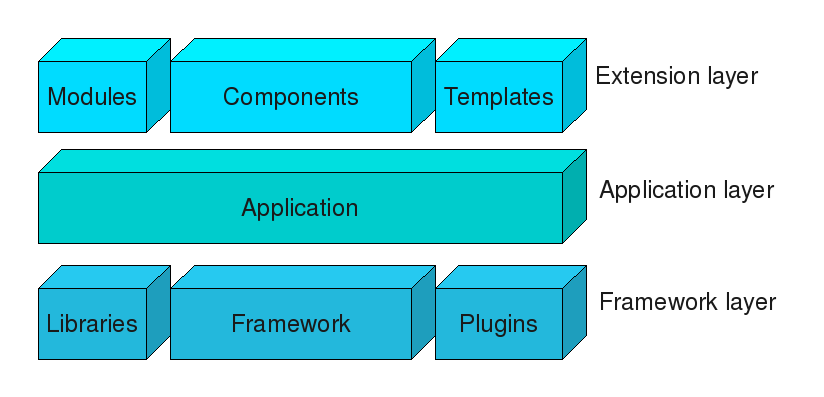
### Browser Specific Solutions:

A template can be designed to alter how it displays on different web browsers, allowing you to take full advantage of the latest developments without making your site inaccessible to those who are not able to run "up-to-the-minute" system upgrades (such as certain companies who limit what software their employees can use).

**+ Joomla Framework:**

The Joomla Framework is an important part of the Joomla architecture. It's based on modern object-oriented design patterns that make the Joomla core highly maintainable and easily extendable.

Third party developers benefit from the rich, and easily accessible functionality that the Joomla Framework provides. On this page we'd like to provide you a reference of all classes and respective methods. The links will take you to further information about each class including, where possible, examples of use.

****

Joomla is a three tiered system:-

* The top, Extensions layer, consists of [Extensions](http://docs.joomla.org/Extension) to the Joomla [Framework](http://docs.joomla.org/Framework) and its applications:
  + - [Modules](http://docs.joomla.org/Module)
    - [Components](http://docs.joomla.org/Component)
    - [Templates](http://docs.joomla.org/Template)
* The middle, Application layer, consists of applications that extend the Framework [*JApplication*](http://docs.joomla.org/JApplication) class. There are three applications included in the Joomla distribution:
  + - [*JInstallation*](http://docs.joomla.org/JInstallation) is responsible for installing Joomla on a web server and is deleted after the installation procedure has been completed.
    - [*JAdministrator*](http://docs.joomla.org/index.php?title=JAdministrator&action=edit&redlink=1) is responsible for the back-end Administrator.
    - [*JSite*](http://docs.joomla.org/JSite) is responsible for the front-end of the website.
* The bottom, Framework layer, consists of:
  + - The Joomla [Framework](http://docs.joomla.org/Framework) itself, whose classes are listed below.
    - [Libraries](http://docs.joomla.org/Library) that are required by the [Framework](http://docs.joomla.org/Framework) or are installed for use by third-party developers.
    - [Plugins](http://docs.joomla.org/Plugin) extend the functionality available in the [Framework](http://docs.joomla.org/Framework).

**+ Open sources:**

We will use open sources: iGiveTest and PhpBB3. **iGiveTest** is a comprehensive solution for creating, administering, and providing thorough analysis of tests on the Internet and Intranet. **phpBB** is a free flat-forum bulletin board software solution that can be used to stay in touch with a group of people or can power your entire website. No other bulletin board software offers a greater complement of features, while maintaining efficiency and ease of use. Best of all, phpBB is **completely** **free**. We welcome you to [test it](http://www.phpbb.com/demo/?sid=3abfae8fe8e7858d29ca16157e6d176c) for yourself today. If you have any questions please visit our [Community Forum](http://www.phpbb.com/community/?sid=3abfae8fe8e7858d29ca16157e6d176c) where our staff and members of the community will be happy to assist you with anything from configuring the software to modifying the code for individual needs.

We use phpBB version 3.0 to develop forum’s module and iGiveTest to develop test’s module.

**+ MySQL:**

MySQL is the world's most popular open source database. Whether you are a fast growing web property, technology ISV or large enterprise, MySQL can cost-effectively help you deliver high performance, scalable database applications. MySQL is a popular choice of database for use in web applications, and is a central component of the widely used [LAMP](http://en.wikipedia.org/wiki/LAMP_(software_bundle)) open source web application software stack—LAMP is an acronym for "[Linux](http://en.wikipedia.org/wiki/Linux), [Apache](http://en.wikipedia.org/wiki/Apache_HTTP_Server), MySQL, [Perl](http://en.wikipedia.org/wiki/Perl)/[PHP](http://en.wikipedia.org/wiki/PHP)/[Python](http://en.wikipedia.org/wiki/Python_(programming_language))".

MySQL is an open source database management system and is used in some of the most frequently visited websites on the Internet, including [Flickr](http://en.wikipedia.org/wiki/Flickr) [Nokia.com](http://en.wikipedia.org/wiki/Nokia), [YouTube](http://en.wikipedia.org/wiki/YouTube).

[Free-software](http://en.wikipedia.org/wiki/Free_software)-open source projects that require a full-featured database management system often use MySQL. For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: [TYPO3](http://en.wikipedia.org/wiki/TYPO3), [Joomla](http://en.wikipedia.org/wiki/Joomla), [WordPress](http://en.wikipedia.org/wiki/WordPress), [phpBB](http://en.wikipedia.org/wiki/PhpBB), [MyBB](http://en.wikipedia.org/wiki/MyBB), [Drupal](http://en.wikipedia.org/wiki/Drupal)

We choose MySql is our database server because it is completely free. Besides, softwares which we choose as Joomla, PhpBB and iGivetest also use MySQL.

## 4.2.2 Discuss of choose designs

## 4.2.2 Description of System Interface

All screens on the “E-Learning” system use the same format. Since the header, footer, navigation, and toolbar are consistent, the only place where the content will change is in the front page. The area is used to display content available only to the specific page. It will also be used to display a general application error message if the website is unavailable.

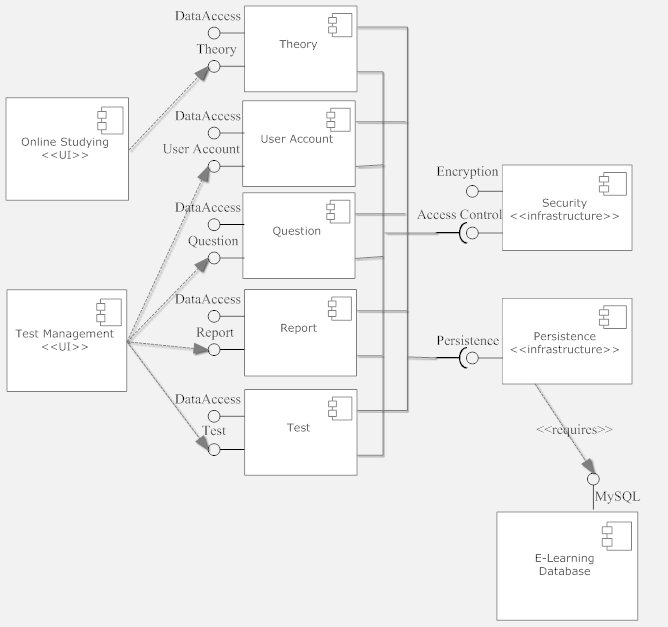
*Page Header*: It is displayed at the top of all pages. It contains logo, slogan and menu bar include a set of links about “Thư viện bài tập”, “Thư viện lý thuyết”, “Diễn đàn”, “Đề thi”, “Thành viên”. The header is standard on all “E-Learning” website.

*Navigation:* It displays relevant links to the categories of theory and question that exist on “E-Learning” system, and the affiliate links that are meant to add value to users.

*Page Footer:* It displays the following links:

* About Us
* Contact
* Policies
* Service
* Help

# 4.3. Component Diagram



**Figure 4.4:** Component diagram of “E-Learning” system.

# 4.4. Detailed Description of Components

## 4.4.1. Theory

**4.4.1.1. Theory – User Interface Design**

**4.4.1.1.1. Layout**

4.4.1.1.1.1. Layout – Screen Images



**Figure 4.5:** “E-Learning” layout page

4.4.1.1.1.2. Layout – Description of the User Interface

|  |  |  |
| --- | --- | --- |
| No | Name | Description |
| 1 | Menu bar | Display menu bar which include links to other sites of the website |
| 2 | Body content | Display the content for specific request of user. |
| 3 | “Home page” panel | The panel contains list of categories in “E-Learning” system. |
| 4 | Specify category | Click on the category to access to other sites of the system. |
| 5 | Login text boxes | Allow user to login to the system by entering username and password to text boxes. |
| 6 | Login button | Click on the button to send request to the system. |
| 7 | Forgot password button | Click on the button to send request to get new password. |
| 8 | Forgot username button | Click on the button to send request to find username. |
| 9 | Create account button | Click on the button to register to the system. |
| 10 | New topic area | The area includes links to new topic. |

**4.4.1.1.2. Concern question**

4.4.1.1.2.1. Concern question – Screen Images

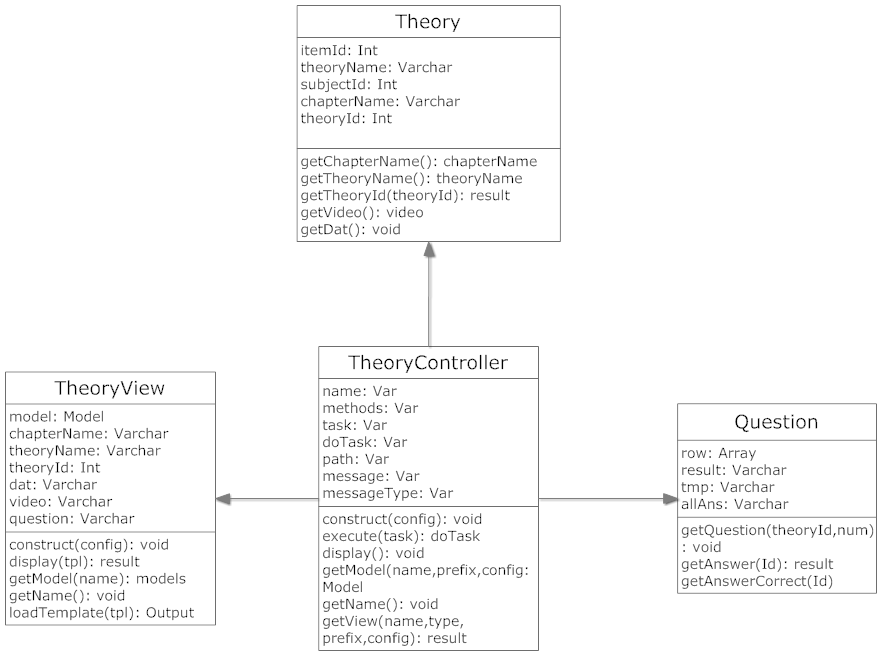


**Figure 4.6:** Question concern of theory panel

4.4.1.1.2.2. Concern question – Description of the User Interface

|  |  |  |
| --- | --- | --- |
| No | Name | Description |
| 1 | Question label | Display description of question concern of theory. |
| 2 | Question content | Show content of question. |
| 3 | Question answer | Show answers of the question |
| 4 | Case study | Show case study of the question |

**4.4.1.2. Theory – Class Diagram**



**Figure 4.7:** Theory’s class diagram

**4.4.1.3. Theory – Class Diagram Explanation**

**4.4.1.3.1. Theory class:**

4.4.1.3.1.1 Attributes:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | Itemid | Int | ID of theory, it is set to be unique. |
| 2 | TheoryName | Varchar | Theory name. |
| 3 | Subjectid | Int | ID of subject which has the theory. |
| 4 | Chapter\_name | Varchar | Name of chapter which has the theory. |
| 5 | FileVideoPath | Varchar | Path of video file (if has) of theory |
| 6 | FileDatPath | Varchar | File keeps content of theory. |

4.4.1.3.1.2. Methods:

* Method Get chapter:

Purpose: get chapter of subject which user selected.

Parameters & return:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | Subjectid | Int | ID of subject |
|  | <return> | None | None |



**Figure 4.8:** Sequence diagram for get chapter

* Method Get theory:

Purpose: get theory which user selected.

Parameters & return:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | Chapter\_name | Varchar | Name of chapter |
|  | <return> | None | None |

 **Figure 4.9:** Sequence diagram for get theory

**4.4.1.3.2. Question class:**

4.4.1.3.2.1 Attributes:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | Row | Array | Array of row to show question and answers. |
| 2 | Result | Varchar | Description of question which concern with theory. |
| 3 | Tmp | Varchar | Keep content of question. |
| 4 | AllAns | Varchar | Content of answers. |

4.4.1.3.2.2 Operations:

* Method Get question:
* Purpose: get question which concern of the theory user selected.
* Parameters & return:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | QuestionID | Int | ID of question |
| 2 | num | Int | Number of question |
|  | <return> | None | None |



**Figure 4.10:** Sequence diagram for get question

* Method Get answer:
  + Purpose: get answers which correspond with question.
  + Parameters & return:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | QuestionID | Int | ID of question |
|  | <return> | Result |  |



**Figure 4.11:** Sequence diagram for get answer

* Method Get answerCorrect:
  + Purpose: get answers is correct.
  + Parameters & return:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | Id | Int | ID of answer. |
|  | <return> | Result |  |

**4.4.1.3.3. TheoryView class:**

4.4.1.3.3.1 Attributes:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | model | Model | Instance of Model class. |
| 2 | chapterName | Varchar | Name of chapter which includes the theory. |
| 3 | theoryName | Varchar | Name of theory. |
| 4 | theoryId | Int | Id of theory. |
| 5 | Dat | Varchar | Content of theory. |
| 6 | Video | Varchar | Path of video file. |
| 7 | Question | Varchar | Content of question. |

4.4.1.3.3.2. Methods:

* Method Construct:
  + Purpose: set view name, layout, and charset used by the variable escaping functions.
  + Parameters & return:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | Config | Array | Array includes configuration of the system. |
|  | <return> | None | None |

* Method Display :
  + Purpose: Execute and display a template script, show chapter and content of theory.
  + Parameters & return:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | Tpl | Var | The name of the template files to parse. |
|  | <return> | Result |  |

* Method GetModel:
  + Purpose: method to get the model object.
  + Parameters & return:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | Name | Var | The name of the model (optional) |
|  | <return> | Methods |  |

* Method GetName:
  + Purpose: method to get the name of the model.
  + Parameters & Return:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
|  | <return> | Name |  |

* Method LoadTemplate:
  + Purpose: Load a template file -- first look in the templates folder for an override.
  + Parameters & Return:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | Tpl | Var | The name of the template source file. |
|  | <return> | Output | The output of the the template script. |

**4.4.1.3.4. TheoryController class:**

4.4.1.3.4.1 Attributes:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | Name | Var | Name of controller |
| 2 | Methods | Var | Array of class methods |
| 3 | Task | Var | Mapped task that was performed |
| 4 | DoTask | Var | Set of search directories for resources (views) |
| 5 | Path | Var | URL for redirection |
| 6 | Message | Var | Redirect message type |
| 7 | MessageType | Var | Section for the controller |

4.4.1.3.4.2. Methods:

* Method construct:
  + Purpose: recognized key values include “name”, “default task”, “model path” and “view path” (this list is not meant to be comprehensive).
  + Parameter & Return:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | Config | Array | Optional associative array of configuration setting |
|  | <return> | Void |  |

* Method execute:
  + Purpose: execute a task by triggering a method in the derived class.
  + Parameter & Return:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | Task | Var | Task to perform. If no matching task is found, the “default” task is executed, if defined. |
|  | <return> | retval |  |

* Method display:
  + Purpose: typical view method for MVC based architecture. This method is provided as a default implementation, in most cases we will need to override it in our own controllers.
  + Parameter & Return:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
|  | <return> | Void |  |

* Method getModel:
  + Purpose: to get model to implement.
  + Parameter & Return:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | Name | Var | Model name |
| 2 | Prefix | Var | Class prefix |
| 3 | Config | Array | Configuration array for model |
|  | <return> | Model | Object model |

* Method getName:
  + Purpose: method to get the controller name.
  + Parameter & Return:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
|  | <return> | Var | The name of the dispatcher |

* Method getView:
  + Purpose: method to get a reference to the current view and load it if necessary.
  + Parameter & Return:

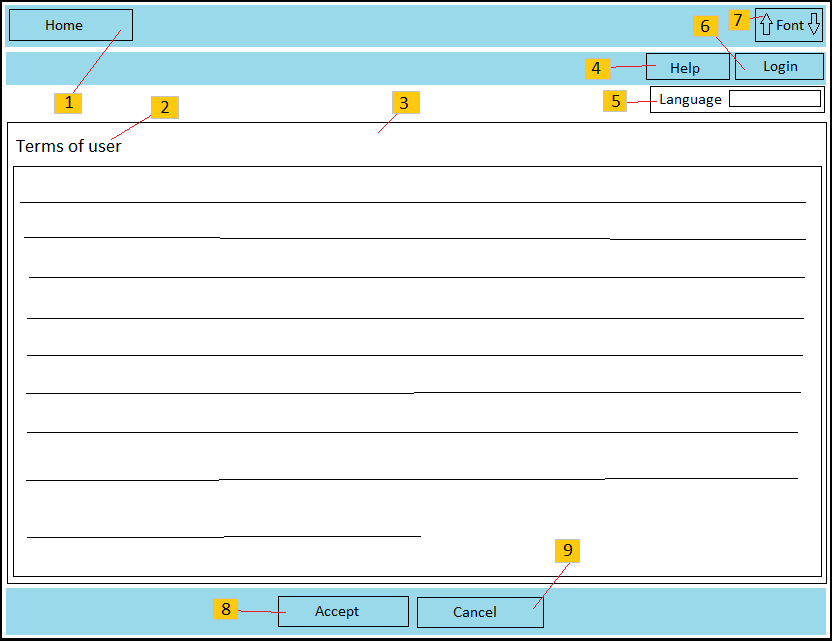
|  |  |  |  |
| --- | --- | --- | --- |
| No | Parameter | Type | Description |
| 1 | Name | Var | The view name, optional, defaults to the controller name. |
| 2 | Type | Var | The view type |
| 3 | Prefix | Var | The class prefix |
| 4 | Config | Array | Configuration array for view |
|  | <return> | Result | Reference to the view or an error |

## 4.4.2. User Account

**4.4.2.1. User Account – User Interface**

**4.4.2.1.1. Register**

4.4.2.1.1.1. Register - Screen Images



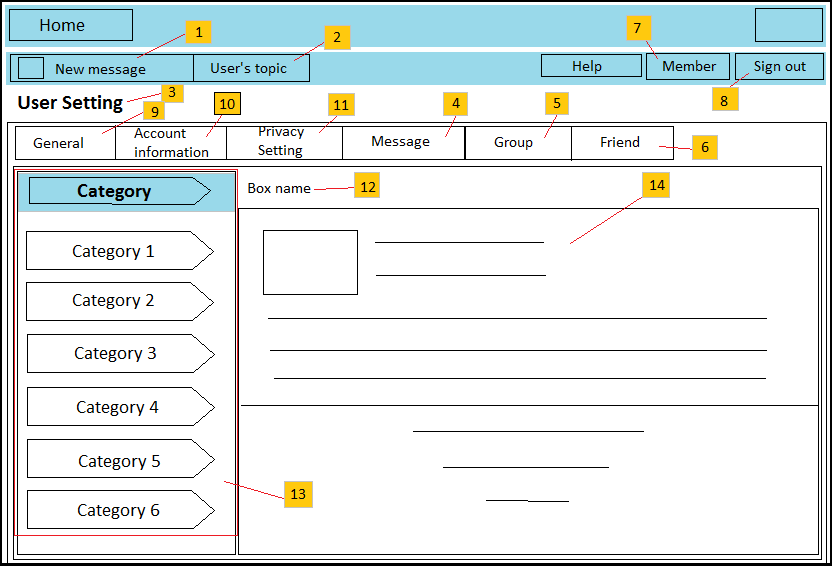
**Figure 4.12:** “E-Learning” register page

4.4.2.1.1.2. Register - Description of the User Interface

|  |  |  |
| --- | --- | --- |
| No | Name | Description |
| 1 | “Home page” link | Click on the link to access “Home page” site. |
| 2 | Register label | Show page where user is in. |
| 3 | “Term of users” content area | Display the content for “term of users” between user and “E-Learning” owners. |
| 4 | FAQ link | Show frequently asked questions can meet. |
| 5 | “Language” list box | Allow user to choose language to display the website. |
| 6 | Login link | Click on the link to access login site. |
| 7 | “Font size” button | Allow user to change font size of the website. Click on down arrow which is left to decrease font size and click on up arrow which is right to increase font size. |
| 8 | Agree button | Click on the button will allow user send the registration request to “E-Learning” system. |
| 9 | Disagree button | Click on the button to refuse to register. |

**4.4.2.1.2. User Setting**

4.4.2.1.2.1. User Setting - Screen images



**Figure 4.13:** “E-Learning” user setting page

4.4.2.1.2.2. User Setting - Description of the User Interface

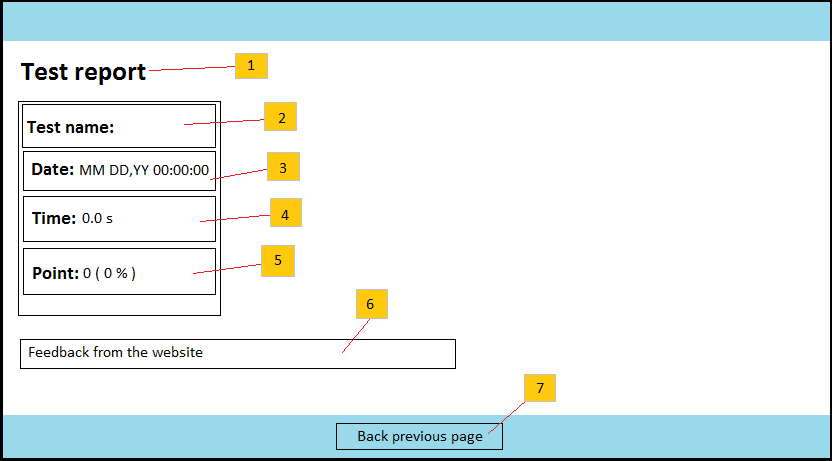
|  |  |  |
| --- | --- | --- |
| No | Name | Description |
| 1 | Message label | Show numbers of messages of user are read yet. |
| 2 | “Article of user” link | Click on the link to show articles of user. |
| 3 | User Setting label | Show the place where user is in. |
| 4 | Message box | Display messages of user and allow user to write message for another user. |
| 5 | Group box | Display group which user is joined. |
| 6 | Friend box | Display close friends and enemies of user. |
| 7 | Member link | Show all of members joined to the website. |
| 8 | Sign out button | User can sign out of the system when click on the button. |
| 9 | General box | Allow user to manage favorite page, draft article, attach file…. |
| 10 | Account Information box | Allow user to see and change privacy information. |
| 11 | Privacy setting box | Allow users to manage privacy system setting of them. |
| 12 | Box label | Display name of box where user is in. |
| 13 | Categories of box | Display categories of box where user is in. |
| 14 | Content of category | Display content of each of category. |

## 4.4.3. Report

**4.4.3.1. Report – User Interface**

**4.4.3.1.1. Default report**

4.4.3.1.1.1. Default report – Screen Images



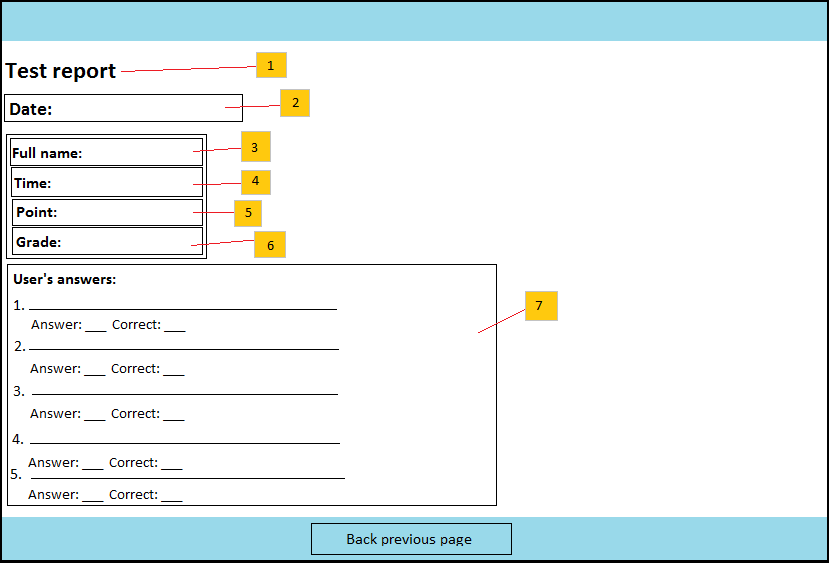
**Figure 4.14:** Test result page

4.4.3.1.1.1. Default report – Description of the User Interface

|  |  |  |
| --- | --- | --- |
| No | Name | Description |
| 1 | Report label | Display the page where user is in. |
| 2 | Test name | Display name of test which user did recently. |
| 3 | Test date | Display date and time of report is published. |
| 4 | Test time | Display time which user use to do the test. |
| 5 | Test mark | Display mark of test which user achieved. |
| 6 | Test feedback | Show feedback of the system about process which user did the test. |
| 7 | Previous page button | Click on the button to go back preparing test page. |

**4.4.3.1.2. Detail report**

4.4.3.1.2.1. Detail report – Screen Images



**Figure 4.15:** Detail result page

4.4.3.1.2.2. Detail report – Description of the User Interface

|  |  |  |
| --- | --- | --- |
| No | Name | Description |
| 1 | Report label | Display the page where user is in. |
| 2 | Test date | Display date and time of report is published. |
| 3 | Name of user | Show last name and first name of user. |
| 4 | Test time | Display time which user use to do the test. |
| 5 | Test mark |  |
| 6 | Test grade |  |
| 7 | Answer detail | Display answer of each question of the test which user choose. |

**4.4.3.2. Report – Class Diagram**

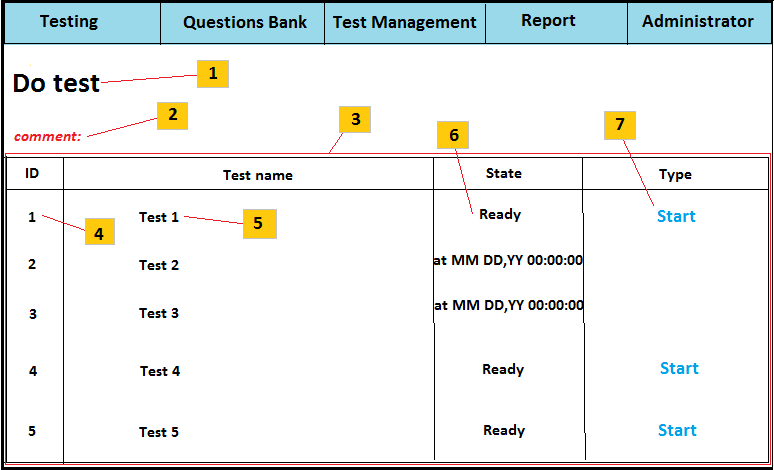
**4.4.3.3. Report – Class Diagram Explanation**

## 4.4.4. Test

**4.4.4.1. Test – User Interface**

**4.4.4.1.1. Preparing test**

4.4.4.1.1.1. Preparing test - Screen images



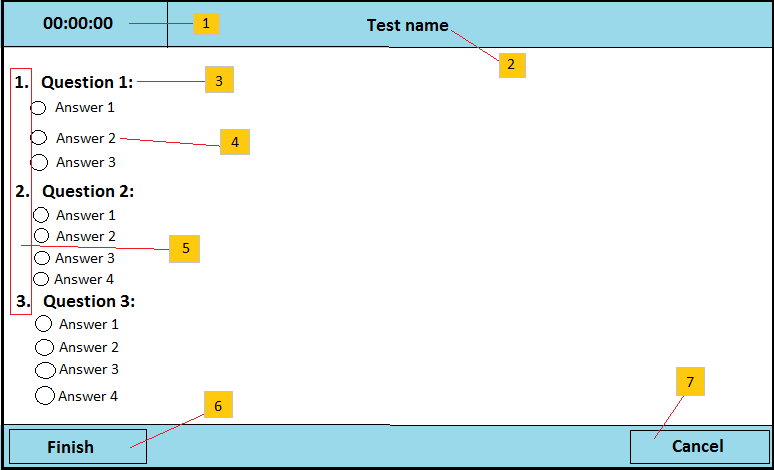
**Figure 4.16:** Test preparing page

4.4.4.1.1.2. Preparing test - Description of the User Interface

|  |  |  |
| --- | --- | --- |
| No | Name | Description |
| 1 | Test label | Show the place where user is in. |
| 2 | Test feedback | Show feedback if a test doesn’t have any question. |
| 3 | Test panel | The panel contains information of test |
| 4 | Test ID | Show ID of test |
| 5 | Test name | Show name of test |
| 6 | Test status | Show status of test. |
| 7 | Start button | Click on the button to start doing a test |

**4.4.4.1.2. Test process:**

4.4.4.1.2.1. Test process – Screen Images:



**Figure 4.17:** Test page

4.4.4.1.2.2. Test process - Description of the User Interface

|  |  |  |
| --- | --- | --- |
| No | Name | Description |
| 1 | Test time | Time to do the test; time will count down to 0. |
| 2 | Test name | Name of test |
| 3 | Name of question | Show name of question |
| 4 | Answers of question | Show answers of question |
| 5 | Order of questions | Display order of questions |
| 6 | Finish button | Click on button to finish the test. |
| 7 | Cancel button | Click on button to cancel the test. |

**4.4.4.2. Test – Class Diagram**

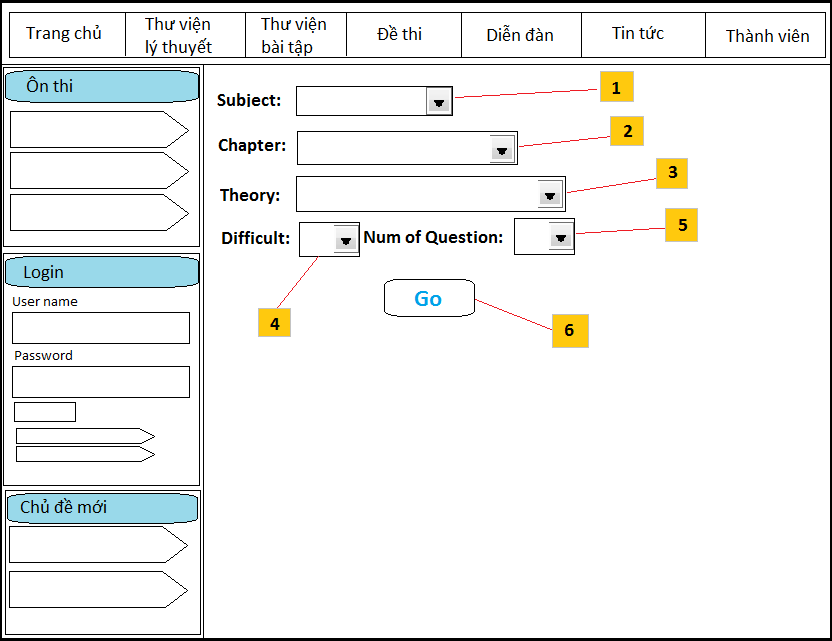
**4.4.4.3. Test – Class Diagram Explanation**

## 4.4.5. Question

**4.4.5.1. Question – User Interface**

**4.4.5.1.1. Choice Question**

4.4.5.1.1.1. Choice Question – Screen Images

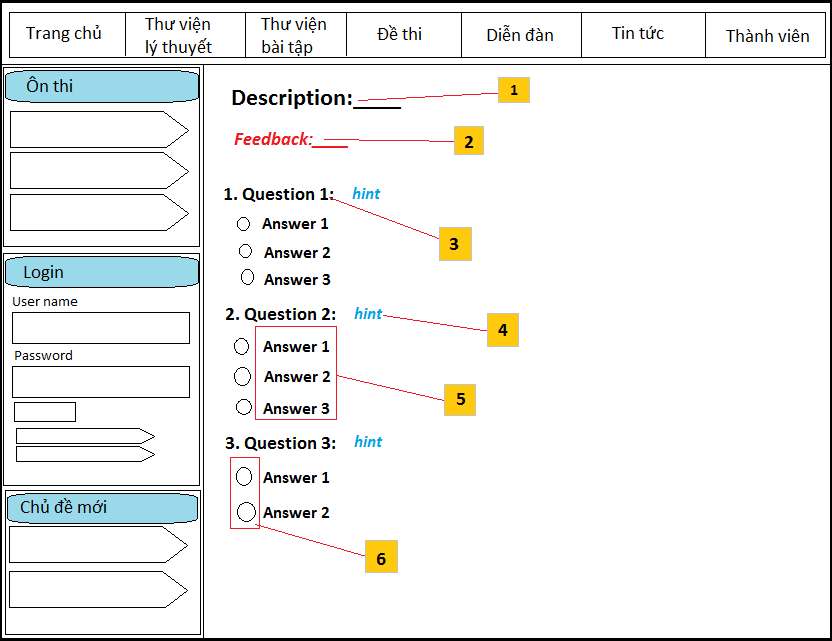


4.4.5.1.1.1. Choice Question – Description of the User Interface

|  |  |  |
| --- | --- | --- |
| No | Name | Description |
| 1 | List box subject | Allow user to choose subject. |
| 2 | List box chapter | Allow user to choose chapter of subject. |
| 3 | List box theory | Allow user to choose theory of chapter. |
| 4 | List box difficult | Allow user to choose difficult to do. |
| 5 | List box number of question | Allow user to choose number of question to do |
| 6 | Button Go | Click on button to start doing exercises. |

**4.4.5.1.2. Do exercise**

4.4.5.1.2.1. Do exercise – Screen Images



4.4.5.1.2.1. Do exercise – Description of the User Interface

|  |  |  |
| --- | --- | --- |
| No | Name | Description |
| 1 | Description label | Show description of exercise includes: subject, chapter, theory, difficult which user choose. |
| 2 | Feedback label | Show feedback of the system about the exercise. |
| 3 | Content of question | Content of question |
| 4 | Button hint | Click on button to show case study following to the question. |
| 5 | Content of answers | Content of answers of question. |
| 6 | Check box answer | Click on check box to choose answers of user. |

**4.4.5.2. Question – Class Diagram**

**4.4.5.3. Question – Class Diagram Explanation**

# 4.5. Database Design or Data Structures

## 4.5.1 Detailed database design for “E-Learning” system:

## 4.5.2 Table and columns description and explanation

1. Table Users:

This table includes information of users: user name, password, email, fullname, date of joining, date of login…The table include 60 columns, we only describe some of typical information.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | UserID | INT(10) | No | Auto increment |
| 2 | User\_Name | VARCHAR(255) | No | Using to login to the system |
| 3 | User\_Passhash | VARCHAR(32) | No | Password was encrypted by MD5 |
| 4 | User\_email | VARCHAR(255) | No | Email to register of user |
| 5 | User\_enable | TINYINT | No | Situation of user of the system |
| 6 | User\_joindate | INT | No | date which user register |
| 7 | User\_logindate | INT | No | Date which user login to the system |

1. Table Tests:

This table stores information of tests: test name, test time, date which test is started, shuffling question, number of user’s attempts…

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | TestID | INT(10) | No | Auto increment |
| 2 | SubjectID | INT(10) | No |  |
| 3 | Test\_Name | VARCHAR(255) | No | Name of test |
| 4 | Test\_Code | VARCHAR(255) | No | Code of test |
| 5 | Test\_Description | VARCHAR(255) | No | Brief description |
| 6 | Test\_Time | INT(10) | No | Time which test enable |
| 7 | Test\_datestart | INT | No | Date which test is started |
| 8 | Test\_dateend | INT | No | Date which test ends |
| 9 | Test\_shuffleq | TINYINT | No | Shuffling questions |
| 10 | Test\_shufflea | TINYINT | No | Shuffling answers |
| 11 | Test\_timeforceout | TINYINT | No | End test when the time limit is reached |
| 12 | Test\_attempts | INT | No | Number of attempts are allowed |
| 13 | Test\_showqfeedback | TINYINT | No | Showing feedback after choose each of answer |
| 14 | Test\_qsperpage | TINYINT | No | Number of questions are showed on each page |
| 15 | Test\_result\_showanswers | TINYINT | No | Showing number of questions are correct |
| 16 | Test\_result\_showpoints | TINYINT | No | Showing point of test after finishing |
| 17 | Test\_result\_showgrade | TINYINT | No | Showing grade of test after finishing |
| 18 | Test\_result\_showgradefeedback | TINYINT | No | Showing grade feedback of test after finishing |
| 19 | Test\_result\_showhtml | TINYINT | No | Showing report about result by HTML format |
| 20 | Test\_result\_showpdf | TINYINT | No | Showing report about result by .pdf format |
| 21 | Test\_result\_rtemplateid | INT(10) | No | templateID about report of result. |
| 22 | Test\_reportgradecondition | TINYINT | No | Condition about grade to show report. |
| 23 | Test\_prevtestid | INT | No | ID of previous test |
| 24 | Test\_nexttestid | INT | No | ID of next test |
| 25 | Test\_contentprotection | TINYINT | No | Allowing to protect by javascrip |
| 26 | Test\_notes | TEXT | No | Test notes for internal use |
| 27 | Test\_other\_repeatuntilcorrect | TINYINT | No | Repeat test until all questions are answered correctly |
| 28 | Test\_createdate | INT | No | Date which test are created |
| 29 | Test\_enable | TINYINT | No | State of test |

1. Table Questions:

This table is considered as question bank. It include information of questions: question time, content of question, point of question…

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | QuestionID | Int(10) | No | Auto increment |
| 2 | SubjectID | Int(10) | No | ID of subject about test |
| 3 | Question\_Time | Int(10) | No | Time which question is finished |
| 4 | Question\_Text | Text | No | Content of question |
| 5 | Question\_Points | Text | No | Point of question |
| 6 | Question\_Solution | Text | No | Solution of question |
| 7 | Question\_Type | Int(10) | No | Type of question |
| 8 | Question\_Type2 | Tinyint(3) | No | Allowing partially correct answers if Question\_Type is “Multiple Answer” |
| 9 | Question\_Shufflea | Tinyint(3) | No | Shuffle answers of question |
| 10 | Question\_Difficult | INT(10) | No | Difficult level of question |
| 11 | Theoryid | INT(10) | No | ID of theory which correspond with question. |

1. Table Answers:

This table stores answers of each question.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | AnswerID | Int(10) | No | Auto increment |
| 2 | QuestionID | Int(10) | No | ID of question of answer |
| 3 | Answer\_Text | Text | No | Content of answer |
| 4 | Answer\_Feedback | Text | No | Feedbacks of answer (if have) |
| 5 | Answer\_Correct | Tinyint(3) | No | Answer is correct |
| 6 | Answer\_Percents | float | No | Percentage correct of answer |
| 7 |  |  |  |  |

1. Table Results:

This table stores result about test of user. We use the table to count.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | ResultID | Int(10) | No | Auto increment |
| 2 | TestID | Int(10) | No | ID of test |
| 3 | UserID | Int(10) | No | ID of user which have report of test result |
| 4 | Result\_DateStart | Int(10) | No | Date which result of test is started |
| 5 | Result\_TimeSpend | Int(10) | No | Time which user spend to test |
| 6 | Result\_TimeExceeded | Tinyint(3) | No | Time which user exceed to test |
| 7 | Result\_Points | Float | No | Point of test of user |
| 8 | Result\_PointMax | Float | No | Max point of test |
| 9 | GscaleID | Int(10) | No | ID of Grading scale |
| 10 | Gscale\_GradeID | Int(10) | No | ID of grade corresponds with grading scale |

1. Table Result\_Answers:

This table stores statistic of questions. It includes: number of correct answers, number of partially correct answers, number of incorrect answers…

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | Result\_AnswerID | Int(10) | No | Auto increment |
| 2 | ResultID | Int(10) | No | ID of result |
| 3 | QuestionID | Int(10) | No | ID of question |
| 4 | TestQuestionID | Int(10) | No | ID of question of test |
| 5 | Result\_Answer\_Text | Text | No | Content of answer which is correct |
| 6 | Result\_Answer\_Points | Float | No | Point of answer |
| 7 | Result\_Answer\_Iscorrect | Tinyint(3) | No | Answer is correct |
| 8 | Result\_Answer\_Feedback | TEXT | No | Feedback of answer (if have) |
| 9 | Result\_Answer\_TimeSpent | Int(10) | No | Time to spend to answer question. |
| 10 | Result\_Answer\_TimExceeded | Tinyint(3) | No | Time which is exceeded |

1. Table Theories:

This table stores information of theories. It includes: theory name, theory description, objective, reference questions…

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | Theoryid | INT(10) | No | Auto increment |
| 2 | Theory\_description | TEXT | No | Description of theory |
| 3 | Theory\_name | VARCHAR(255) | No | Name of theory |
| 4 | Objective | TEXT | No | Objectives which theory bring out |
| 5 | Theory\_body | TEXT | No | Content of theory |
| 6 | Theory\_file\_path | VARCHAR(255) | No | Path of theory source (video, text…) |
| 7 | Subjectid | INT(10) | No | ID of subject of theory |

1. Table Gscales:

This table stores information of grading scale.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | Gscaleid | INT(10) | No | Auto increment |
| 2 | Gscale\_name | VARCHAR(255) | No | Name of grading scale |
| 3 | Gscale\_description | VARCHAR(255) | No | Description of grading scale |

1. Table Gscales\_grades:

This table stores grades of grading scale.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | Gscaleid | INT(10) | No | ID of grading scale |
| 2 | Gscale\_gradeid | INT(10) | No | Auto increment |
| 3 | Grade\_name | VARCHAR(255) | No | Name of grade of grading scale |
| 4 | Grade\_description | VARCHAR(255) | No | Description of grade |
| 5 | Grade\_feedback | TEXT | No | Feedback of grade (if have) |
| 6 | Grade\_from | FLOAT | No | Value of grade is minimum |
| 7 | Grade\_to | FLOAT | No | Value of grade is maximum |

1. Table Groups:

This table stores information of authorities which correspond with user’s groups.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | Groupid | INT(10) | No | Auto increment |
| 2 | Group\_name | VARCHAR(255) | No | Name of group |
| 3 | Group\_description | VARCHAR(255) | No | Description of group |
| 4 | Access\_tests | TINYINT(3) | No | Authority to test |
| 5 | Access\_testmanager | TINYINT(3) | No | Authority to see test’s manager |
| 6 | Access\_gradingsystems | TINYINT(3) | No | Authority to see grading systems |
| 7 | Access\_emailtemplates | TINYINT(3) | No | Email which is sent has template form |
| 8 | Access\_reporttemplates | TINYINT(3) | No | Report has template form |
| 9 | Access\_reportsmanager | TINYINT(3) | No | Authority to see report manager |
| 10 | Access\_questionbank | TINYINT(3) | No | Authority to see and create questions |
| 11 | Access\_subjects | TINYINT(3) | No | Authority to see and create subjects. |
| 12 | Access\_groups | TINYINT(3) | No | Managing authority about the system |
| 13 | Access\_users | TINYINT(3) | No | Managing authority about users |
| 14 | Access\_visitors | TINYINT(3) | No | Managing authority about visitors |
| 15 | Access\_config | TINYINT(3) | No | Managing configuration of the system |

1. Table etemplates:

This table stores templates of email.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | Etemplateid | INT(10) | No | Auto increment |
| 2 | Etemplate\_name | VARCHAR(255) | No | Name of email template |
| 3 | Etemplate\_description | VARCHAR(255) | No | Description of email template |
| 4 | Etemplate\_from | VARCHAR(255) | No | Email to send |
| 5 | Etemplate\_subject | VARCHAR(255) | No | Subject of email |
| 6 | Etemplate\_body | TEXT | No | Content of email |

1. Table rtemplates:

This table stores templates of report.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | Rtemplateid | INT(10) | No | Auto increment |
| 2 | Rtemplate\_name | VARCHAR(255) | No | Name of report template |
| 3 | Rtemplate\_description | VARCHAR(255) | No | Description of report template |
| 4 | Rtemplate\_body | TEXT | No | Content of report template |

1. Table groups\_tests:

This table is joined with groups table, tests table by groupid and testid.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | Groupid | INT(10) | No |  |
| 2 | Testid | INT(10) | No |  |

1. Table groups\_users:

This table is joined with groups table, users table by groupid and userid.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | Groupid | INT(10) | No |  |
| 2 | Userid | INT(10) | No |  |

1. Table tests\_attempts:

This table stores number of attempts to take a test.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | Testid | INT(10) | No | ID of test |
| 2 | Userid | INT(10) | No | ID of user which takes the test |
| 3 | Test\_Attempt\_count | INT(10) | No | Number of attempts to take the test |

1. Table tests\_questions:

This table stores questions of tests.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column name | Data type | Nullable | Description |
| 1 | Test\_questionid | INT(10) | No | Position of questions in a test |
| 2 | Testid | INT(10) | No | ID of test |
| 3 | Test\_sectionid | INT(10) | No | Section of questions in a test |
| 4 | questionid |  | No | ID of question which is taken in question bank |

# 4.6. Other material (if any)