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

# 4.1. Design Overview

# 4.2. System Architectural Design

## 4.2.1 Choice of System Architecture

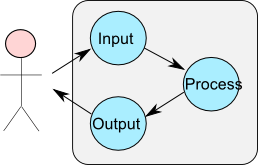
The following diagram represents the Model-View-controller pattern:



Model-View-Controller Diagram[1]

Model-View-Controller (herein referred to as MVC) 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.

[1] <http://netbeans.org/images_www/articles/69/javaee/ecommerce/design/mvc-model.png>



These three main roles are the basis for the Joomla MVC. They are described here in brief, but for a more thorough explanation, please refer to the links provided at the end of this tutorial.

### 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.

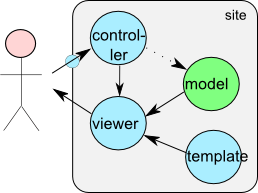
### View

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.

### MVC connection

[](http://docs.joomla.org/File:MVC_joomla.png)

The simplified picture on the right depicts the basic components being used within Joomla. Besides the Model, the View and the Controller, an Entry Point has been added that is depicted as a small circle. Attached to the viewer (view) a Template has been added. With these five components you should be able to understand this tutorial about making a basic Joomla! MVC component.

Part 1 of the tutorial only focuses on the Controller and the View (with the use of the Template); these are marked with the blue color in the picture. Part 2 adds and Part 3 extends the model functionality for the data manipulation abstraction; marked with the green color in the picture.

Keep in mind that this simplified picture only applies for the site section (i.e the front-end). An identical picture is applicable for the admin section (i.e. the back-end). The administrative section is taken care of in Parts 4 through 6 of this component development tutorial. Both the site and the admin section are maintained and configured in an XML based installation file (typically termed a manifest file).

**Why should MVC?**

* **Re-use of Model components:** The separation of model and view allows multiple views to use the same enterprise model. Consequently, an enterprise application's model components are easier to implement, test, and maintain, since all access to the model goes through these components.
* **Easier support for new types of clients:** To support a new type of client, you simply write a view and some controller logic and wire them into the existing enterprise application.
* **Increased design complexity:** This pattern introduces some extra classes due to the separation of model, view, and controller.

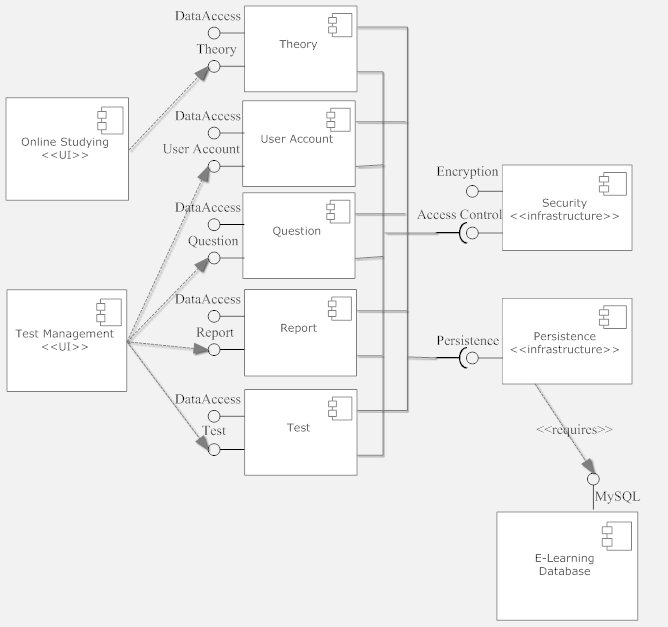
**Disadvantage**

* Require more technical skill to understand and implement
* Time consuming
* Easy debugging
* Maintaining file structure is import

## 4.2.2 Description of System Interface

“E-Learning” system

# 4.3. Component Diagram



# 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

4.4.1.1.1.2. Layout – Description of the User Interface

4.4.1.1.2.

4.4.1.1.2.1. 2 – Screen Images

4.4.1.1.2.2. 2 – Description of the User Interface

4.4.1.1.3.

4.4.1.1.3.1. 3 – Screen Images

4.4.1.1.3.2. 3 – Description of the User Interface

4.4.1.2. Theory – Class Diagram

s

4.4.1.3. Theory – Class Diagram Explanation

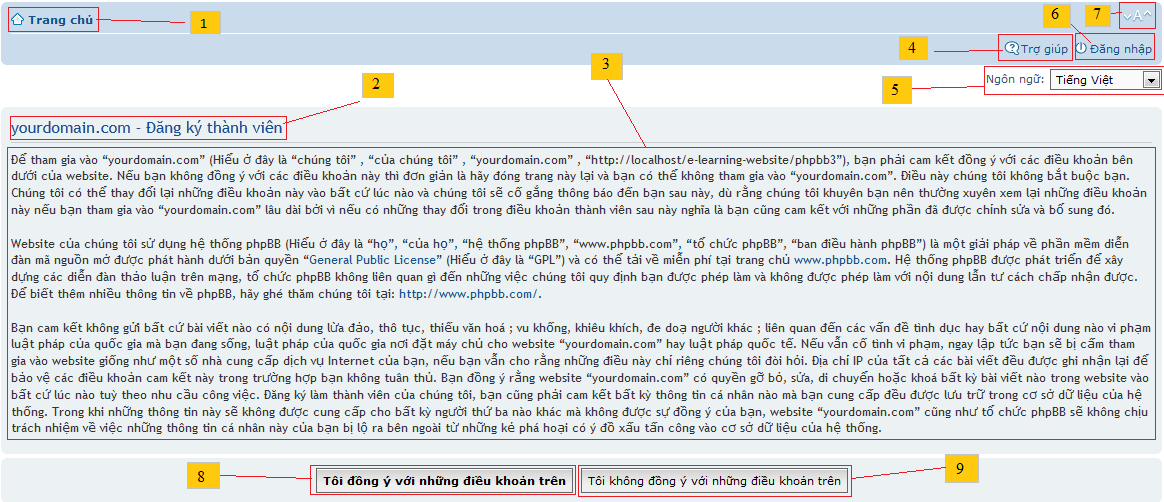
s

## 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



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



4.4.2.1.2.1. 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. Question

4.4.3.1. Question – User Interface

4.4.3.1.1.

4.4.3.1.2.

4.4.3.1.3.

4.4.3.2. Question – Class Diagram

4.4.3.3. Question – Class Diagram Explanation

## 4.4.4. Report

4.4.4.1. Report – User Interface

4.4.4.1.1.

4.4.4.1.2.

4.4.4.1.3.

4.4.4.2. Report – Class Diagram

4.4.4.3. Report – Class Diagram Explanation

## 4.4.5. Test

4.4.5.1. Test – User Interface

4.4.5.1.1. Preparing test

4.4.5.1.1.1. Preparing test - Screen images

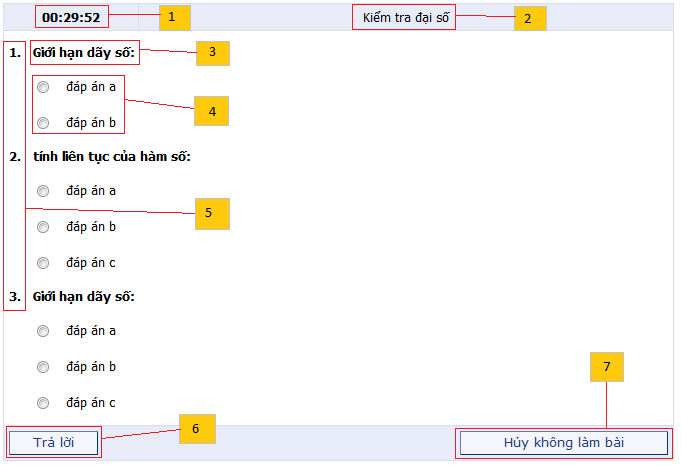


4.4.5.1.1.2. Preparing test - Description of the User Interface

|  |  |  |
| --- | --- | --- |
| 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 |

4.4.5.1.2. Test process:

4.4.5.1.2.1. Test process – Screen Images:



4.4.5.1.2.2. 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. |

4.4.5.2.

4.4.5.3.

4.4.5.2. Test – Class Diagram

4.4.5.3. Test – Class Diagram Explanation

# 4.5. Sequence Diagram

# 4.6. User Interface Design

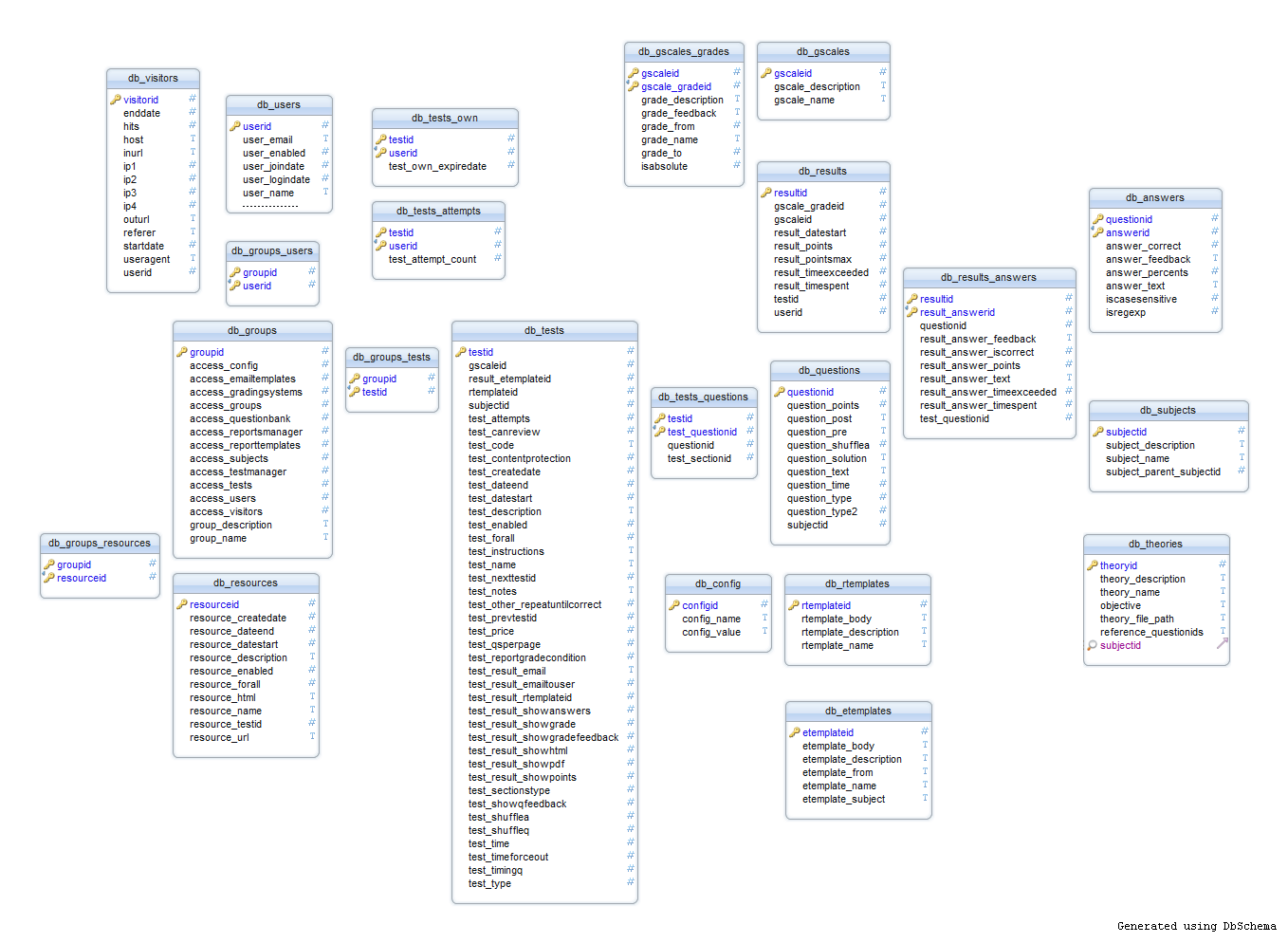
## 4.6.1 Description of the User Interface

4.12.1.1 Screen Images

4.12.1.2 Objects and Actions

# 4.7. Database Design or Data Structures

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



## 4.7.2 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 |
| 8 | …….. |  |  |  |

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 |

# 4.8. Other material (if any)

cái to nhất là "thư viện lý thuyết"

click vào cái menu "thư viện .." thì nó sổ ra tên các môn học ở phía dưới

click vào các môn học thì sẽ hiện ra các theory của môn học đó ở phần content giữa màn hình

rồi chọn vào cái theory nào thì nó sẽ in ra nội dung phía dưới luôn

hiện tại đơn giản làm thế đã