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

# 4.1. Design Overview

# 4.2. System Architectural Design

## 4.2.1 Choice of System Architecture

MVC programming is the application of this 3-way factoring

* + Objects of different classes take over the operations related to the application domain (the model)
  + The display of the application's state (the view)
  + The user interaction with the model and the view (The controller)
* Models is:
  + the domain-specific software simulation
  + Implementation of the application's central structure.
* Views deal with everything graphical:
  + Request data from their model
  + And display the data
* Controllers contain the interface between:
  + Associated models
  + Associated views
  + The input devices (e.g., keyboard, pointing device, time).

We choose MVC-I which is system architecture of the website system.

The connection between the Controller-View and the Model:

* + Can be designed in a pattern of subscribe-notify
    - the Controller-View subscribes to the Model
    - the Model notifies the Controller-View of any changes.
  + The Controller-View is an observer of the data in the Model module.

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

DB

Model

View - Controller

Model-View-Controller –I Diagram [1]

* The MVC-I is a simple version of MVC architecture
* The system is simply decomposed into 2 sub-systems:
* The Controller-View:
  + - takes care of input and output processing and their interfaces
    - takes appropriate action upon the changes on the model
* The Model:
  + - Copes with all core functionality and the data.
    - notifies the Controller-View module of any data changes so that any graphics data display will be changed accordingly
* The Controller-View module registers with the Model module.

**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 Discussion of Alternative Designs

<<>>

## 4.2.3 Description of System Interface

# 4.3. Component Diagram

# 4.4. Detailed Description of Components

## 4.4.1. CRC Cards (Class-Responsibility-Collaborators)

4.4.1.*n.* Component-*n*

4.4.1.n.1. Class Diagram

4.4.2.n.2. Class Diagram Explanation

*<Provide a brief explanation of the class diagram above. You do not need to explain “obvious” parts of your class diagram>*

4.4.3.n.3. Algorithms of important methods in each class specified in pseudo code or by Flow-Chart

# 4.5. Sequence Diagram

# 4.6. User Interface Design

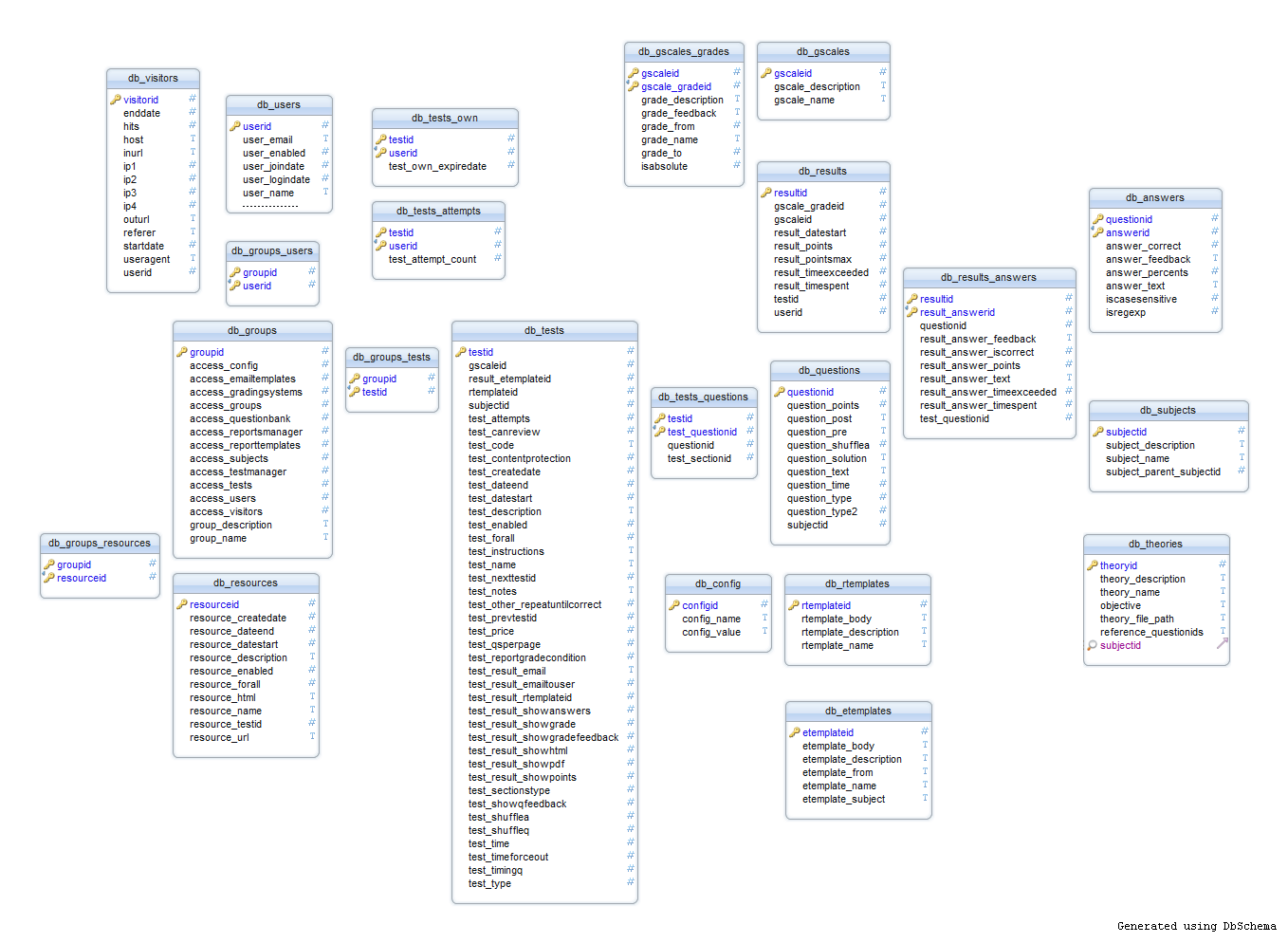
## 4.6.1 Description of the User Interface

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)