

**Online learning**

**Final Release Document**

– Hanoi, Mar 2023 –

# Table of Contents

[I. Deliverable Package 3](#_Toc98677685)

[II. Installation Guides 3](#_Toc98677686)

[III. User Manual 3](#_Toc98677687)

[1. Overview 3](#_Toc98677688)

[2. <<Workflow Name 1>> 3](#_Toc98677689)

[2. <<Workflow Name 2>> 3](#_Toc98677690)

# Overview

## Description

The online learning website is a platform that provides educational resources, courses, and programs that can be accessed through the internet. These websites allow learners to access a wide range of materials including video lectures, online textbooks, quizzes, assessments, and interactive multimedia content. Some online learning websites may offer courses for free, while others require a fee or subscription for access to their content.

The online learning website typically provides a user-friendly interface that allows learners to browse available courses, register for classes, and track their progress. Many websites also offer discussion forums, chat rooms, and other interactive features that allow learners to communicate with instructors and other students.

## Flow chart

Diagram

Description automatically generated

# Deliverable Package

| **No.** | **File** | **Notes** |
| --- | --- | --- |
| 1 | SWP-Project.sql | Database, including both table structures and table data |
| 2 | SWP391\_Group6\_Se1642\_SRS.docx | Final SRS (Requirement) Document |
| 3 | SWP391\_Group6\_Se1642\_SDS.docx | Final SDS (Design) Document |
| 4 | SWP391\_Group6\_Se1642\_Project Tracking.xlsx | Final status for the application functions, which include:  - Function name, feature, roles, function description (changes compared to the original requirements are highlighted)  - In Charge, Planned iteration & Actual iteration  - Final status & note on the update you made  - The links to the SRS & SDS are tracked |
| 5 | SWP391\_Group6\_Se1642\_IssuesReport.xlsx | Final issues tracking list of the whole project |

Other related deliverables:

* Tagged source codes: https://gitlab.com/group68841806/online-learning

# II. Installation Guides

1. Step 1: Open gitlab int this link and click download source code of project

Link: https://gitlab.com/group68841806/online-learning

Graphical user interface, text, application, email

Description automatically generated

1. Step 2: Unzip the file and save in a storage

*If you use the RAR you can click in Wizard and choose the storage you want save.*

Graphical user interface, application

Description automatically generated

1. Step 3: Open Apache Netbeans to open a project

Background pattern

Description automatically generated

1. Step 4: Select “file” at the top left of the screen. Then choose “Open project” then select the project you saved in the folder before. And click Open Project.

Graphical user interface, application, Word

Description automatically generated

1. Step 5: Right click on the library then select Add Jar/Folder. Then Open lib which have a list library and framework for the main project.

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. Step 6: Open The script SWP-Project.sql And run it on Microsoft SQL server (you must install Microsoft SQL Server to run)

Graphical user interface, table

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

1. Step 7: After add library and data for the project Right click on project choose Clean and Build. Then Click run project or you can click on green rectangle on top.

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. Step 8: After running you the website will display on local web. Now you can use the local web and test all function of website.

Graphical user interface, application, website

Description automatically generated

# III. User Manual

## Home page

1. Screen

*Graphical user interface, application

Description automatically generated*

1. Description

* This will be the main screen for user when entering the website. The HomeController will be responsible for display all items that home page contains.
* Functional Requirements:
* Entering website will show logos, headings, buttons, courses, details, links
* Courses must be displayed by using DAO class, connect to database and then display on website.

1. Class diagram

Graphical user interface, diagram

Description automatically generated

1. Class Specifications

**HomeController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *List all courser from databse into homepage* |

***DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

***User***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***Enroll***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***Course***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***Chapter***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***DAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *GetAllCourse(List<Course>)* | *Get all Cousre from database* |

1. Sequence Diagram(s)

Diagram

Description automatically generated

## 2. Register

1. Screen

Graphical user interface, application, Teams

Description automatically generatedGraphical user interface, application, Teams

Description automatically generated

1. Description

* This will be what screen should be when user register a new account. The SignUpServlet class will responsible for the function.
* Functional Requirements:
* Check if username is existed or not. If it was existed, display an error that user should choose another username to register.
* Username and password must be validated.
* When user type in password and confirm password, that two passwords must be exactly the same. Otherwise, system should display an error show that passwords don’t match each other.
* Register successfully will save username and password into database.
* Allow system to remember user, so that user don’t have to login again.
* Security:
* All passwords must be encrypted and stored securely.

1. Class diagram

Diagram

Description automatically generated

1. Class Specifications

**SignUpServlet**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *Get email, password and fullname from resgister form, check it and add to database* |

***DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

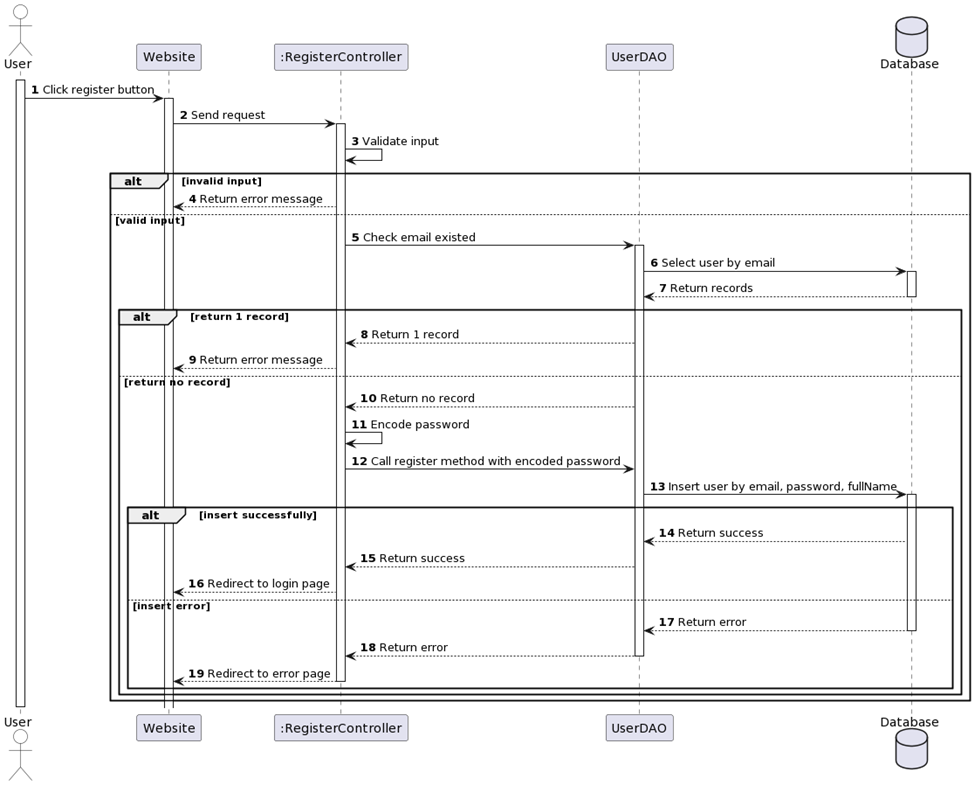
***User***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***UserDAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *CheckUser(String s)* | *Check if user name is existed or not* |
| *02* | *CheckUserGoogle*  *(String username )* | *Check google email is existed or not in the system* |

1. Sequence Diagram(s)



## 3. Login

1. Screen

Graphical user interface, application, Teams

Description automatically generated

1. Description

* The purpose of this document is to specify the requirements for a Login Controller, which will be a module in a larger software system. The Login Controller will be responsible for handling user authentication and authorization, allowing users to access secure areas of the software system based on their credentials.
* Functional Requirements:
* The Login Controller must provide the following functionality:
* Accept a user's login credentials (username and password).
* Verify the user's credentials against the system's user database.
* Grant access to the system's secure areas if the user's credentials are valid.
* Display an error message if the user's credentials are invalid.
* Allow users to reset their passwords if they forget them.
* Logout Functionality:
* Allow users to log out of the system.
* End the user's session and revoke their access to the system's secure areas.
* Non-Functional Requirements.
* Security:
* All passwords must be encrypted and stored securely.
* Password reset functionality must follow secure password reset guidelines.

1. Class diagram

Diagram

Description automatically generated

1. Class Specifications

***Login Controller***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *Get email, password and fullname from resgister form, check it and add to database* |

***DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

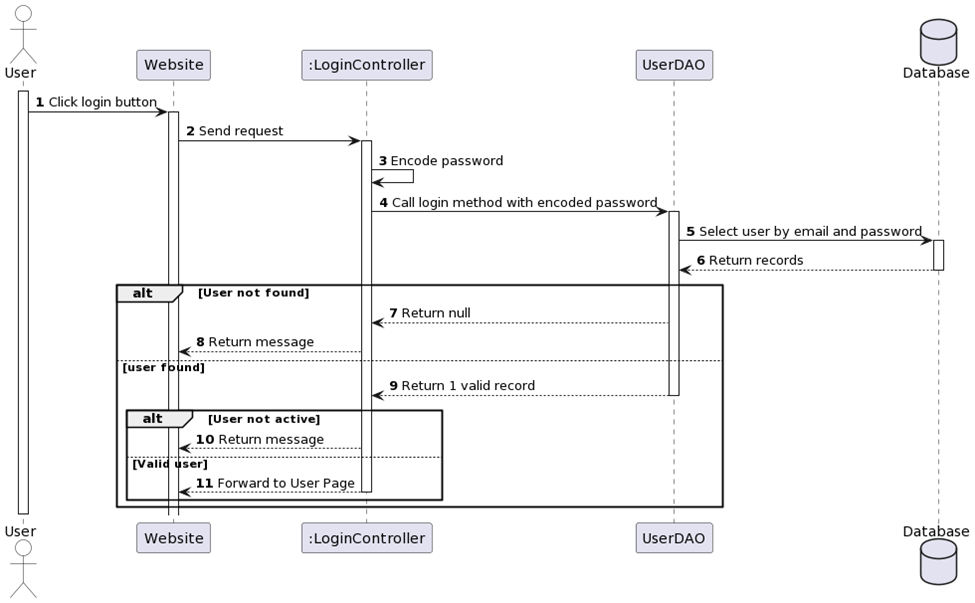
***User***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***DAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Check(String username, String password)* | *Check account in database to login* |

1. Sequence Diagram(s)



## 4. Change password

1. Screen
2. Description

* This is the function that can help user to change password is necessary. ChangePassController is the servlet responsible for the function. Connect to database and update user password through username using SQL query.
* Functional Requirements:
* Verify old password is correct.
* Display an error message if the old password is not correct.
* Validate new password and should be different from old password.
* Password changed in database if successful.
* Security:
* All passwords must be encrypted and stored securely.

1. Class diagram

Diagram

Description automatically generated

1. Class Specifications

**ChangePassword**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *Check user is existed or not, if not then you can change password* |

***DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

***User***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***DAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Change(User a)* | *Check user in database to change password* |

1. Sequence Diagram(s)

Diagram, engineering drawing

Description automatically generated

## 5. Course Detail

1. Screen

Graphical user interface, website

Description automatically generated

1. Description

* When user click on a course for more details, they will see a screen look like above. Course detail will contain that course’s description, type, all chapters, number of lessons, … Therefore, DAO class and CourseDetail servlet will be main factors which responsible for this screen.
* Functional Requirements:
* View all chapters
* View number of lessons using database.
* Through Enroll, can see how many students entered this course and how this course rated.

1. Class diagram

Graphical user interface, application

Description automatically generated

1. Class Specifications

**CourseDetail**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Get Course by id from database and show in page* |

***DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

***Course***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***Chapter***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***Lesson***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***DAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getCourseById(int id)* | *Get course from database by id and display on website* |

1. Sequence Diagram(s)**Diagram

   Description automatically generated**

## 6. Discussion

1. Screen

Graphical user interface, application, Teams

Description automatically generated

1. Description

* Discussion is the section where users can post their comments about course or anything they might need help or to discuss with others. DiscussionDAO and TopicServlet will in charge of this. Authorization will help we know whose topic or comments belongs to, so that they can reach to each other easily.
* Functional Requirements:
* Display all topics and all comments.
* Allow user to create a new Topic to discuss

1. Class diagram

Diagram

Description automatically generated

1. Class Specifications

***TopicServlet***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *Get discussion and reply of user* |
| *02* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *Reply discussion with all discussion detail* |

***DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

***User***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***Discussion***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***DiscussionDAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getDiscussionbyID(int id)* | *Get Discussionof user by ID* |
| *02* | *getReplyDiscussion()* | *Get Reply Discussion of user* |
| *03* | *replyDiscussion(int id, Discussion d) {* | *Reply discussion with all discussion detail* |

1. Sequence Diagram(s)Diagram

   Description automatically generated

## 7. Chapter Detail

1. Screen

Graphical user interface, application

Description automatically generated

1. Description
2. Class diagram
3. Class Specifications
4. Sequence Diagram(s)

## 8. Lesson Detail

1. Screen

Graphical user interface, text, application, email

Description automatically generated

1. Description
2. Class diagram

***A picture containing graphical user interface

Description automatically generated***

1. Class Specifications

***LessonServlet***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *Get all lesson from course by FindAllLessonByChapterId* |
| *02* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *get a percentage at the end of a lesson* |

***DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

***User***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***Lesson***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***Percentage***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***LessonDAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *FindAllLessonByChapterId(int chapterId)* | *Get each lesson in each Chapter containing that lesson* |
| *02* | *getPercentageByUserID(int user\_id)* | *Get the percentage each time the user finishes each lesson* |

1. Sequence Diagram(s)Diagram

   Description automatically generated

## 9. Practice

1. Screen

Graphical user interface, website

Description automatically generated

1. Description
2. Class diagram

Diagram

Description automatically generated

1. Class Specifications

**QuizzServlet**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getChoice(String parameter, Question question)* | *Get Question, choices and answer from database* |

**DBContext**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

***Quizz***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***QuizzDAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getListQuiz(int id)* | *Get List of quizz from database* |

1. Sequence Diagram(s)

## 10. Mark report

1. Screen

Graphical user interface, text, application, email

Description automatically generated

1. Description
2. Class diagram
3. Class Specifications

***ReviewServlet***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *Get a list of questions and answers for review* |

**DBContext**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

***Question***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***QuizDAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getRecordbyID(int id)* | *Get record of quizz to review* |

1. Sequence Diagram(s)

## 11. View correct answer

1. Screen

Graphical user interface, text, application, email

Description automatically generated

1. Description
2. Class diagram

Graphical user interface

Description automatically generated with low confidence

1. Class Specifications

***RecordServlet***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *Get the quizz's record* |

***DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

***Question***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***QuizzDAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getRecord(int user, int course)* | *Get record of Quizz* |

1. Sequence Diagram(s)Diagram

   Description automatically generated

## 12. Comment course

1. Screen
2. Description
3. Class diagram

Diagram

Description automatically generated with medium confidence

1. Class Specifications

***CommentCourseServlet***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *processing comments for the course* |
| *02* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *processing discussion for the course* |

***DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

***CommentCourse***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***Discussion***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***CommentDAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getCommentCourse(int id)* | *Get Comment of user in CommentCourse* |
| *02* | *rateCommentCourse(int id)* | *handle the like for comments, the number of likes increases every time the like button is pressed* |

***DiscussionDAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *addDiscussion(Discussion d)* | *add details of discussion* |
| *02* | *replyDiscussion(int id, Discussion d)* | *reply to the user's discussion* |

1. Sequence Diagram(s)Diagram

   Description automatically generated

## 13. Course Percentage

1. Screen
2. Description
3. Class diagram
4. Class Specifications
5. Sequence Diagram(s)

## 14. Purchase course

1. Screen

Graphical user interface, application, Teams

Description automatically generated

1. Description
2. Class diagram
3. Class Specifications
4. Sequence Diagram(s)

## 15. Search

1. Screen

Graphical user interface, application, Teams

Description automatically generated

1. Description
2. Class diagram

Graphical user interface, application

Description automatically generated

1. Class Specifications

**SearchCourseServlet**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *Get name and description of course* |

**SearchLessonServlet**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *Get name and description of Lesosn* |

**SearchChapterServlet**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *Get name and description of Chapter* |

**DBContext**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

***DAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *SearchChapterByName* | *Fine all Chapter where chapter name like what you want to search* |
| *02* | *SearchCourse* | *Fine all Course where chapter name like what you want to search* |
| *03* | *SearchLesson* | *Fine all lesson where chapter name like what you want to search* |

***Course***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***Lesson***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***Chapter***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

1. Sequence Diagram(s)Diagram, engineering drawing

   Description automatically generated

## 16. Profile

1. Screen

Graphical user interface, application

Description automatically generated

1. Description
2. Class diagram

Diagram, engineering drawing

Description automatically generated

1. Class Specifications

**ProfileServlet**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *Get profile of course* |
|  |  |  |

***DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

***User***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***Course***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***Chapter***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***Enroll***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***DAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Change(User a)* | *Check user in database to change password* |

1. Sequence Diagram(s)

Diagram

Description automatically generated

## 17. Change Profile

1. Screen
2. Description
3. Class diagram
4. Class Specifications
5. Sequence Diagram(s)

## 18. Admin Manage

1. Screen
2. Description
3. Class diagram

***A picture containing graphical user interface

Description automatically generated***

1. Class Specifications

***Manage***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *Get name, gender, email,address,phone,tong in database and show on manage page* |

***DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

***ThongTin***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***DAOAdmin***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *numberOfStudent()* | *Check number of student* |
| *02* | *numberOfCourse()* | *Check number of course* |
| *03* | *numberOfEnroll()* | *Check number of enroll* |
| *04* | *getInfoCustomer()* | *Check all information of customer* |

1. Sequence Diagram(s)Diagram

   Description automatically generated

## 19. Static chart

1. Screen

Chart

Description automatically generated

1. Description
2. Class diagram
3. Class Specifications
4. Sequence Diagram(s)

## 20. User Manage

1. Screen

Graphical user interface, application

Description automatically generated

1. Description
2. Class diagram

Diagram

Description automatically generated with medium confidence

1. Class Specifications

***DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |

***UserDAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *checkUser* | *Check all users exist in database already.* |
| *02* | *CheckUserById* | *Check only user have their id* |
| *03* | *addUser* | *Add user to the database* |
| *05* | *CheckUserGoogle* | *Check user login with google (check email has exist)* |
| *06* | *GetAllUser()* | *Take all user from database to the system.* |

***UserManage***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *Get list user from database to display in the screen.* |

***User***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
|  |  |  |

1. Sequence Diagram(s)Diagram

   Description automatically generated

## 21. Manage course

1. Screen

Graphical user interface, text, application, email

Description automatically generated

1. Description
2. Class diagram

Graphical user interface

Description automatically generated with medium confidence

1. Class Specifications

**CourseManage**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *Get all course in database and show on courseManager page* |

***DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

***Course***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value of each attributes* |

***DAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getAllCourse()* | *Get all course from database and display on page* |

1. Sequence Diagram(s)

Diagram

Description automatically generated

## 2. <<Workflow Name 2>>

1. Screen
2. Description
3. Class diagram
4. Class Specifications
5. Sequence Diagram(s)