CATEGORY

TABLE OF CONTENT PAGE

[Acknowledgements 3](#_Toc520706465)

[REVIEW 1 4](#_Toc520706466)

[1. Problem Definition: 5](#_Toc520706467)

[1.1 Introduction: 5](#_Toc520706468)

[1.2 Existing Scenario: 5](#_Toc520706469)

[2. User Requirements Specifications: 5](#_Toc520706470)

[3. Function Requirement Specification: 5](#_Toc520706471)

[1.1 Guest: 5](#_Toc520706472)

[1.2 Learner: 6](#_Toc520706473)

[1.3 Administrator: 6](#_Toc520706474)

[4. System Requirement: 6](#_Toc520706475)

[1.1 Hardware Requirement: 6](#_Toc520706476)

[1.2 Software Requirement: 7](#_Toc520706477)

[Task Sheet 1 8](#_Toc520706478)

[REVIEW 2 9](#_Toc520706479)

[1. Software Architecture 10](#_Toc520706480)

[2. User Case and Sequence Diagram 11](#_Toc520706481)

[1.1 Administrator, Learner and Teacher Case 11](#_Toc520706482)

[1.2 Learner and Teacher Case 16](#_Toc520706483)

[1.3 Learner Case 18](#_Toc520706484)

[1.4 Teacher and Administrator Case 23](#_Toc520706485)

[3. Entities Definition 26](#_Toc520706486)

[1.1 Entities in table 26](#_Toc520706487)

[1.2 Diagram 31](#_Toc520706488)

[Task Sheet 2 32](#_Toc520706489)

# Acknowledgements

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

We are grateful to our project guide Mr. Ngo Phuoc Nguyen for the guidance, inspiration and constructive suggestions that helpful us in the preparation of this project.

We also thank our colleagues who have helped in successful completion of the project.

Group 01: F2-13-1610-S1

# REVIEW 1

1. Problem Definition:

### Introduction:

TodayIGotIt(TiGi) is an online website have a lot of IT course to help the user learning more and more effective with a lot of ralative videos, documents and a code editor online with no implementation in user’s device.

### Existing Scenario:

Nowaday, technology is developing so fast, the IT learner want to find effective resources to improve their knowledges. An online course website is one of these resource they want.

1. User Requirements Specifications:
2. Guest (Normal User):

Guest will able searching for all course in website.

1. Learner:

Learner can review course, register course and send feedback to admin.

1. Function Requirement Specification:

### Guest:

* Guest can register as learner.
* Guest can login as learner.
* Guest can review all course in the website.

### Learner:

* Learner can explore all course in website, search course by course name, instructor name…
* Learner can register some course to learn.
* Learner can add some course to wishlist.
* Learner can following the learning process in dashboard.
* Learner can send the feedback to instructor.
* Learner can send text messages to instructor.

### Administrator:

* Administrator manage all course as:
* Upload course including: videos, documents.
* Block course.
* Remove course.
* Administrator can view and reply feedback from learner.
* Administartor can block and unblock the learner.
* Administrator can view the report of all course in website.

1. System Requirement:

### Hardware Requirement:

* AMD Opteron, AMD Athlon 64, Intel Xeon with Intel EM64T support, Intel Pentium IV with EM64T support
* Processor Speed minimum 1.4 hz
* 1Gb of RAM or higher. ( 4 GB **Recommended** )
* Hard disk requirement : minimum 50GB
* Windows 7 ultimate, or window server 2008 R2 or higher.

### Software Requirement:

* SQL Server 2012 x64 bit
* Netbean 8.0 or higher
* JavaEE 7 SDK
* Java Development Kit (JDK) 1.7u89 or higher ( Recommended jdk 1.8)
* Glassfish Server 4.0
* Type 4-2.0 JDBC driver for SQL Server 2012 RC4
* Internet Explorer 10 or higher

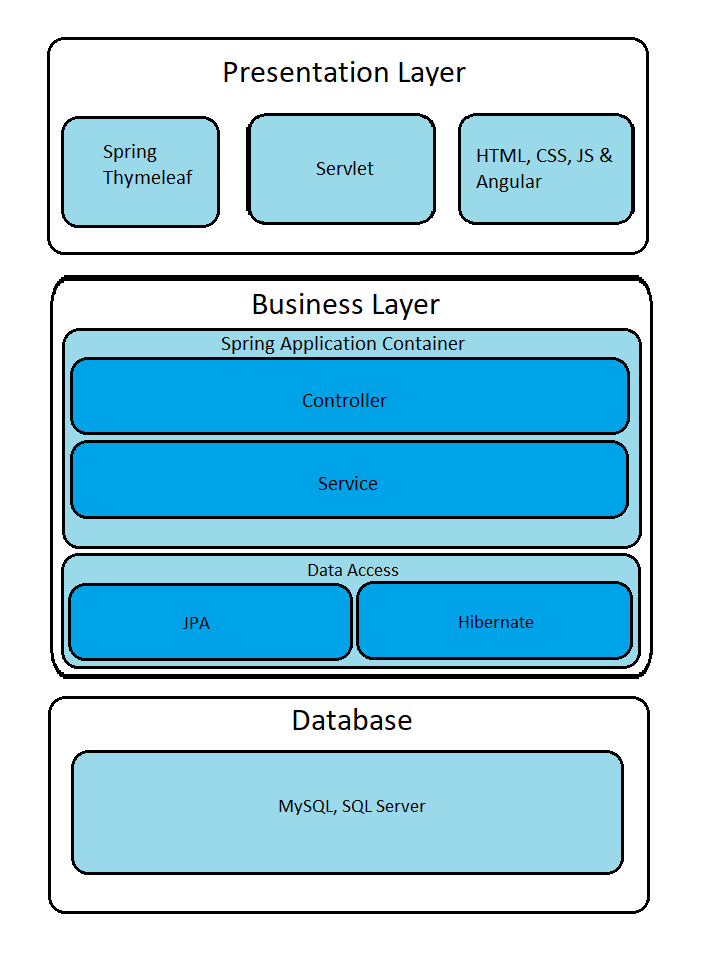
## 

## Task Sheet 1

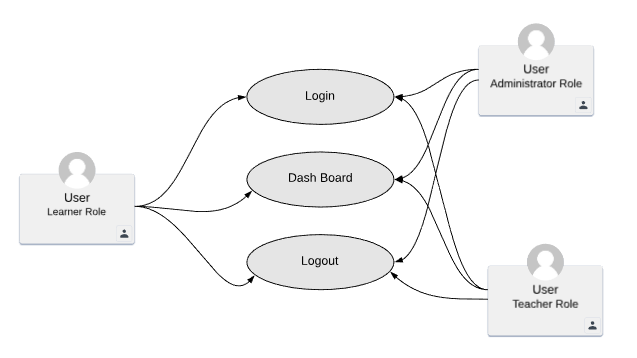
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Members Group** | | **Date Prepareation Of Activity Plan** | | | |
| **No** | **Name** | **Description Task** | **Start Date** | **End Date** | **Status** |
| 1 | Le Duc Anh |  | 04/07 | 04/07 | Finished |
| 2 | Luu Duc Trung |  | 04/07 | 04/07 | Finished |
|  | | | | | |
| **Teacher** | | | **Team Leader** | | |
|  | | |  | | |
| **Ngo Phuoc Nguyen** | | | **Le Duc Anh** | | |

# REVIEW 2

1. Software Architecture



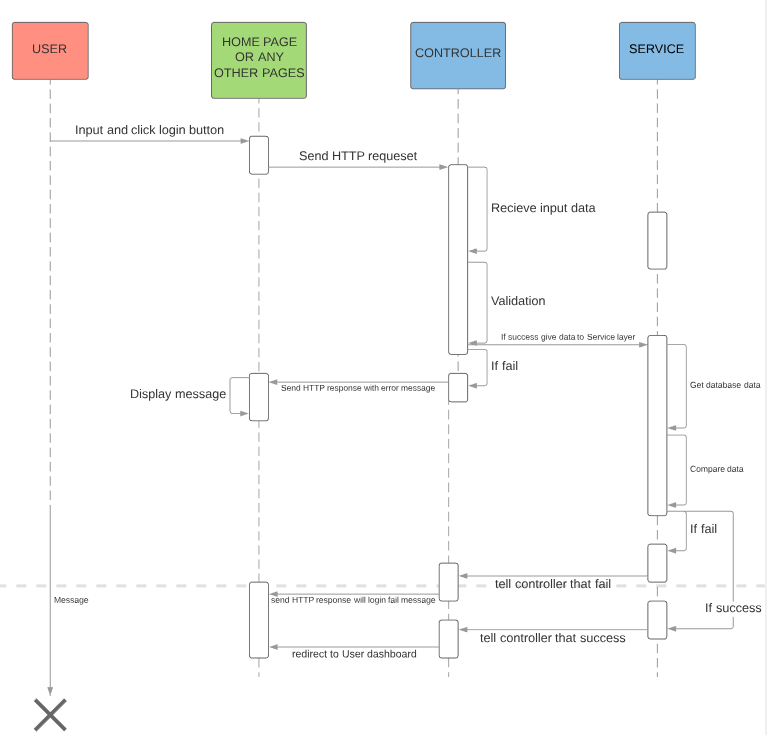
1. User Case and Sequence Diagram



### Administrator, Learner and Teacher Case

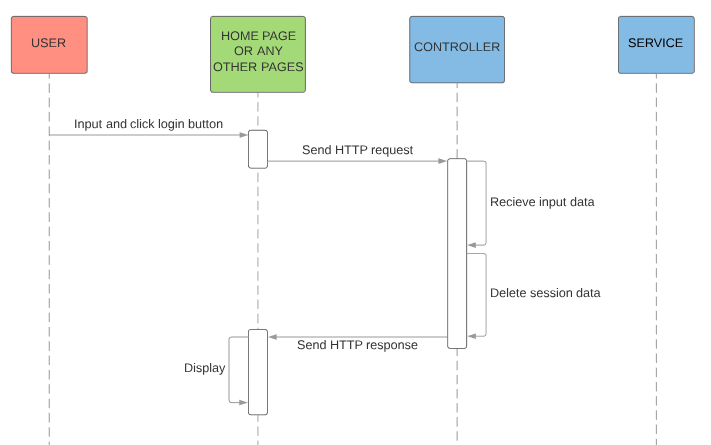
* Login:

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Login | |
| **Actors** | User | |
| **Description** | Use login as Learner, Teacher or Administrator | |
| **Requirements** | Actor provides username unencrypted password | |
| **Pre-conditions** |  | |
| **Post-conditions** | Success: User logged in to website.  Fail: Refill information | |
| **Basic flow** | Client side:  1. Stand in what ever pages, press the Signin button on navigation bar  2. type username and password in a popup logjn form  3. press Login to submit the form | Server side:  4.The Controller layer will receive the input data from HTTP request  5. and send the data to the Service layer  6. The Service layer get data from database and decrypted the password then compare these both information |
| **Exception** | Client side:  1. Leave one of username and password field blank  2. Input the correct username or password | Server side:  1.1 The Controller layer receive the input data and call the Validation Service to check the input. Then send back the error message to Client side  1.2 The Service layer after compare the input data and data in database then send the error message to Client side |



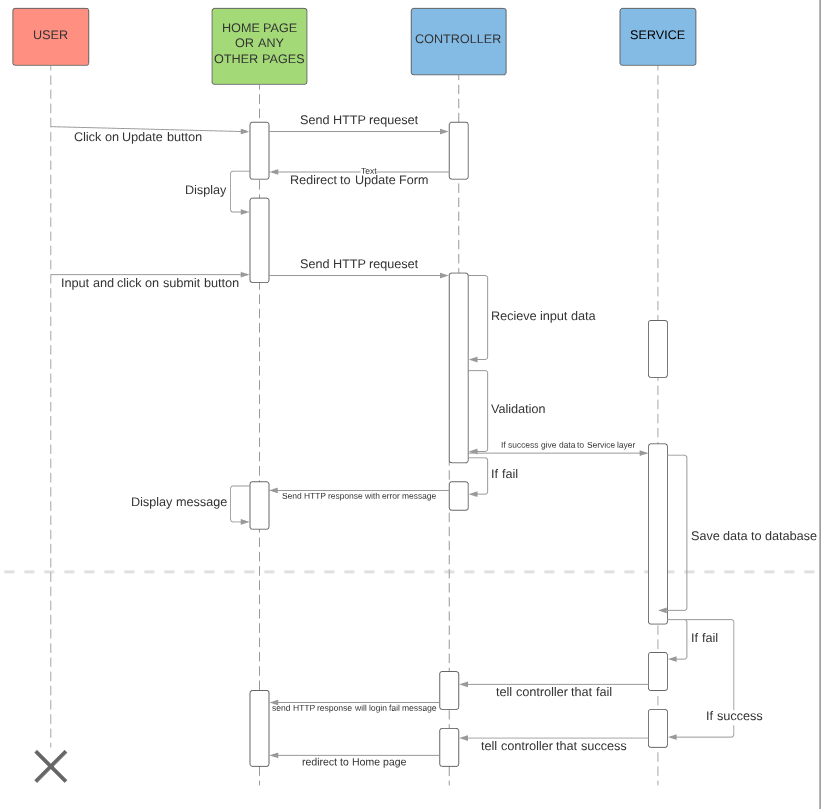
* Logout:

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Logout | |
| **Actors** | User | |
| **Description** | Logout in current session | |
| **Requirements** | none | |
| **Pre-conditions** | User login already | |
| **Post-conditions** | Success: state in current page but session data will be disconnected | |
| **Basic flow** | Client side:  1. Click on logout button | Server side:  4.The Controller layer will HTTP request  5. Delete the current session data  6. Navigate to home page |
| **Exception** |  |  |



* Update Profile:

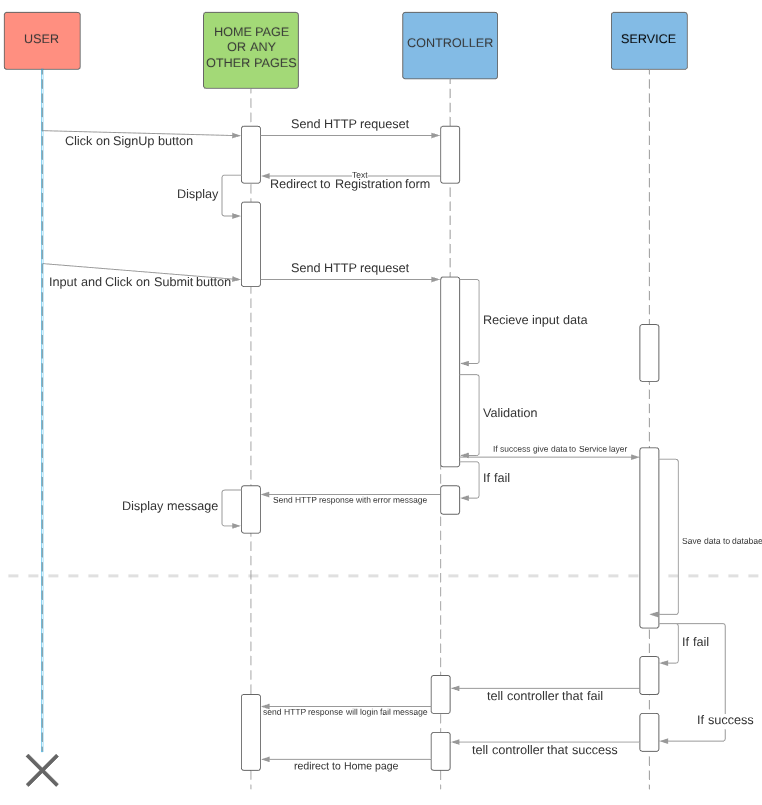
|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update Profile | |
| **Actors** | User | |
| **Description** | Learner, Teacher, Administrator change account information | |
| **Requirements** | Have account already | |
| **Pre-conditions** | Already login | |
| **Post-conditions** | Success: new account’s information will be update and save in database  Fail: Refill information | |
| **Basic flow** | Client side:  1. Click on Update button  2. The update form appear  3. Input the form  4. Click on Submit button | Server side:  5. The Controller layer will receive the input data from HTTP request  6. And send the data to the Service layer  7. The Service layer find the existed account  8. Then Encrypt the password and change existed account data. Finally save to database |
| **Exception** | Client side:  1. Input the in correct data | Server side:  1.1 The Controller layer receive the input data and call the Validation Service to check the input. Then send back the error message to Client side |



### Learner and Teacher Case

* Registration:

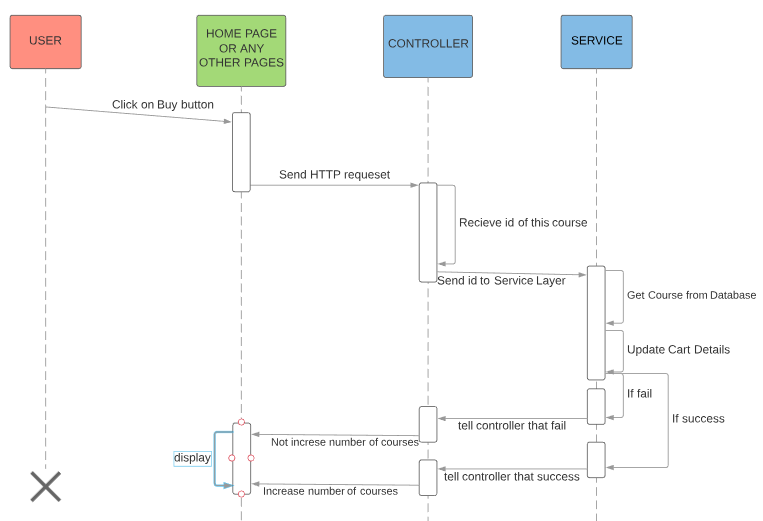
|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Registration | |
| **Actors** | Learner or Teacher | |
| **Description** | Create a new account and depend on user’s choice to use this account as Learner of Teacher | |
| **Requirements** | none | |
| **Pre-conditions** | none | |
| **Post-conditions** | Success: new account will be created and save in database  Fail: Refill information | |
| **Basic flow** | Client side:  1. Click on SignUp button  2. The registration form appear  3. Input the form  4. Click on Submit button | Server side:  5. The Controller layer will receive the input data from HTTP request  6. And send the data to the Service layer  7. The Service layer encrypt the password then save all data to database |
| **Exception** | Client side:  1. Input the in correct data | Server side:  1.1 The Controller layer receive the input data and call the Validation Service to check the input. Then send back the error message to Client side |



### Learner Case

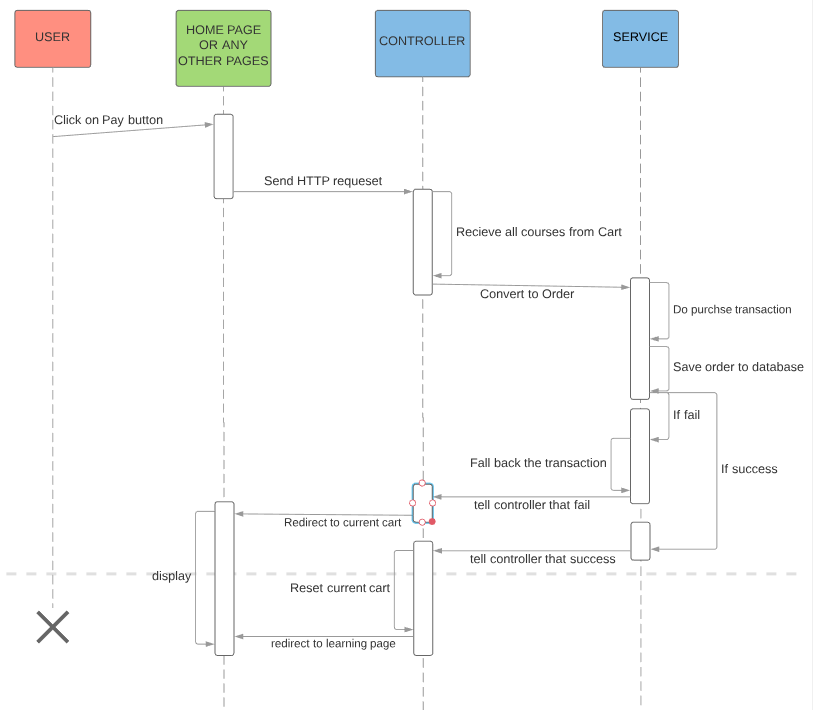
* Add to Cart:

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Add to Cart | |
| **Actors** | Learner | |
| **Description** | Add new course to current Cart | |
| **Requirements** | Already have account | |
| **Pre-conditions** | Already login | |
| **Post-conditions** | Success: new course will be added to Cart  Fail: no course was added | |
| **Basic flow** | Client side:  1. Click on Buy button of specific course  10. show number of courses | Server side:  5. The Controller layer will receive the id of this course from HTTP request  6. And send the id to the Service layer  7. The Service layer find the existed course from database  8. Then update to the Cart Details and save to database  9. The Controller update number of courses in the Cart to session data |
| **Exception** | Client side:  1. No course was added | Server side: |



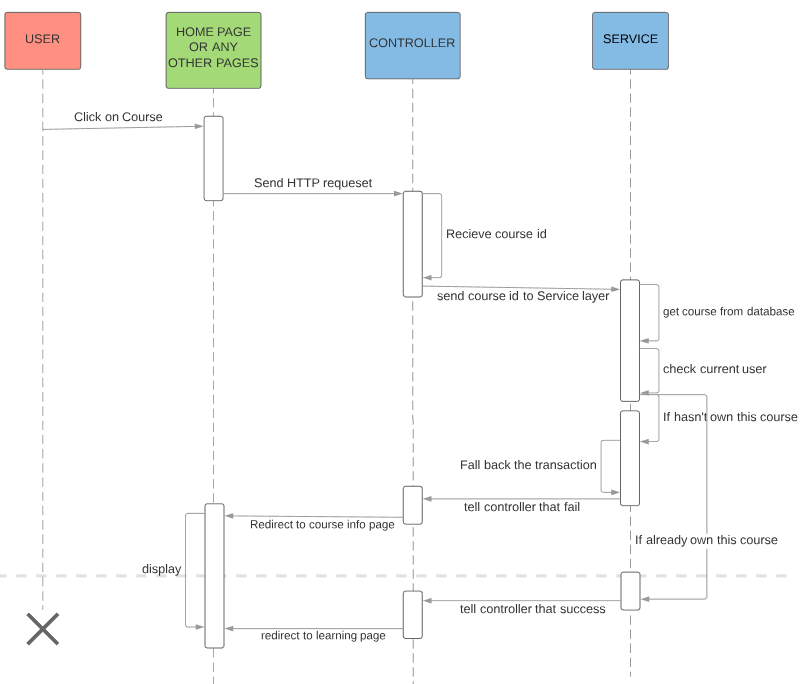
* Purchase Order:

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Purchase Course | |
| **Actors** | Learner | |
| **Description** | Earn new course from Teacher. This course will be bound to Learner account and they can start to learn imediately | |
| **Requirements** | Already have account | |
| **Pre-conditions** | Already login and Already have Course in Cart | |
| **Post-conditions** | Success: own new course to account  Fail: purchase transaction will be fallback | |
| **Basic flow** | Client side:  1. Click on Pay button  2. The update form appear  3. Input the form  4. Click on Submit button | Server side:  5. The Controller layer will receive the input data from HTTP request  6. And send the data to the Service layer  7. The Service layer find the existed account  8. Then Encrypt the password and change existed account data. Finally save to database |
| **Exception** | Client side:  1. problem from purchase transaction | Server side:  1.1 The Service layer will fallback the transaction  1.2 The Controller will response to redirect to current User’s Cart |



* Learning:

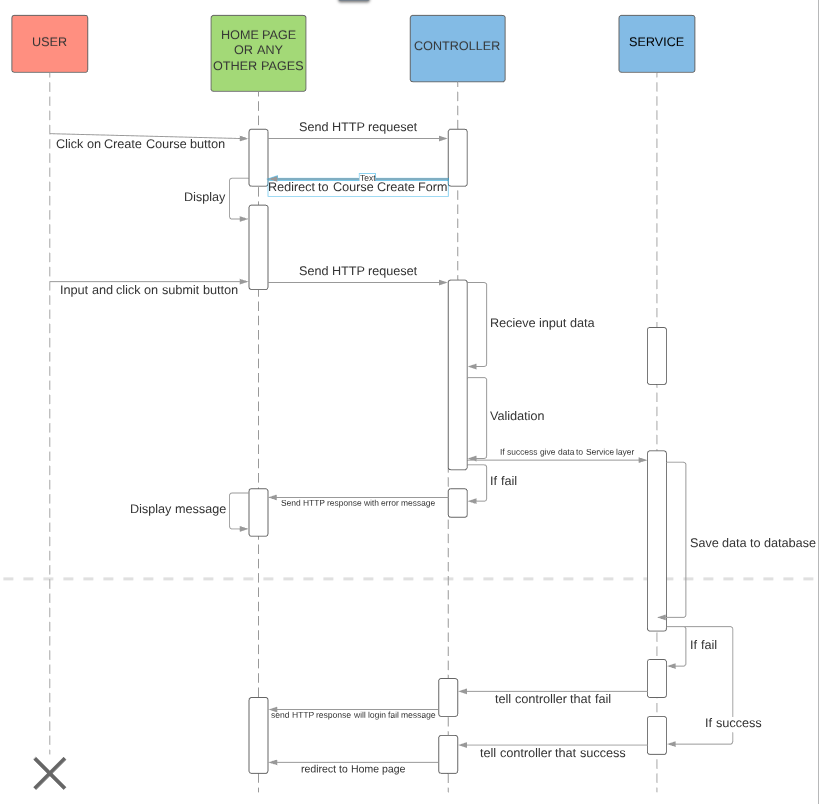
|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Learning | |
| **Actors** | Learner | |
| **Description** | Start learning and track learning process | |
| **Requirements** | Already have account | |
| **Pre-conditions** | Already login and Already own Course | |
| **Post-conditions** | Success: own new course to account  Fail: | |
| **Basic flow** | Client side:  1. Click on specific Course  9. play course | Server side:  5. The Controller layer will receive the course id from HTTP request  6. And send the data to the Service layer  7. The Service layer find the existed course and make sure this course is own by current Learner  8. Then the Controller will redirect to learning page |
| **Exception** |  |  |



### Teacher and Administrator Case

* Create Course:

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Create Course | |
| **Actors** | Teacher or Administrator | |
| **Description** | Create a new course and pubish for Learners | |
| **Requirements** | Already has account and this account is Teacher role | |
| **Pre-conditions** | Already login | |
| **Post-conditions** | Success: new course will be created and save in database  Fail: Refill information | |
| **Basic flow** | Client side:  1. Click on Create Course button  2. The course form appear  3. Input the form and post video  4. Click on Submit button | Server side:  5. The Controller layer will receive the input data from HTTP request  6. And send the data to the Service layer  7. The Service layer save all data to database |
| **Exception** | Client side:  1. Input the in correct data | Server side:   * 1. The Controller layer receive the input data and call the Validation Service to check the input. Then send back the error message to Client side |



* Update Course:

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update Course | |
| **Actors** | Teacher or Administrator | |
| **Description** | Teacher, Administrator change account information | |
| **Requirements** | Have account already | |
| **Pre-conditions** | Already login | |
| **Post-conditions** | Success: new account’s information will be update and save in database  Fail: Refill information | |
| **Basic flow** | Client side:  1. Click on Update button  2. The update form appear  3. Input the form  4. Click on Submit button | Server side:  5. The Controller layer will receive the input data from HTTP request  6. And send the data to the Service layer  7. The Service layer find the existed account  8. Then Encrypt the password and change existed account data. Finally save to database |
| **Exception** | Client side:  1. Input the in correct data | Server side:   * 1. The Controller layer receive the input data and call the Validation Service to check the input. Then send back the error message to Client side |

1. Entities Definition

### Entities in table

* User:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Key | Field Name | Data Type | Allow Null | Default Value | Table-Field-Ref | Description |
| PK | id | INT(11) | No |  |  | Primary key of Account table |
|  | username | VARCHAR(255) | No |  |  | Username of account |
|  | encryptedPassword | VARCHAR(255) | No |  |  | Encryoted Password of account |
|  | version | VARCHAR(255) | No |  |  | Version of account |
|  | dateaCreated | DATETIME | Yes |  |  | Date create of account |
|  | lastUpdated | DATETIME | Yes |  |  | Date last update |
| FK | customerId | Int | No |  |  | Foreign key of Customer which content user details information |

* Role:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Key | Field Name | Data Type | Allow Null | Default Value | Table-Field-Ref | Description |
| PK | id | INT(11) | No |  |  | Primary key of Role table |
|  | version | VARCHAR(255) | No |  |  | Version of Role |
|  | dateaCreated | DATETIME | Yes |  |  | Date create of Role |
|  | lastUpdated | DATETIME | Yes |  |  | Date last update of Role |
|  | type | INT(11) | No |  |  | Role type |

* Customer:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Key | Field Name | Data Type | Allow Null | Default Value | Table-Field-Ref | Description |
| PK | id | INT(11) | No |  |  | Primary key of Customer table |
|  | version | VARCHAR(255) | No |  |  | Version of Customer |
|  | dateaCreated | DATETIME | Yes |  |  | Date create of Customer |
|  | lastUpdated | DATETIME | Yes |  |  | Date last update of Customer |
|  | billingAddress\_addressLine1 |  | No |  |  | Billing Address line1 |
|  | billingAddress\_addressLine2 |  | Yes |  |  | Billing Address line2 |
|  | billingAddress\_city |  | No |  |  | Billing City |
|  | billingAddress\_state |  | No |  |  | Billing State |
|  | billingAddress\_zipCode |  | No |  |  | Billing Zipcode |
|  | shippingAddress\_addressLine1 |  | No |  |  | Shipping Address line1 |
|  | shippingAddress\_addressLine2 |  | Yes |  |  | Shipping Address line2 |
|  | shippingAddress\_city |  | No |  |  | Shipping City |
|  | shippingAddress\_state |  | No |  |  | Shipping State |
|  | shippingAddress\_zipCode |  | No |  |  | Shipping Zipcode |
|  | email |  | No |  |  | Emai |
|  | firstName |  | No |  |  | First name |
|  | lastName |  | No |  |  | Last name |
|  | phoneNumber |  | No |  |  | Phone number |
| FK | userId | INT(11) | No |  |  | Foreign key of User which content acccount information |

* Cart:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Key | Field Name | Data Type | Allow Null | Default Value | Table-Field-Ref | Description |
| PK | id | INT(11) | No |  |  | Primary key of Cart table |
|  | version | VARCHAR(255) | No |  |  | Version of Cart |
|  | dateaCreated | DATETIME | Yes |  |  | Date create of Cart |
|  | lastUpdated | DATETIME | Yes |  |  | Date last update of Cart |
| FK | userId | INT(11) | No |  |  | Foreign key of User which content acccount information |

* Cart Details:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Key | Field Name | Data Type | Allow Null | Default Value | Table-Field-Ref | Description |
| PK | id | INT(11) | No |  |  | Primary key of Role table |
|  | version | VARCHAR(255) | No |  |  | Version of Role |
|  | dateaCreated | DATETIME | Yes |  |  | Date create of Role |
|  | lastUpdated | DATETIME | Yes |  |  | Date last update of Role |
|  | quantity | INT(11) | No |  |  | Courses quantity |
| FK | cartId | INT(11) | No |  |  | Foreign key of Cart which content cart information |
| FK | courseId | INT(11) | No |  |  | Foreign key of Course which content course information |

* Order:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Key | Field Name | Data Type | Allow Null | Default Value | Table-Field-Ref | Description |
| PK | id | INT(11) | No |  |  | Primary key of Order table |
|  | version | VARCHAR(255) | No |  |  | Version of Order |
|  | dateaCreated | DATETIME | Yes |  |  | Date create of Order |
|  | lastUpdated | DATETIME | Yes |  |  | Date last update of Order |
|  | shipToAddress\_addressLine1 |  | No |  |  | Shipping Address line1 |
|  | shipToAddress\_addressLine2 |  | Yes |  |  | Shipping Address line2 |
|  | shipToAddress\_city |  | No |  |  | Shipping City |
|  | shipToAddress\_state |  | No |  |  | Shipping State |
|  | shipToAddress\_zipCode |  | No |  |  | Shipping Zipcode |
|  | dateShipped |  | No |  |  | Date shipped |
|  | orderStatus |  | No |  |  | Order Status |
|  | phoneNumber |  | No |  |  | Phone number |
| FK | userId | INT(11) | No |  |  | Foreign key of User which content acccount information |

* Order Details:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Key | Field Name | Data Type | Allow Null | Default Value | Table-Field-Ref | Description |
| PK | id | INT(11) | No |  |  | Primary key of Order Details table |
|  | version | VARCHAR(255) | No |  |  | Version of Order Details |
|  | dateaCreated | DATETIME | Yes |  |  | Date create of Order Details |
|  | lastUpdated | DATETIME | Yes |  |  | Date last update of Order Details |
|  | quantity | INT(11) | No |  |  | Courses quantity |
| FK | orderId | INT(11) | No |  |  | Foreign key of Order which content order information |
| FK | courseId | INT(11) | No |  |  | Foreign key of Course which content course information |

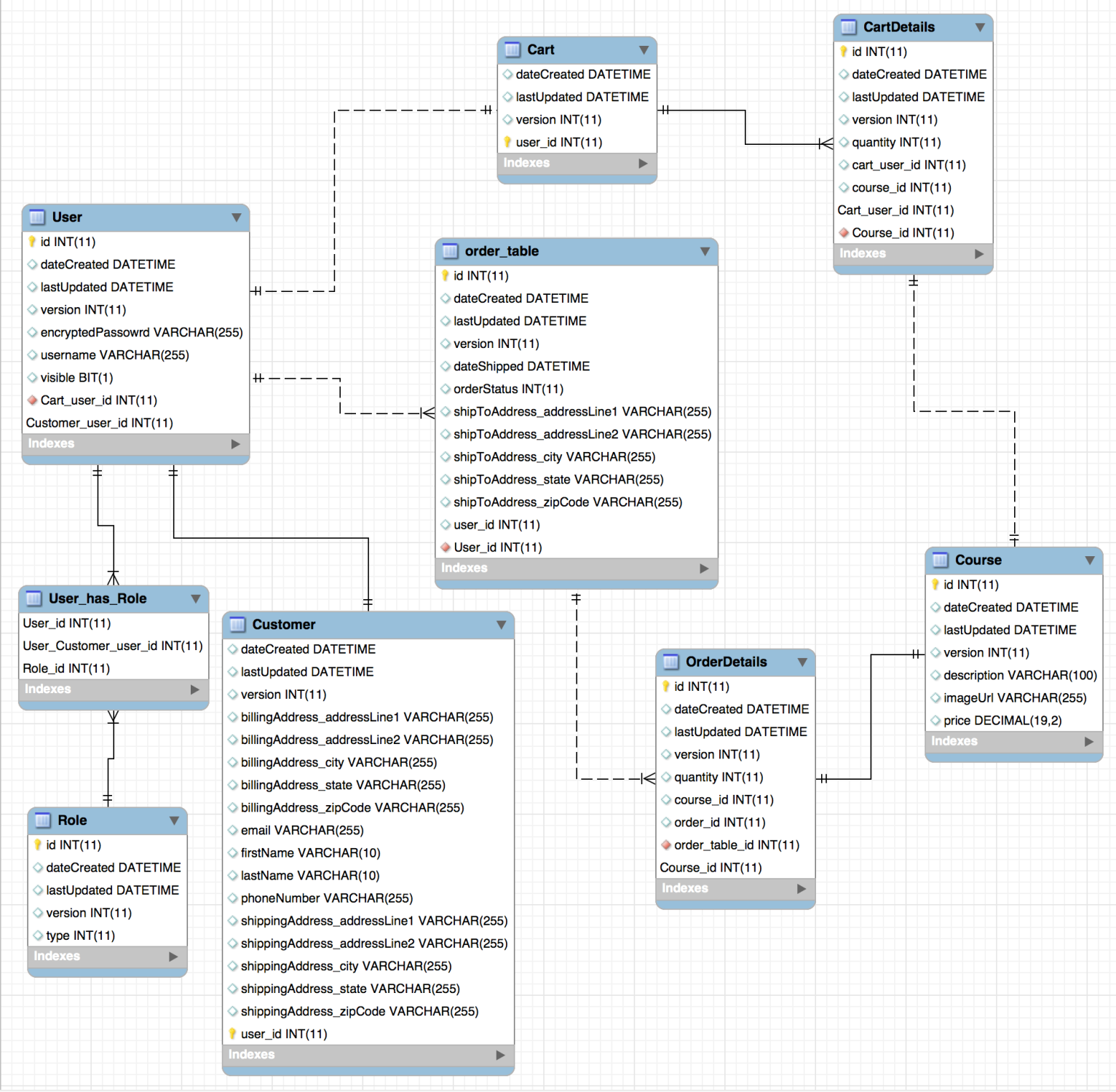
* Course:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Key | Field Name | Data Type | Allow Null | Default Value | Table-Field-Ref | Description |
| PK | id | INT(11) | No |  |  | Primary key of Course table |
|  | version | VARCHAR(255) | No |  |  | Version of Course Details |
|  | dateaCreated | DATETIME | Yes |  |  | Date create of Course Details |
|  | lastUpdated | DATETIME | Yes |  |  | Date last update of Course Details |
|  | description | VARCHAR(100) | No |  |  |  |
|  | imageUrl | VARCHAR(255) | No |  |  |  |
|  | price | DECIMAL(19,2) | No |  |  |  |

* Course Owner:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Key | Field Name | Data Type | Allow Null | Default Value | Table-Field-Ref | Description |
| PK | id | INT(11) | No |  |  | Primary key of Course Owner table |
|  | version | VARCHAR(255) | No |  |  | Version of Course Owner |
|  | dateaCreated | DATETIME | Yes |  |  | Date create of Course Owner |
|  | lastUpdated | DATETIME | Yes |  |  | Date last update of Course Owner |
|  | ownerType | INT(11) | No |  |  | Owner Type |

### Diagram



## Task Sheet 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Members Group** | | **Date Prepareation Of Activity Plan** | | | |
| **No** | **Name** | **Description Task** | **Start Date** | **End Date** | **Status** |
| 1 | Le Duc Anh | - design software architecture |  |  | Finished |
| 2 | Luu Duc Trung | - create user case  - database design, entities definition |  |  | Finished |
|  | | | | | |
| **Teacher** | | | **Team Leader** | | |
|  | | |  | | |
| **Ngo Phuoc Nguyen** | | | **Le Duc Anh** | | |