

**MINISTRY OF**

**EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

**Chess Online Learning System**

|  |  |
| --- | --- |
| **GROUP 04** | |
| **Group members** | Phạm Hoàng Tuyết Ngân – SE62778 (Leader)  Lưu Duy Hòa – SE62529  Đặng Văn Hoàng – SE62687 |
| **Supervisor** | Nguyễn Huy Hùng |
| **Ext. Supervisor** | N/A |
| **Capstone Project code** | COLS |

- Ho Chi Minh City, ***13th May, 2019*** -

*This page is intentionally left blank*

**Table of Contents**

[List of Table 6](#_Toc20099333)

[List of Figure 7](#_Toc20099334)

[1 .Capstone Project Register 9](#_Toc20099335)

[2. Report 1 11](#_Toc20099336)

[2.1. Project Information 11](#_Toc20099337)

[2.2. Introduction 11](#_Toc20099338)

[2.3. Current Situation 11](#_Toc20099339)

[2.4. Problem Definition 12](#_Toc20099340)

[2.5. Proposed Solution 12](#_Toc20099341)

[2.5.1 Feature functions 12](#_Toc20099342)

[2.5.2 Benefits and Trade-offs 12](#_Toc20099343)

[2.6. Functional Requirements 13](#_Toc20099344)

[2.7. Role and Responsibility 14](#_Toc20099345)

[3. Software Project Model 14](#_Toc20099346)

[4. Conceptual Diagram 15](#_Toc20099347)

[5. Use Case Diagram 16](#_Toc20099348)

[5.1. Overview Use Case1 16](#_Toc20099349)

[5.2 Use case Specification (Core Flow) 17](#_Toc20099350)

[5.2.1 <Learner> Overview Use Case 17](#_Toc20099351)

[5.2.2 <System Bot> Overview Use Case 29](#_Toc20099352)

[5.2.3 <Instructor> Overview Use Case 32](#_Toc20099353)

[5.2.4 <Admin> Overview Use Case 43](#_Toc20099354)

[6. System Architecture Design2 47](#_Toc20099355)

[7. Component Diagram1 50](#_Toc20099356)

[8. Class Diagram 1 51](#_Toc20099357)

[9. Entity Relationship Diagram3 55](#_Toc20099358)

[10. Interactive Diagram1 56](#_Toc20099359)

[10.1 Learner 56](#_Toc20099360)

[10.1.1 Enrol Course 56](#_Toc20099361)

[10.1.2 Create Review 57](#_Toc20099362)

[10.1.3 View List Of Suggested Course 58](#_Toc20099363)

[10.2 Instructor 58](#_Toc20099364)

[10.2.1 Create Course 58](#_Toc20099365)

[10.2.2 Submit Course 59](#_Toc20099366)

[10.2.3 Create Lesson 60](#_Toc20099367)

[10.2.4 Edit Lesson 61](#_Toc20099368)

[10.3 Admin (Update Course Status) 62](#_Toc20099369)

[10.4 System Bot (Play Chess With Learner) 63](#_Toc20099370)

[11. Database Relationship Diagram 64](#_Toc20099371)

[11.1 Physical Diagram 64](#_Toc20099372)

[11.2 Data dictionary 66](#_Toc20099373)

[12. Algorithms 70](#_Toc20099374)

[12.1 Convert chess move list from database to display list 70](#_Toc20099375)

[12.1.1 Definition 70](#_Toc20099376)

[12.1.2 Define problem 70](#_Toc20099377)

[12.1.3 Solution 71](#_Toc20099378)

[12.1.4 Complexity 72](#_Toc20099379)

[12.2 Building chess bot 72](#_Toc20099380)

[12.2.1 Definition 72](#_Toc20099381)

[12.2.2 Define problem 73](#_Toc20099382)

[12.2.3 Solution 73](#_Toc20099383)

[12.2.4 Flow chart 76](#_Toc20099384)

[12.3 Building Elo rating system 76](#_Toc20099385)

[12.3.1 Definition 76](#_Toc20099386)

[12.3.2 Define problem 76](#_Toc20099387)

[12.3.3 Solution 76](#_Toc20099388)

[12.4 Finding Similar User 78](#_Toc20099389)

[12.4.1 Definition 78](#_Toc20099390)

[12.4.2 Define problem 78](#_Toc20099391)

[12.4.3 Solution 78](#_Toc20099392)

[12.4.4 Example 78](#_Toc20099393)

[12.4.5 Complexity 78](#_Toc20099394)

[12.5 Course Rating Prediction (User-based filtering) 78](#_Toc20099395)

[12.5.1 Definition 78](#_Toc20099396)

[12.5.2 Define problem 79](#_Toc20099397)

[12.5.3 Solution 79](#_Toc20099398)

[12.5.4 Example 79](#_Toc20099399)

[12.5.5 Complexity 79](#_Toc20099400)

[12.6 Item-based filtering 79](#_Toc20099401)

[12.6.1 Definition 79](#_Toc20099402)

[12.6.2 Define problem 79](#_Toc20099403)

[12.6.3 Solution 79](#_Toc20099404)

[12.6.4 Example 80](#_Toc20099405)

[12.6.5 Complexity 80](#_Toc20099406)

[13. Future plan 80](#_Toc20099407)

[14. Appendix 81](#_Toc20099408)

# List of Table

[Table 1: Roles and Responsibilities 14](#_Toc20098265)

[Table 2: Enrol Course specification 18](#_Toc20098266)

[Table 3: View Enrolled Course specification 19](#_Toc20098267)

[Table 4: : Create Review specification 21](#_Toc20098268)

[Table 5: Read Interactive Lesson specification 22](#_Toc20098269)

[Table 6: Read Uninteractive Lesson specification 24](#_Toc20098270)

[Table 7: Do Exercise specification 26](#_Toc20098271)

[Table 8: View List Of Suggested Course specification 27](#_Toc20098272)

[Table 9: Play Chess With The Bot specification 29](#_Toc20098273)

[Table 10: Play Chess With Learner specification 30](#_Toc20098274)

[Table 11: Update Learner Skill Level specification 31](#_Toc20098275)

[Table 12: Make Course Suggestion For Learner specification 32](#_Toc20098276)

[Table 13: Create Course specification 34](#_Toc20098277)

[Table 14: Edit Course Status specification 36](#_Toc20098278)

[Table 15: Create Interactive Lesson specification 38](#_Toc20098279)

[Table 16: Create Uniteractive Lesson specification 40](#_Toc20098280)

[Table 17: <Instructor> Create Exercise specification 42](#_Toc20098281)

[Table 18: Create Category specification 44](#_Toc20098282)

[Table 19: Review New Course specification 46](#_Toc20098283)

[Table 20: Review Instructor specification 47](#_Toc20098284)

[Table 21: Front-end Servers Component Dictionary 50](#_Toc20098285)

[Table 22: Back-end Servers Component Dictionary 51](#_Toc20098286)

[Table 23: Class Dictionary 54](#_Toc20098287)

[Table 24: Entity data dictionary 56](#_Toc20098288)

[Table 25: Data Table Dictionary 66](#_Toc20098289)

[Table 26: Data Table Description 70](#_Toc20098290)

[Table 27: Move list dictionary 71](#_Toc20098291)

# List of Figure

[Figure 1: Scrum Model 14](#_Toc20098292)

[Figure 2: Conceptual diagram 15](#_Toc20098293)

[Figure 3: System Overview Use Case 16](#_Toc20098294)

[Figure 4: <Learner> Overview Use Case 17](#_Toc20098295)

[Figure 5: <Learner> Enrol Course 17](#_Toc20098296)

[Figure 6: <Learner> View Enrolled Course 19](#_Toc20098297)

[Figure 7: <Learner> Create Review 20](#_Toc20098298)

[Figure 8: <Learner> Read Interactive Lesson 21](#_Toc20098299)

[Figure 9: <Learner> Read Uninteractive Lesson 23](#_Toc20098300)

[Figure 10: <Learner> Do Exercise 24](#_Toc20098301)

[Figure 11: <Learner> View List Of Suggested Course 26](#_Toc20098302)

[Figure 12: <Learner> Play Chess With The Bot 27](#_Toc20098303)

[Figure 13: <System Bot> Overview Use Case 29](#_Toc20098304)

[Figure 14: <System Bot> Play Chess With Learner 29](#_Toc20098305)

[Figure 15: <System Bot> Update Learner Skill Level 30](#_Toc20098306)

[Figure 16: <System Bot> Make Course Suggestion For Learner 31](#_Toc20098307)

[Figure 17: <Instructor> Overview Use Case 32](#_Toc20098308)

[Figure 18: <Instructor> Create Course 33](#_Toc20098309)

[Figure 19: <Instructor> Edit Course Status 34](#_Toc20098310)

[Figure 20: <Instructor> Create Interactive Lesson 36](#_Toc20098311)

[Figure 21: <Instructor> Create Uninteractive Lesson 38](#_Toc20098312)

[Figure 22: <Instructor> Create Exercise 40](#_Toc20098313)

[Figure 23: <Admin> Overview Use Case 43](#_Toc20098314)

[Figure 24: <Admin> Create Category 43](#_Toc20098315)

[Figure 25: <Admin> Review New Course 45](#_Toc20098316)

[Figure 26: <Admin> Review Instructor 46](#_Toc20098317)

[Figure 27: System architecture overview 48](#_Toc20098318)

[Figure 28: Front-end Servers Component diagram 50](#_Toc20098319)

[Figure 29: Back-end Server Component diagram 51](#_Toc20098320)

[Figure 30: Class Diagram 52](#_Toc20098321)

[Figure 31: Entity relationship diagram (ERD) 55](#_Toc20098322)

[Figure 32: Sequence Diagram - <Learner> Enrol Course 57](#_Toc20098323)

[Figure 33: Sequence Diagram - <Learner> Create Review 58](#_Toc20098324)

[Figure 34: Sequence Diagram - <Learner> View List Of Suggested Course 58](#_Toc20098325)

[Figure 35: Sequence Diagram - <Instructor> Create Course 59](#_Toc20098326)

[Figure 36: Sequence Diagram - <Instructor> Submit Course 60](#_Toc20098327)

[Figure 37: Sequence Diagram - <Instructor> Create Lesson 61](#_Toc20098328)

[Figure 38: Sequence Diagram - <Instructor> Edit Lesson 62](#_Toc20098329)

[Figure 39: Sequence Diagram - <Admin> Update Course Status 63](#_Toc20098330)

[Figure 40: Sequence Diagram - <Admin> Update Course Status 64](#_Toc20098331)

[Figure 41: Physical diagram 65](#_Toc20098332)

[Figure 42: Target Move List 70](#_Toc20098333)

[Figure 43: Sample move list from database 71](#_Toc20098334)

[Figure 44: Example simple linked list 71](#_Toc20098335)

[Figure 45: Example tree 72](#_Toc20098336)

[Figure 46: Move list result 72](#_Toc20098337)

[Figure 47: A chessboard with square in AN 73](#_Toc20098338)

[Figure 48: Sample rank in FEN String 74](#_Toc20098339)

[Figure 49: Sample Position in FEN String 74](#_Toc20098340)

[Figure 50: Building Chess Bot Flow chart 76](#_Toc20098341)

# 1 .Capstone Project Register

**

**CAPSTONE PROJECT REGISTER**

Class: Duration time: from …. To /…..

(\*) Profession: <Software Engineer> Specialty: <ES> <IS>

x

(\*) Kinds of person make registers: Lecturer Students

x

1. Register information for supervisor (if have)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Full name** | **Phone** | **E-Mail** | **Title** |
| Supervisor 1 | Nguyễn Huy Hùng |  | hungnh@fpt.edu.vn | Mr. |

2. Register information for students (if have)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Full name** | **Student code** | **Phone** | **E-mail** | **Role in Group** |
| Student 1 | Phạm Hoàng Tuyết Ngân | SE62778 | 0974072150 | <nganphtse62778@fpt.edu.vn> | Leader |
| Student 2 | Lưu Duy Hòa | SE62529 | 0762629668 | [hoaldse62529@fpt.edu.vn](mailto:hoaldse62529@fpt.edu.vn) | Member |
| Student 3 | Đặng Văn Hoàng | SE62687 | 0902388104 | <hoangdvse62687@fpt.edu.vn> | Member |

3. Register content of Capstone Project

(\*) 3.1. Capstone Project name:

English: Chess Online Learning System

Vietnamese: Hệ Thống Dạy Đánh Cờ Vua

Abbreviation: COLS

(\*) 3.2. Main proposal content (including result and product)

1. Theory and practice (document):
   * Student should apply the software development process and the UML 2.0 in modeling the system
   * Software artifacts include User Requirement, Software Requirement Specification, Architecture Design, Detail Design, System Implementation and Testing Document, Installation Guide, sources code, and deployable software packages
   * Server side technique:
     + Database design, OOA, OOD, OOP, MVC, Java or .Net technology, …
     + Apache Lucene, ElasticSearch
   * Client side technique
     + HTML5, CSS, JavaScript, JQuery, Ajax
     + Mobile Platform (iOS, Android)
2. Program:

Build a web site allowing users to learn chess. The following main features need to be implemented:

- Instructor can create/edit/remove chess courses

- Auto bot plays chess with learners

- Bot evaluates levels of learners and suggests appropriate courses

- Learner register and enrol in courses based on user’s level

- Support learning chess by articles and practice on the website

- Checking practice results and open new content

- Manage courses and learning process

1. Other products:

* All of management functions of the system must be implemented to support the operating system.

4. Other comment (propose all relative thing if have)

N/A

|  |  |
| --- | --- |
| **Supervisor (If have)**  *(Sign and full name)* | HCM city, date 26/04/2019 …..  **On behalf of Registers**  *(Sign and full name)* |

# 2. Report 1

## 2.1. Project Information

* Project name: **Chess Online Learning System**
* Project Code: **COLS**
* Product Type: **Website application**
* Start Date: **13th May, 2019**
* End Date: **October, 2019**

## 2.2. Introduction

In this document, we introduce an e-learning portal for people who are willing to learn, improve and entertain the chess. Chess is a strategic board game for 2 players. Chess helps people to relax and improve their thinking. Nowadays, chess is a popular sport in the world. Currently, there are so many ways, which learn chess such as reading books, directly teaching and watching the online course. In a short research from wikihow.com, the most effective learning way is direct teaching or face to face practice, but it also takes too much cost. Meanwhile, learning online from e-portal currently, are the most successful way. However, there are some preventions in learning on the e-portal which could be named like confliction in communicating, boring traditional learning, non-evaluating ranked leaner and non-suggestion road for better playing. At last, reading book is the worst effective learning chess for general people, because it does not interact directly with the learner. Based on our researches and analysis, we proposed an e-learning system to help people to learn chess and improve their level. Especially, the main target is Vietnamese people.

In our system, the player can learn chess by tutorial and practice with exercises, and the instructor can post their course to system. Moreover, the system can be a bridge between chess players and instructors by providing a platform to help their communication.

This document also describes our working process in four months includes our perspective in the system, component designs, and detailed core workflows. We hope the system and our solution will help to resolve the problems from learning chess for the Vietnamese community.

## 2.3. Current Situation

When starting with chess, generally people are going to search for information and looking for a suitable tutorial for themselves online or offline. In the offline method, both instructors and learners are depended on their time, location and costly for infrastructure. By the way, for the online method, the most factor that brings the pain to leaner that is going to make them avoid choosing a website is language. Not only for a learner who does not know the common language, but also for the learner who knows. The second problem is the tutorials, which is not highly detailed or has bored traditional teaching ways. Therefore, the tutorial is static while the learner always wants to be challenged. Moreover, it causes frustration inside the learner.

## 2.4. Problem Definition

From the above current situation, we recognize some disadvantages:

* There is some online interactive courses chess (chess.com/lessons, lichess.org/learn#), but it is fully English, so Vietnamese people are hard to learn.
* Some chess course by Vietnamese is provided, but they do not have any place to practice.
* An instructor, who wants to teach chess must prepare a place with full infrastructure.
* An offline learner must depend on instructor about learning time and location.

## 2.5. Proposed Solution

Our proposed solution is building a system, which named is Chess Online Learning System (COLS) for Vietnamese people with the following functions:

### **2.5.1 Feature functions**

* + Provide interactive chess courses with practice environment by Vietnamese for the learner.
  + Provide an environment for the instructor, who wants to share their chess knowledge.
  + Provide chess bot, who can play automatically with the learner.
  + Collect learner learning history and suggest suitable courses.
  + Support Elo rating system to classify the level of learners and courses. A learner can only learn courses which are equal or lower than their level.

### **2.5.2 Benefits and Trade-offs**

* Benefits:
  + - An instructor does not need to spend the cost to prepare face to face classes.
    - Flexible in learning time and place.
    - Everything is free.
    - Connecting people who want to learn chess and share chess knowledge without caring about their location.
    - A learner can practice conveniently in our system.
* Trade-offs:
  + - The type of lesson or exercise is limited.
    - The course and user activities are checking manually.
    - The instructor benefits are unclear, there is no method to collect money for them, and it will be developed in the next phase.
    - There is only support to play chess with an automated bot, without supporting peer-to-peer playing.

## 2.6. Functional Requirements

Function requirements of the system are listed below:

* Authentication and Authorization:
  + Authenticate and authorize users to learn or manage system content and user.
* New account registration:
  + A guest can sign up to be a learner or an instructor.
  + A new learner must estimate their chess skill level to start the learning progress. Skill level will be converted to a point for the learner.
  + A new instructor must apply their information to prove competence.
* Manage profile
  + An authorized user can view and edit his/her information.
* View courses
  + A guest or learner can search for any course by course name.
  + A guest or learner can filter courses by category and skill level.
  + A guest or learner can view course details.
* Enrol course
  + A learner can enrol their wanted course if they have enough skill level.
* Learn course
  + A learner can learn any course which he/she have enrolled.
  + A learner can review course he/she has already learned.
  + Learning includes learn chess game, static content and do exercises
  + A learner can manage his/her learning progress.
* Course suggestion
  + A learner can view suggested list which he/she should learn.
* Manage course
  + An instructor can manage his/her courses such as lessons, exercises or feedbacks.
  + An instructor can submit his/her course and wait for review. If a course is rejected, the instructor can restore them else the instructor can unpublish their course to re-build content.
  + An admin can publish or unpublish the course.
* Manage category
  + A course must be classified to some categories, which were given from system.
  + An admin can manage system categories.
* Manage user
  + An admin can control learner or instructor account status.
  + An admin can review new instructor registration.
* Play chess
  + A learner can play chess with a bot.
  + Learners can increase their point by win or draw games.
  + Learners can lose their point if they lose the game.
* Manage the learner skill level
  + An Elo rating system is applied to calculate learner point and evaluate learner chess skill level. Each level will be linked with point range.
  + The system can classify learner skill level based on their point.

## 2.7. Role and Responsibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Full Name** | **Role** | **Position** | **Contact** |
| 1 | Nguyễn Huy Hùng | Project Manager | Supervisor | hungnh@fpt.edu.vn |
| 2 | Phạm Hoàng Tuyết Ngân | Developer | Leader | nganphtse62778@fpt.edu.vn |
| 3 | Lưu Duy Hòa | Developer | Member | hoaldse62529@fpt.edu.vn |
| 4 | Đặng Văn Hoàng | Developer | Member | hoangdvse62687@fpt.edu.vn |
| 5 | Lâm Thanh Tùng | Developer | Member | tungltse61895@fpt.edu.vn |

Table : Roles and Responsibilities

# 3. Software Project Model

As the project problem which we mention above, we choose Scrum Model to manage project development with each sprint is 5 working days. This model will help us:

- Scrum team always work together to reach the project goal. Team member working progress is daily updated.

- Minimize risks when changing requirement.

- The working cycle “Learn - Revise – Apply” is done concurrently and continuously, a team member can apply new research and get the result quickly.

- Product coding is focused rather than document to deliver higher product value.

More information: <https://www.scrum.org/>

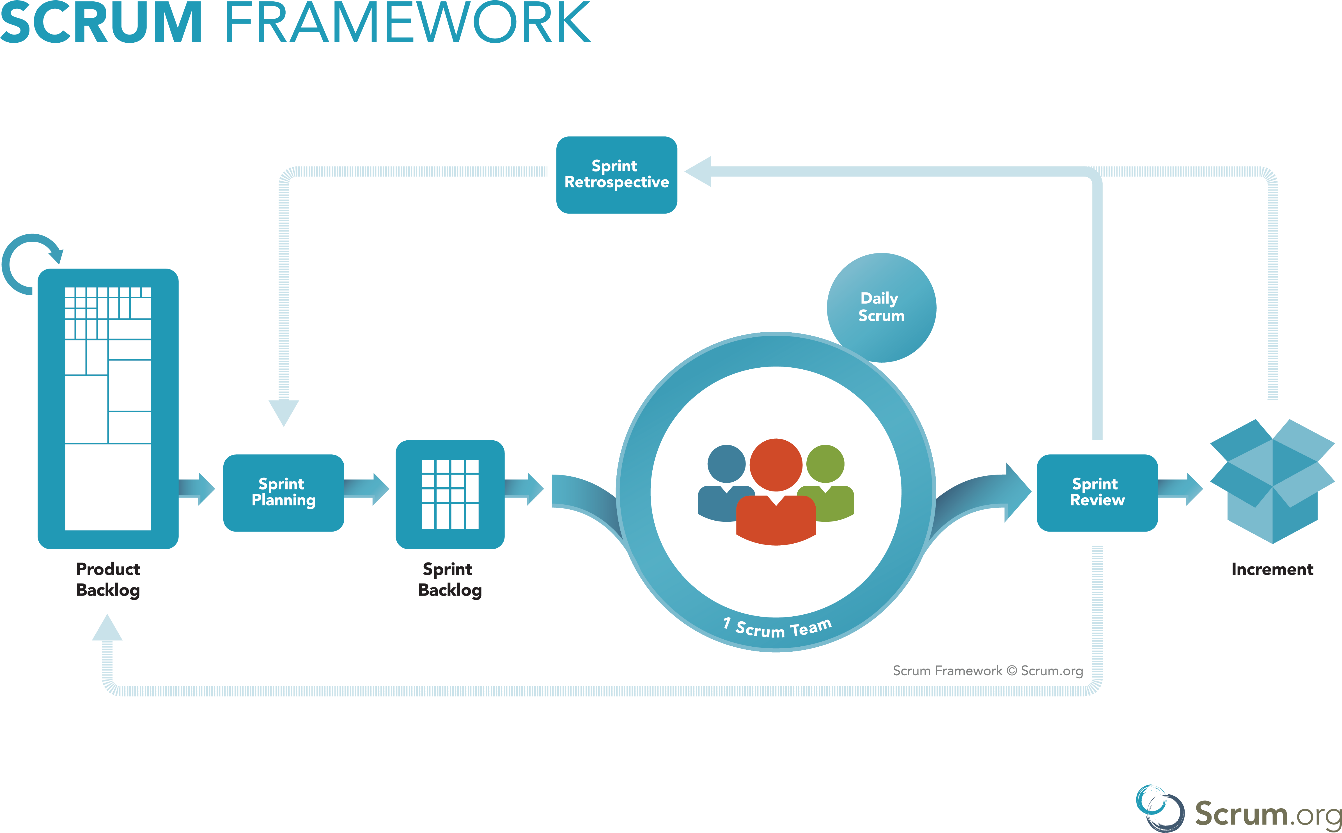


Figure : Scrum Model

# 4. Conceptual Diagram

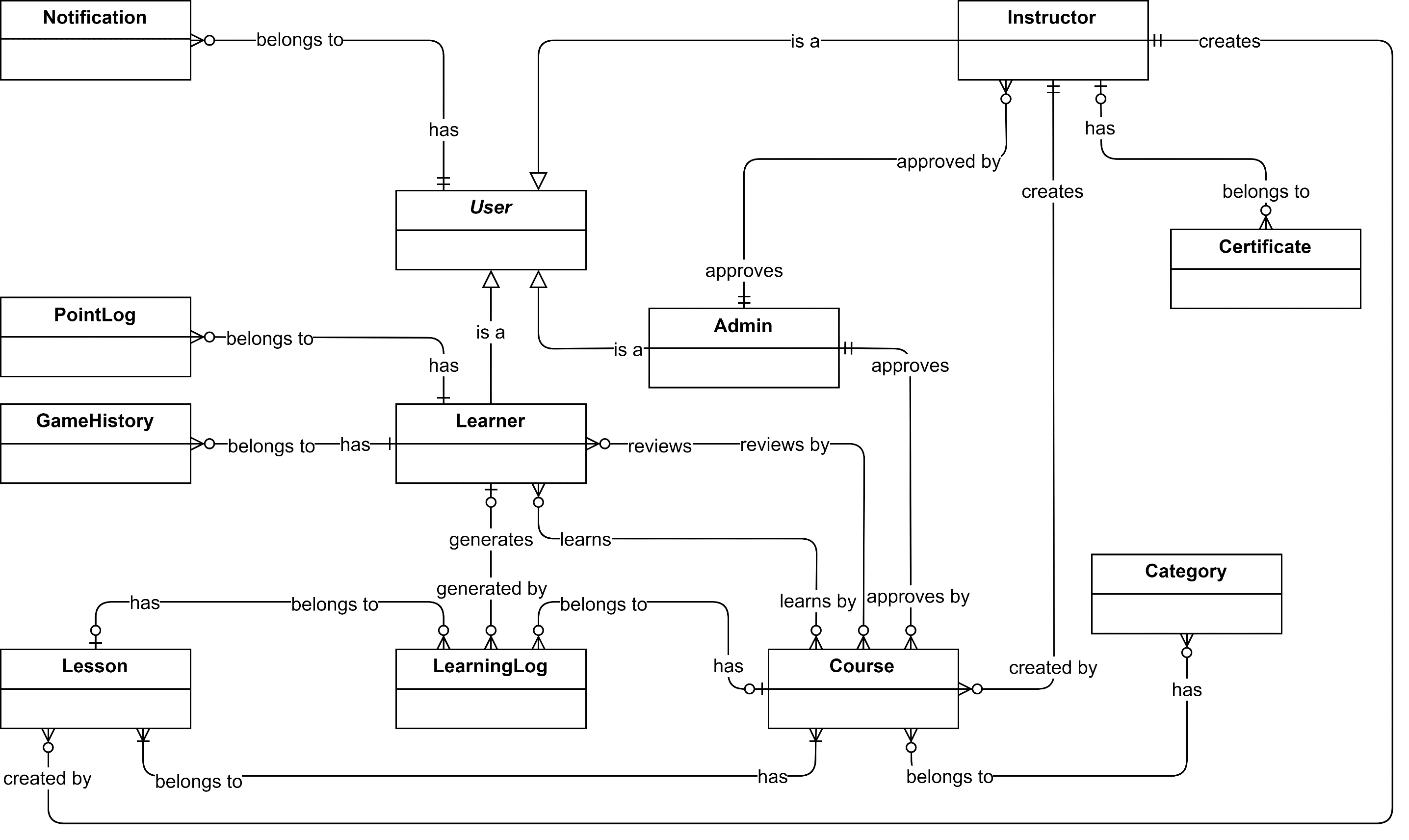


Figure : Conceptual diagram

# 5. Use Case Diagram

## 5.1. Overview Use Case1

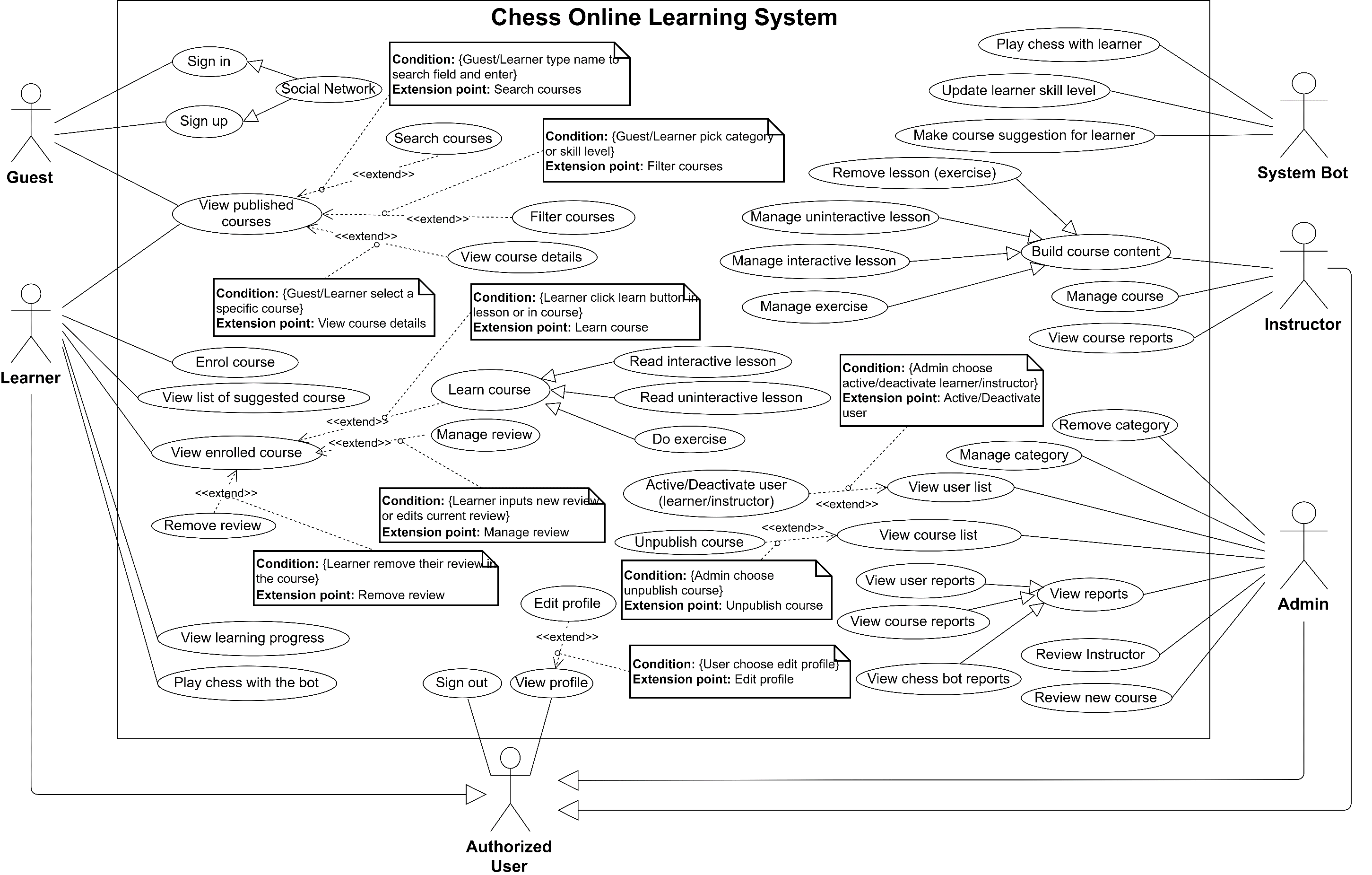


Figure : System Overview Use Case

## 5.2 Use case Specification (Core Flow)

### **5.2.1 <Learner> Overview Use Case**

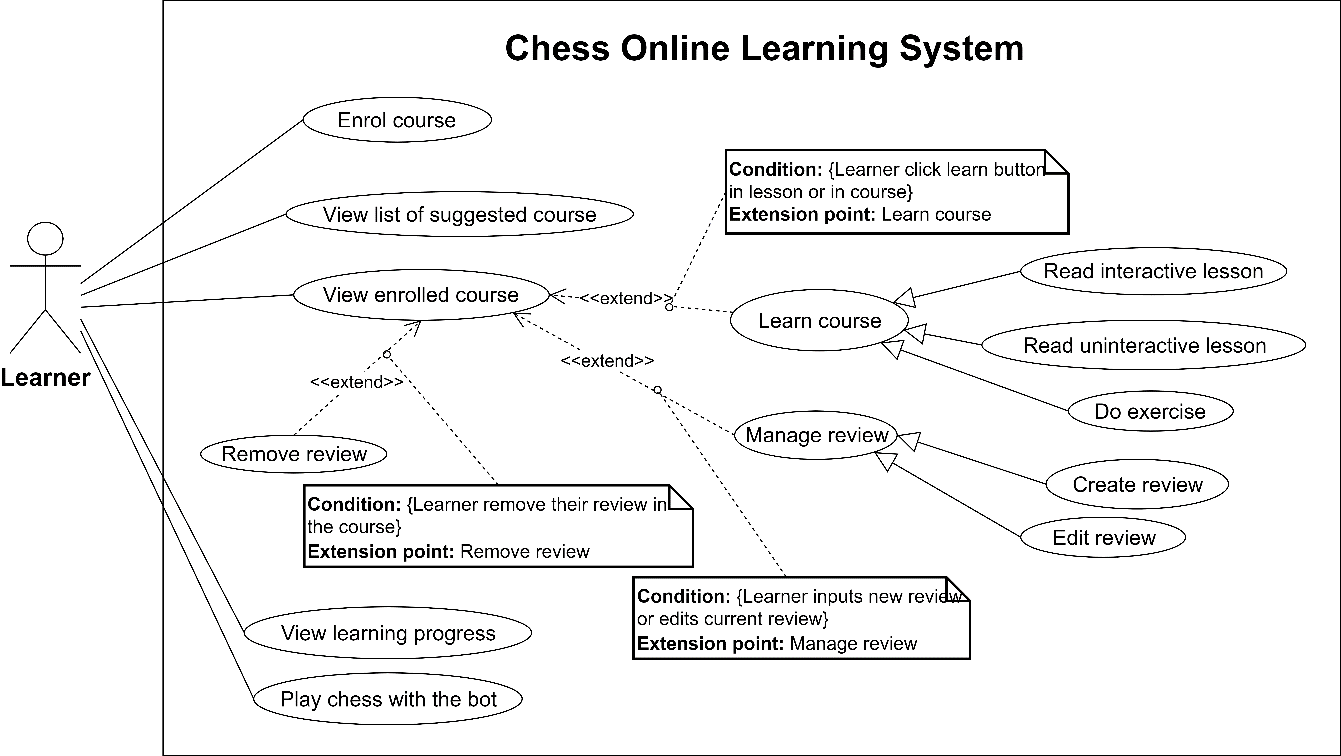


Figure : <Learner> Overview Use Case

#### 5.2.1.1 <Learner> Enrol Course

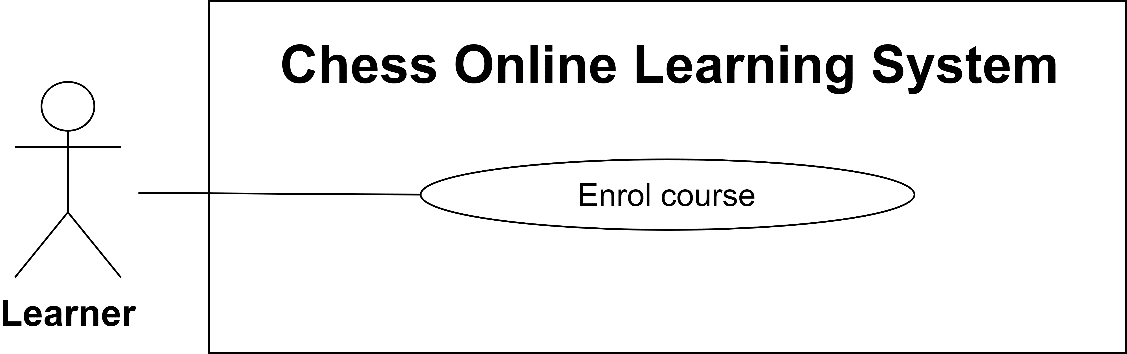


Figure : <Learner> Enrol Course

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE –**  **UC\_LE.01** | | | |
| **Use Case No.** | UC\_LE.01 | **Use Case Version** | 2.0 |
| **Use Case Name** | Enrol Course | | |
| **Author** | Lưu Duy Hòa | | |
| **Date** | 20/07/2019 | **Priority** | High |
| **Actor:**   * Learner   **Summary:**   * This use case allows the learner to enrol system course.   **Goal:**   * Learner can enrol to learn a specific course under their skill level.   **Triggers:**   * Learner sends the enrol course command.   **Preconditions:**   * User must sign in into the system with role Learner.   **Postconditions:**   * Success: System displays enrol course success messages. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Learner navigates to “Đăng ký khóa học” from the course details page. | System shows confirmation popup:   * “Xác nhận”: button * ”Hủy”: button | | 2 | Learner click “Xác nhận” button on the confirmation popup.  [Alternative 1] | System closes the confirmation popup and shows a message “Đăng kí khóa học thành công”.  System unlocks lesson list.  The “Đăng ký khóa học” button is changed to “Bắt đầu học” button.  [Exceptions 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Learner clicks on “Hủy” button. | System closes confirmation popup. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | The Learner has not enough required skill level to enrol. | System shows an error message “Bạn chưa đủ điều kiện để đăng kí khóa học này. Xin thử lại các khóa học khác!”. |   **Relationships:** N/A  **Business Rules:**   * System will show the enrol button for learners in the course details page. * Courses in the system require a skill level to enrol. * A learner can only enrol courses whose skill level is lower or equal with Learner skill level. | | | |

Table : Enrol Course specification

#### 5.2.1.2 <Learner> View Enrolled Course

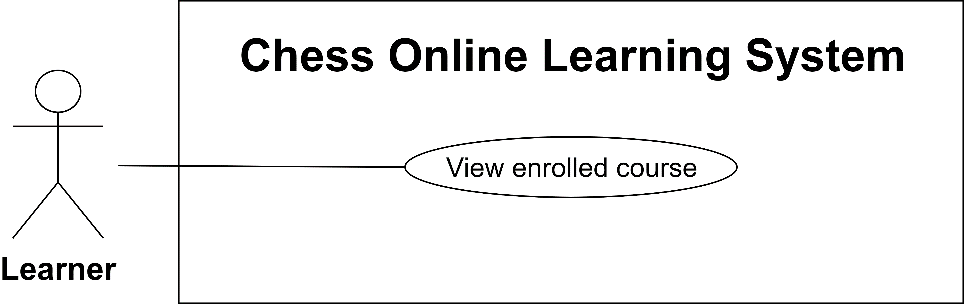


Figure : <Learner> View Enrolled Course

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE –**  **UC\_LE.02** | | | |
| **Use Case No.** | UC\_LE.02 | **Use Case Version** | 2.0 |
| **Use Case Name** | View Enrolled Course | | |
| **Author** | Lưu Duy Hòa | | |
| **Date** | 20/07/2019 | **Priority** | High |
| **Actor:**   * Learner   **Summary:**   * This use case allows the learner can view an enrolled course.   **Goal:**   * Learner can view a specific course which they enrolled.   **Triggers:**   * Learner sends view enrolled course command.   **Preconditions:**   * User must sign in into the system with role Learner. * Learner had enrolled in the course.   **Postconditions:**   * Success: System displays enrolled course page. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Learner navigates to a course which they enrolled. | System shows course details page and enables review area, unlock lessons. |   **Alternative Scenario:***N/A*  **Exceptions:** N/A  **Relationships:** N/A  **Business Rules:**   * Learner can learn course or learn a specific lesson in enrolled courses. * Learner can review course which is enrolled. | | | |

Table : View Enrolled Course specification

#### 5.2.1.3 <Learner> Create Review

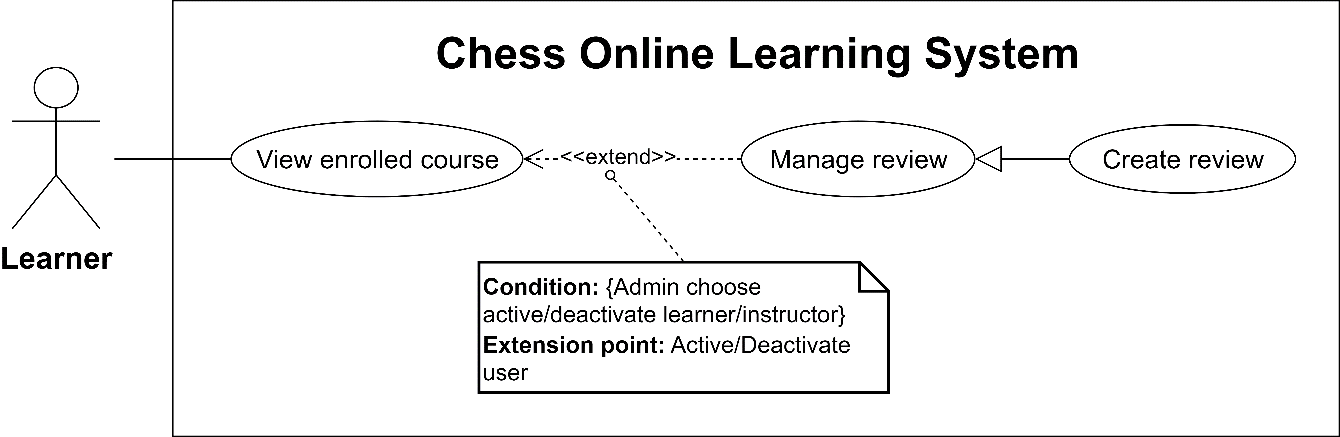


Figure : <Learner> Create Review

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_LE.03** | | | |
| **Use Case No.** | UC\_LE.03 | **Use Case Version** | 2.0 |
| **Use Case Name** | Create Review | | |
| **Author** | Lưu Duy Hòa | | |
| **Date** | 20/07/2019 | **Priority** | Normal |
| **Actor:**   * Learner   **Summary:**   * This use case allows the learner to create a review to course.   **Goal:**   * User can review a specific course.   **Triggers:**   * Learner sends the command to review the course.   **Preconditions:**   * User must sign in into the system with role Learner. * Learner had enrolled in the course.   **Postconditions:**   * Success: System add new review for course successfully. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | On the course details page, learner moves to review area. | System highlights the review area:   * Đánh giá của bạn: rating * Bình luận: text field * Đăng: button   System required information:   * Rating must be chosen | | 2 | Learner clicks “rating” and input text field. | [Exceptions 1] | | 3 | Learner clicks “Đăng” button.  [Alternative 1] | System re-calculates course review and displays new review on the top of list.  [Exception 2] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Learner removes his action. | There is no review added. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **Step** | **Cause** | **System Response** | | 1 | Learner does not rate review. | System shows the error message “Điểm đánh giá phải lớn hơn 0”. | | 2 | Learner has reviewed on this course before. | System show error message “Đánh giá không thành công. Bạn đã đánh giá khóa học này rồi”. |   **Relationships:** UC\_LE.02  **Business Rules:**   * Learner can only review when enrolled course. * Each learner can only have 1 review in each course. * Learner review will be shown on the top of Review list. * Course review is re-calculated. | | | |

Table : : Create Review specification

#### 5.2.1.4 <Learner> Read Interactive Lesson

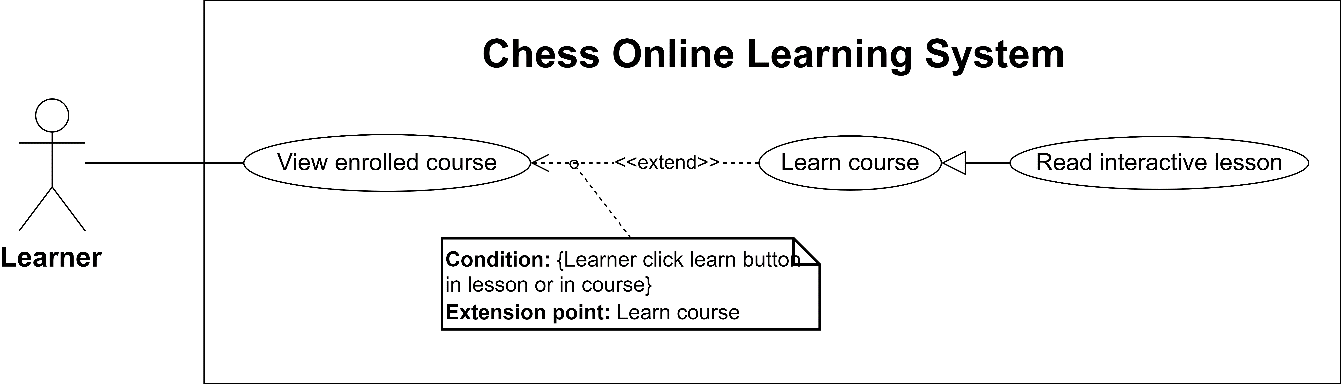


Figure : <Learner> Read Interactive Lesson

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_LE.06** | | | |
| **Use Case No.** | UC\_LE.06 | **Use Case Version** | 2.0 |
| **Use Case Name** | Read Interactive Lesson | | |
| **Author** | Lưu Duy Hòa | | |
| **Date** | 20/07/2019 | **Priority** | High |
| **Actor:**   * Learner   **Summary:**   * This use case allows the learner to read the interactive lesson.   **Goal:**   * Learner can read an interactive lesson.   **Triggers:**   * Learner sends the command to read the interactive lesson.   **Preconditions:**   * User signed in to the system as a learner successfully. * Learner had enrolled in the course.   **Postconditions:**   * Success: Learner can read lesson content. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Learner clicks on “Bắt đầu học” button.  [Alternative 1] | System redirects learner to the next lesson which learner did not learn.  If the type is interactive lesson, system shows the lesson content:   * Chessboard * Move arrow button group * Move content | | 2 | Learner can click on the move or arrow buttons.  [Alternative 2] | System shows the current chess position following the move and may be have a move content. | | 3 | Learner clicks “Bài tiếp” button to continue for the next lesson.  [Alternative 3] | System record that learner has finished this lesson and move to the next lesson. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Learner clicks on a specific lesson. | System redirects to the selected lesson. | | 2 | Learner clicks on the exit button to back to course detail. | System redirects learner to course detail. | | 3 | Learner clicks on “Bài trước” to go to previous lesson. | System shows previous detail lesson. The lesson is not logged to finish. |   **Exceptions:** N/A  **Relationships:** UC\_LE.02  **Business Rules:**   * Interactive lesson is a lesson which contains a chess-game is analysed. All chess-game moves and position following the move will be loaded and learner learns the lesson by view chess position with a specific move. * Interactive lesson is recorded to finish when learner request next lesson. * Course is finished when learner finish all lessons and exercises in the course. | | | |

Table : Read Interactive Lesson specification

#### 5.2.1.5 <Learner> Read Uninteractive Lesson

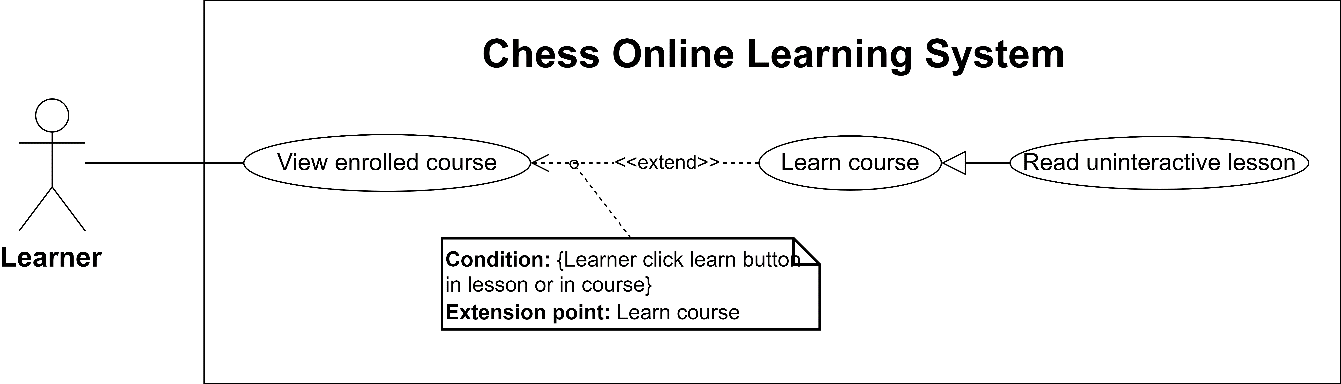


Figure : <Learner> Read Uninteractive Lesson

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_LE.07** | | | |
| **Use Case No.** | UC\_LE.07 | **Use Case Version** | 2.0 |
| **Use Case Name** | Read Uninteractive Lesson | | |
| **Author** | Đặng Văn Hoàng | | |
| **Date** | 23/03/2013 | **Priority** | High |
| **Actor:**   * Learner   **Summary:**   * This use case allows the learner to read the uninteractive lesson.   **Goal:**   * Learner can read an uninteractive lesson.   **Triggers:**   * Learner sends the command to read the uninteractive lesson.   **Preconditions:**   * User must sign in into the system with role Learner. * Learner had enrolled in the course.   **Postconditions:**   * Success: Learner can read the uninteractive lesson content. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Learner clicks on “Bắt đầu học” button.  [Alternative 1] | System redirects learner to the next lesson which learner did not learn. | | 2 | If lesson type is uninteractive, the learner can read lesson content.  [Alternative 2] |  | | 3 | Learner clicks “Bài tiếp” button to continue for the next lesson.  [Alternative 3] | System record that learner has finished this lesson and move to the next lesson. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Learner clicks on a specific lesson. | System redirects to the selected lesson. | | 2 | Learner clicks on the exit button to back to course detail. | System redirects learner to course detail. | | 3 | Learner clicks “bài trước” to previous lesson. | System shows previous detail lesson. Lesson is not logged to finish. |   **Exceptions:** N/A  **Relationships:** UC\_LE.02  **Business Rules:**   * Uninteractive lesson content is text and media, which is a post from WYSIWYG Editor. * Uninteractive is recorded to finish when learner request next lesson. * Course is finished when the learner finishes all lessons and exercises in the course. | | | |

Table : Read Uninteractive Lesson specification

#### 5.2.1.6 <Learner> Do Exercise

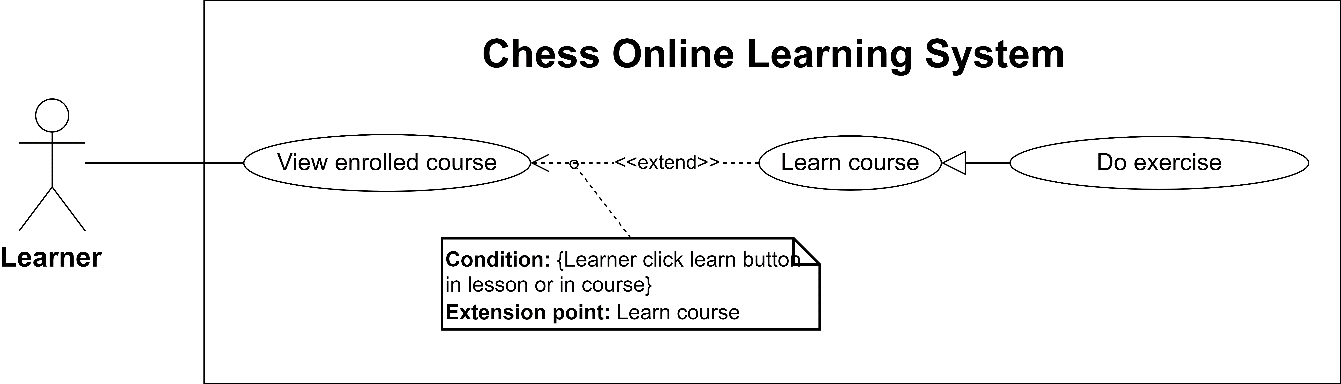


Figure : <Learner> Do Exercise

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_LE.08** | | | |
| **Use Case No.** | UC\_LE.08 | **Use Case Version** | 2.0 |
| **Use Case Name** | Do Exercise | | |
| **Author** | Phạm Hoàng Tuyết Ngân | | |
| **Date** | 20/07/2019 | **Priority** | High |
| **Actor:**   * Learner   **Summary:**   * This use case allows the learner to do exercise.   **Goal:**   * Learner can do exercise.   **Triggers:**   * Learner sends the do exercise command.   **Preconditions:**   * User signed in as a learner successfully. * Learner had enrolled in the course.   **Postconditions:**   * Success: Learner can do exercise by performing their chess move. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Learner clicks on “Bắt đầu học” button.  [Alternative 1] | System redirects learner to the next lesson which learner had not learned. | | 2 | If lesson type is exercise, learner read the question and perform moves to solve the problem.  [Alternative 2] | System checks learner move and sends a response for the wrong move or right move (it depends on the exercise content). If learner performs a right move, the system performs a move and change turn to learner again. This action is looped until learner solved the problem and finish the exercise.  [Exception 1] | | 3 | Learner clicks on “Bài kế tiếp” button to continue for the next lesson.  [Alternative 3] | System record that learner has finished this exercise and move to the next lesson. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Learner clicks on a specific lesson. | System redirects to the selected lesson. | | 2 | Learner clicks on the exit button to back to course detail. | System redirects learner to course detail. | | 3 | Learner clicks on “Bài trước” button to the previous lesson. | System shows the previous detail lesson. Lesson is not logged to finish. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **Step** | **Cause** | **System Response** | | 1 | Learner answers wrong. | System resets exercise and does not record this exercise is finished. |   **Relationships:** UC\_LE.02  **Business Rules:**   * Exercise is a lesson which provides a problem for the learner, learner must solve the problem to pass this lesson. * In our system exercise will contain chess position and require learner must perform their chess move. * Exercise is finished when learner solves the problem successfully. * Exercise contains 2 types: Bot answer and Manual answer. In the Bot answer, a learner must solve the problem to checkmate. In the Manual answer, there are 1 to 3 answer and learner only need to solve right 1 answer. * Course is finished when the learner finishes all lessons and exercises in the course. | | | |

Table : Do Exercise specification

#### 5.2.1.7 <Learner> View List Of Suggested Course

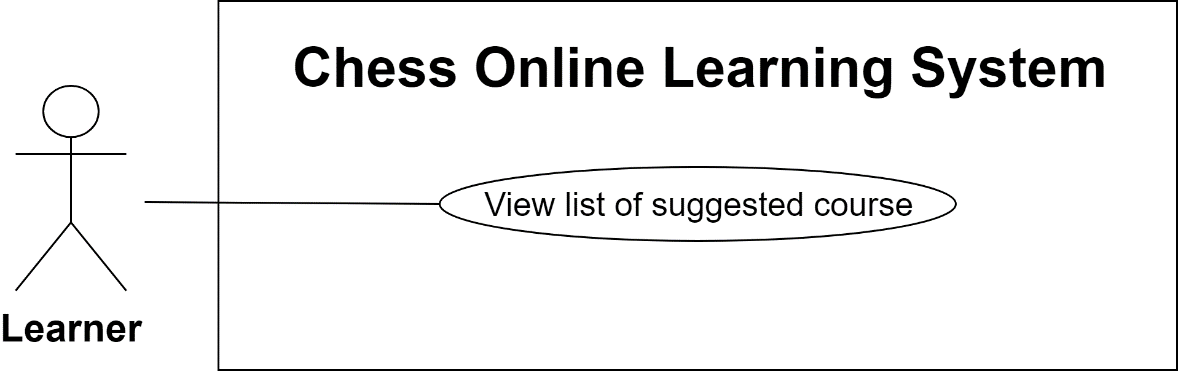


Figure : <Learner> View List Of Suggested Course

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_LE.10** | | | |
| **Use Case No.** | UC\_LE.10 | **Use Case Version** | 2.0 |
| **Use Case Name** | View List Of Suggested Course | | |
| **Author** | Đặng Văn Hoàng | | |
| **Date** | 09/09/2019 | **Priority** | High |
| **Actor:**   * Learner   **Summary:**   * This use case allows the learner to view the course suggested list.   **Goal:**   * Learner can his suggested courses.   **Triggers:**   * Learner sends the view course list or view course details command.   **Preconditions:**   * User signed in to the system as a learner successfully.   **Postconditions:**   * Success:  System displays a suggested course list for the learner. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Learner navigates to suggested course list from the course overview page.  [Alternative 1] | System calculates appropriate course for the learner from learner history and similar learners and displays to learner.  [Exception 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Learner views specific course details. | System displays appropriate courses for the learner based on the current course. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **Step** | **Cause** | **System Response** | | 1 | Learner has not learned any course. | Suggested list is empty. |   **Relationships:** N/A  **Business Rules:**   * Learner can view his suggested course list to get an appropriate course. | | | |

Table : View List Of Suggested Course specification

#### 5.2.1.8 <Learner> Play Chess With The Bot

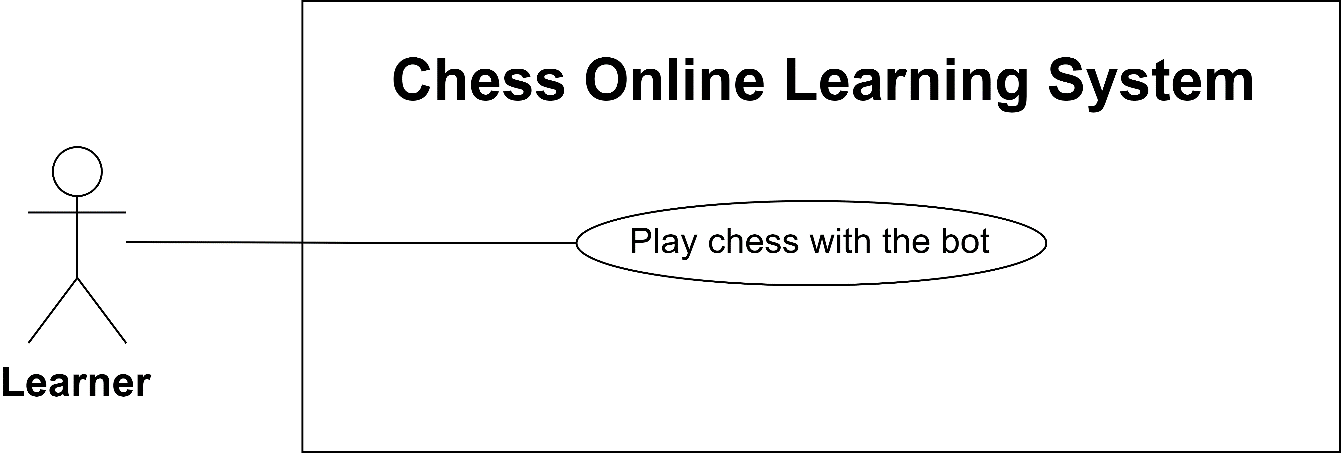


Figure : <Learner> Play Chess With The Bot

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_LE.11** | | | |
| **Use Case No.** | UC\_LE.11 | **Use Case Version** | 2.0 |
| **Use Case Name** | Play Chess With The Bot | | |
| **Author** | Phạm Hoàng Tuyết Ngân | | |
| **Date** | 20/07/2019 | **Priority** | High |
| **Actor:**   * Learner   **Summary:**   * This use case allows the learner to play chess with the bot.   **Goal:**   * Learner can play chess with the bot.   **Triggers:**   * Learner sends the playing with bot command.   **Preconditions:**   * User signed in to the system as a learner successfully.   **Postconditions:**   * Success:  System starts the game and system bot will autoplay with the learner. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Learner navigates to playing game page. | System moves to the playing chess page:   * Chessboard * Nước đi: move content-box * Chess game description: content-box * Ván đấu mới: button | | 2 | Learner clicks on the new game button. | System shows setting game dialogue:   * “Cấp độ”: switch button * “Chọn màu quân”: button group * “Thời gian đấu”: time picker * “Bắt đầu”: button * “Đóng”: button   System required information:   * Game Time must larger than 1 minutes. * Game colour will be random if the learner does not choose. | | 3 | Learner settings and clicks “Bắt đầu  button.  [Alternative 1] | System calculates win, draw and lose point for learner follow the Elo rating system.  The game is started.  [Exception 1] | | 4 | Learner performs a move in the learner turn.  [Alternative 2] | System gets the best move and performs move if the turn is bot. This action loop until the game is drawn or 1 player win.  System record each move on the move content-box.  When ending the game, system calculates a new point for the learner. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Learner clicks “Đóng” button on setting game dialogue. | System closes the setting game dialogue. The game does not start. | | 2 | Learner stops playing the game. | System record that learner has given up and lost the game. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **Step** | **Cause** | **System Response** | | 1 | Game Time less than 1 minutes. | The game cannot be started. |   **Relationships:** N/A  **Business Rules:**   * Learner can play chess with system bot to get bonus point. * Learner skill level can be changed after updating points. * System will record all moves in the game. | | | |

Table : Play Chess With The Bot specification

### **5.2.2 <System Bot> Overview Use Case**

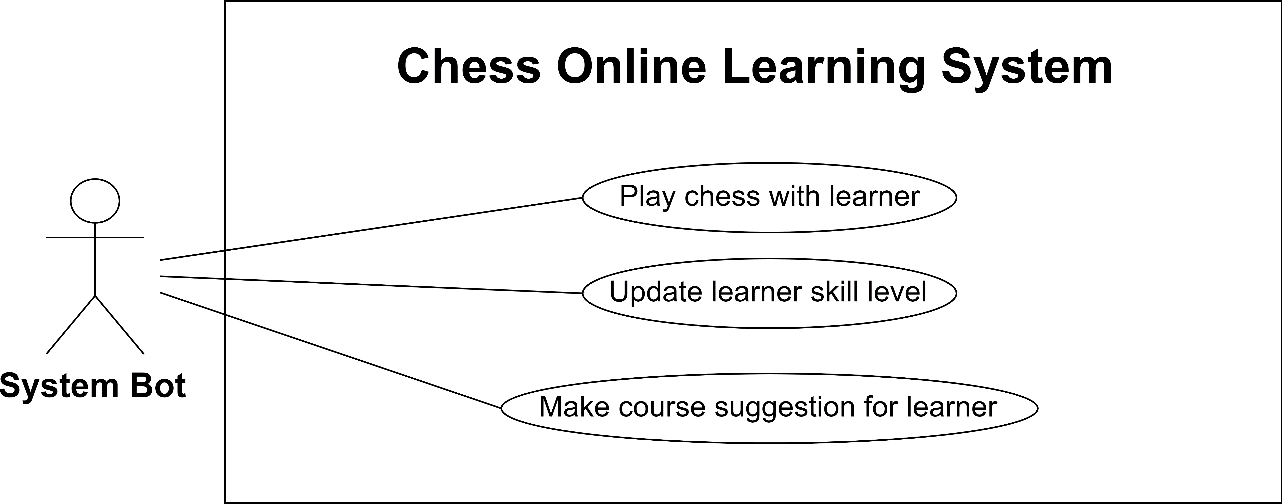


Figure : <System Bot> Overview Use Case

#### 5.2.2.1 <System Bot>  Play Chess With Learner

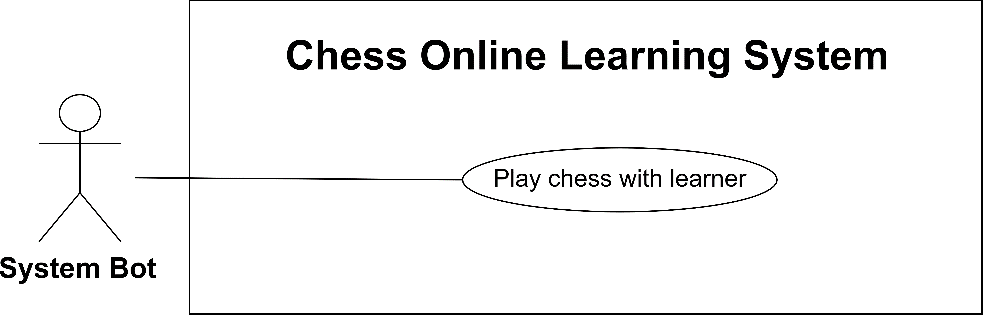


Figure : <System Bot> Play Chess With Learner

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_SB.01** | | | |
| **Use Case No.** | UC\_SB.01 | **Use Case Version** | 2.0 |
| **Use Case Name** | Play Chess With Learner | | |
| **Author** | Phạm Hoàng Tuyết Ngân | | |
| **Date** | 20/07/2019 | **Priority** | High |
| **Actor:**   * System Bot   **Summary:**   * This use case allows the system to autoplay chess with the learner.   **Goal:**   * System can autoplay chess.   **Triggers:**   * System sends the perform move commands when the turn is bot turn.   **Preconditions:** N/A  **Postconditions:**   * Success: New move is performed automatically. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | In bot turn, the system will analyse game position and calculate the best move. The new best move will be sent and performed in chessboard. | Move is performed and added to move content-box, turn is changed. |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** N/A  **Business Rules:**   * System can play chess with the learner. * System bot is triggered if a user sends position, the system will analyse and get the best move. * This use-case is used for playing chess with learner and response for the learner in exercise. | | | |

Table : Play Chess With Learner specification

#### 5.2.2.2 <System Bot>  Update Learner Skill Level

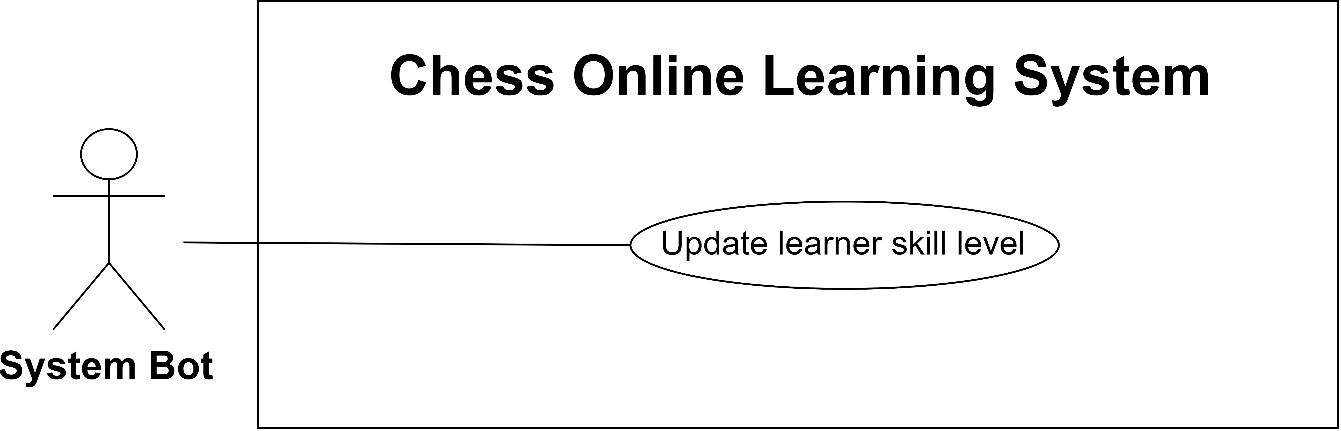


Figure : <System Bot> Update Learner Skill Level

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_SB.02** | | | |
| **Use Case No.** | UC\_SB.02 | **Use Case Version** | 2.0 |
| **Use Case Name** | Update Learner Skill Level | | |
| **Author** | Phạm Hoàng Tuyết Ngân | | |
| **Date** | 09/09/2019 | **Priority** | Normal |
| **Actor:**   * System Bot   **Summary:**   * This use case allows the system to auto-update chess skill level of the learner.   **Goal:**   * Learner skill level is auto-updated base on their point.   **Triggers:**   * Learner point is changed to another range of skill level.   **Preconditions:** N/A  **Postconditions:**   * Success: Learner skill level is updated. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Learner point is increased or decrease after the game is over and point is changed to another range of skill level. | Learner skill level is updated. |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** N/A  **Business Rules:**   * Learner skill level is specified by points. * Each skill level has a range of point. * System check learner skill level based on their points. | | | |

Table : Update Learner Skill Level specification

#### 5.2.2.3 <System Bot>  Make Course Suggestion For Learner

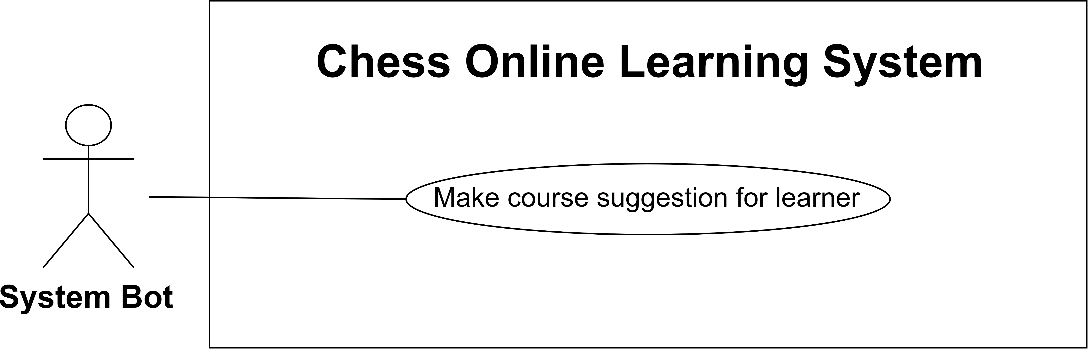


Figure : <System Bot> Make Course Suggestion For Learner

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_SB.03** | | | |
| **Use Case No.** | UC\_SB.03 | **Use Case Version** | 2.0 |
| **Use Case Name** | Make Course Suggestion For Learner | | |
| **Author** | Phạm Hoàng Tuyết Ngân | | |
| **Date** | 09/09/2019 | **Priority** | Normal |
| **Actor:**   * System Bot   **Summary:**   * This use case allows the system to suggest course for learners.   **Goal:**   * System can suggest appropriate courses for learners.   **Triggers:**   * Make course suggestion command is sent.   **Preconditions:** N/A  **Postconditions:**   * Success: System calculates and synthetic suggestion courses for each learner. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | System calculates course suggestion for learners periodically. | Suggestion course list for each learner. |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** N/A  **Business Rules:**   * Suggestion courses are based on the learning history of learners. * System finds new courses which are same with learner courses history list. * System finds new courses by similar learners. | | | |

Table : Make Course Suggestion For Learner specification

### **5.2.3 <Instructor> Overview Use Case**

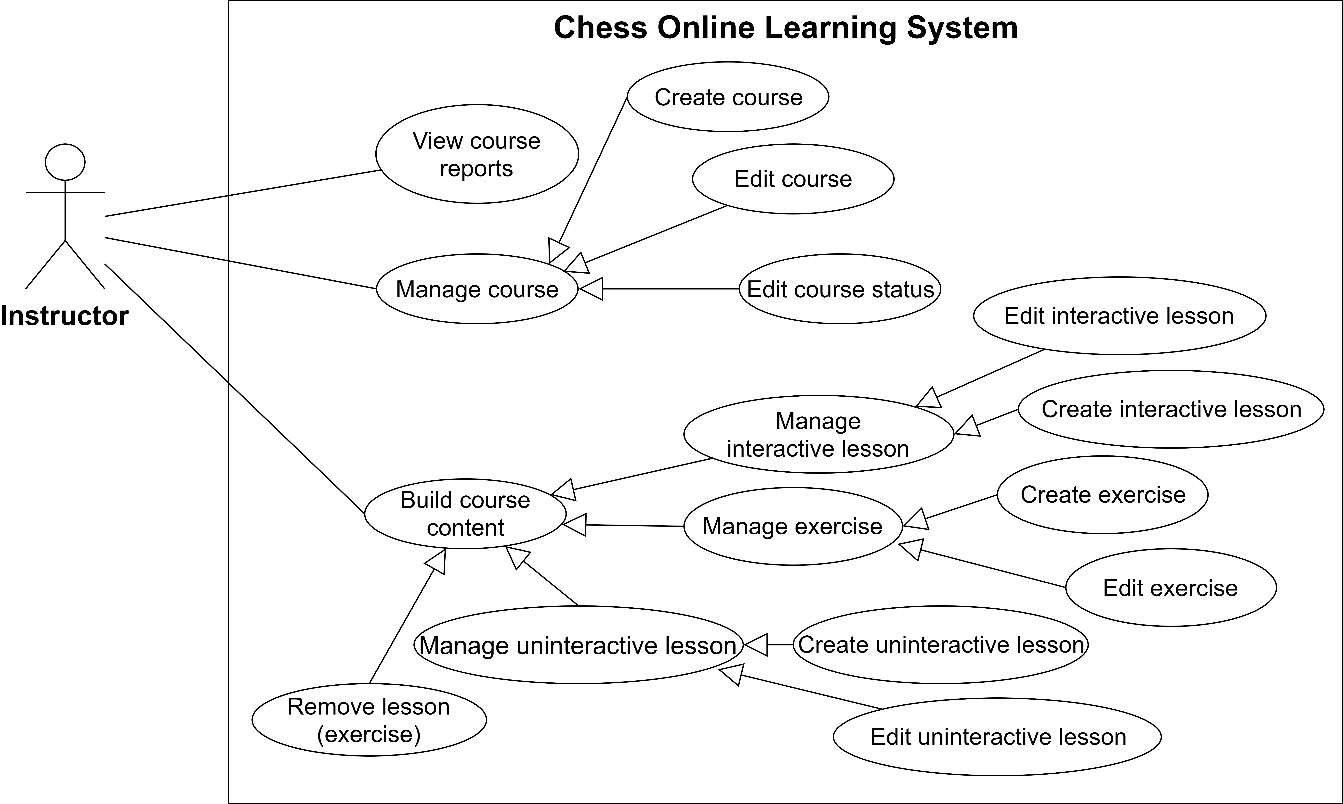


Figure : <Instructor> Overview Use Case

#### 5.2.3.1 <Instructor>  Create Course

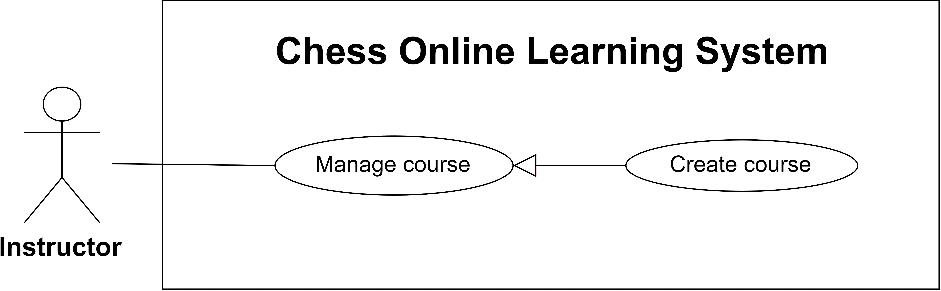


Figure : <Instructor> Create Course

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_IN.01** | | | |
| **Use Case No.** | UC\_IN.01 | **Use Case Version** | 2.0 |
| **Use Case Name** | Create Course | | |
| **Author** | Phạm Hoàng Tuyết Ngân | | |
| **Date** | 20/07/2019 | **Priority** | High |
| **Actor:**   * Instructor   **Summary:**   * This use case allows the instructor to create a course.   **Goal:**   * New course is created.   **Triggers:**   * Instructor sends the create course command.   **Preconditions:**   * User signed in as an instructor successfully.   **Postconditions:**   * Success: New course is created. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Instructor navigates to Create course page. | System goes to create a course page and shows the form:   * Tên khóa học: course name text field * Mô tả: course descrption text area * Yêu cầu: skill level selection, default value is “Mới bắt đầu” * Danh mục: category selection * Ảnh khóa học: course image file field, default is system image * Hoàn tất: submit button * Xóa toàn bộ: reset button   System required information:   * Tên khóa học: min length 6, max length 255 * Mô tả: min length 6 | | 2 | Instructor inputs information. | [Exception 1] | | 3 | Instructor clicks on “Hoàn tất” button.  [Alternative 1] | System shows the message “Tạo khóa học thành công” and create course.  System redirects to course details page with empty content. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | The instructor click “Xóa toàn bộ” button. | The create course form is reset. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **Step** | **Cause** | **System Response** | | 1 | Instructor inputs invalid information. | System shows error message match with fields:   * Empty name: “Tên khóa học không được bỏ trống” * Name is less than 6 characters: “Tên khóa học không được ít hơn 6 kí tự” * Name is more than 225 characters: “Tên khóa học không được nhiều hơn 255 kí tự” * Empty description: “Mô tả khóa học không được bỏ trống” * Description is less than 6 characters: “Mô tả khóa học không được ít hơn 6 kí tự” * Wrong image format: “Hình ảnh không đúng định dạng” |   **Relationships:** N/A  **Business Rules:**   * Instructor can create courses. * A new course is added in the database with status is Drafting and empty learning content. | | | |

Table : Create Course specification

#### 5.2.3.2 <Instructor>  Edit Course Status

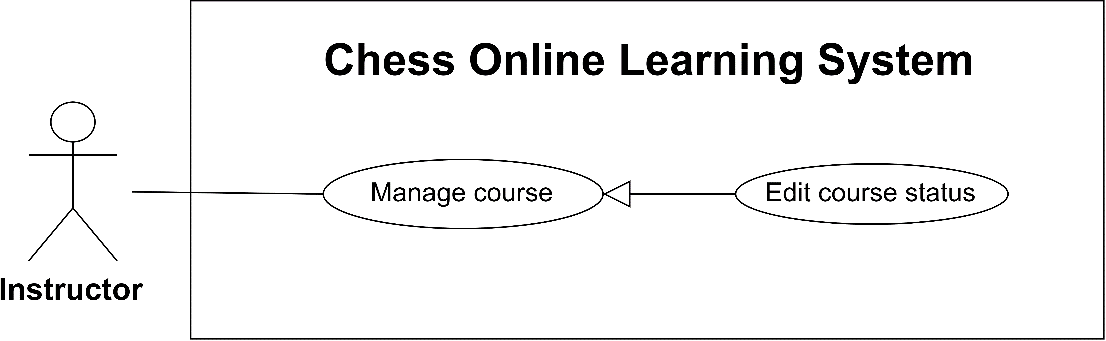


Figure : <Instructor> Edit Course Status

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_IN.03** | | | |
| **Use Case No.** | UC\_IN.03 | **Use Case Version** | 2.0 |
| **Use Case Name** | Edit Course Status | | |
| **Author** | Phạm Hoàng Tuyết Ngân | | |
| **Date** | 20/07/2019 | **Priority** | High |
| **Actor:**   * Instructor   **Summary:**   * This use case allows instructors to change the status of their courses.   **Goal:**   * Instructors can change their course status.   **Triggers:**   * Instructor sends the edit course status command.   **Preconditions:**   * User signed in as an instructor successfully. * The instructor must have at least 1 course.   **Postconditions:**   * Success: Course status is changed. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Instructor clicks on “Setting” button in a specific course. | System displays actions for instructor:   * Submit course: for drafting course * Unpublish course: for published course * Restore course: for rejected course * Resist course: for waiting course | | 2 | Instructor request to change course status. | System displays confirm dialogue with confirming message. | | 3 | Instuctor clicks on “Xác nhận”.  [Alternative 1] | Course status is changed and the system occurs to users. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Instructor clicks on “Đóng” button. | System closes the dialogue without changing. |   **Exceptions:** N/A  **Relationships:** N/A  **Business Rules:**   * Instructor can unpublish their Published courses. * Instructor can submit their Drafting courses if lesson list is not empty. * Instructor can restore their Rejected courses. * Instructor can resist their Waiting courses. * After action successfully, the course status is transformed. * System will send a notification to admin and enrolled learners if the course is unpublished. * System will send a notification to admin if the course is submitted. | | | |

Table : Edit Course Status specification

#### 5.2.3.3 <Instructor>  Create Interactive Lesson

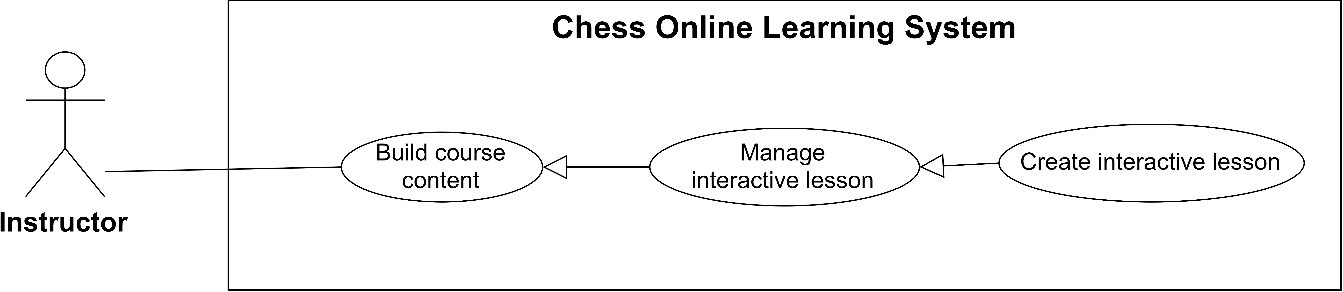


Figure : <Instructor> Create Interactive Lesson

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_IN.05** | | | |
| **Use Case No.** | UC\_IN.05 | **Use Case Version** | 2.0 |
| **Use Case Name** | Create Interactive Lesson | | |
| **Author** | Phạm Hoàng Tuyết Ngân | | |
| **Date** | 20/07/2019 | **Priority** | High |
| **Actor:**   * Instructor   **Summary:**   * This use case allows the instructor to create an interactive lesson.   **Goal:**   * New interactive lesson is created successfully.   **Triggers:**   * Instructor sends the create interactive lesson command.   **Preconditions:**   * User signed in as an instructor successfully. * Instructor must have at least 1 course.   **Postconditions:**   * Success: New lesson is created to course. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Instructor requests to create an interactive lesson. | System displays a screen to add interactive lessons:   * Tên bài học: text field * Mô tả: text area * Nội dung: text area * Nước đi: move content-box * Xóa toàn bộ: button * Chessboard * FEN Editing board button * Xem trước: button (disable, enable when information is validated) * Trở về: button (disable, enable when user the is in preview mode) * Close button   System required information:   * Tên bài học: min length 6, max length 255 * Mô tả: min length 6 * The move list is not empty | | 2 | Instructor input lesson name and description. | [Exception 1] | | 3 | Instructor creates lesson content by performing move on the chessboard. [Alternative 1] | Move is auto-added in move content box. | | 4 | Instructor can add content for each move in “Nội dung” field. | Content is added for the move. | | 5 | Instructor clicks “Xem trước” button  [Alternative 2] [Alternative 3] | System displays lesson content as live mode:   * Lưu: button * Trở về: button | | 6 | Instructor clicks “Lưu” button.  [Alternative 2] | System displays a message “Tạo trận đấu thành công” and create new interactive lesson. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1a | Instructor clicks FEN editing board button. | System shows form “Tạo thế cờ”   * Chessboard * Lưu thế cờ: button * Đóng: button. | | 1b | Instructor drags chess pieces to the chessboard and clicks the save button. | System shows a new chess position on the chessboard and all existing moves is removed. | | 2 | Instructor clicks the close button. | System closes create lesson form and return to previous page. | | 3 | Instructor clicks the remove all button. | System removes all the information were set up. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **Step** | **Cause** | **System Response** | | 1 | Instructor inputs invalid information in the text fields. | System shows error message match with fields:   * Empty name: “Tên bài học không được bỏ trống” * Name is less than 6 characters: “Tên bài học không được ít hơn 6 kí tự” * Name is more than 225 characters: “Tên bài học không được nhiều hơn 255 kí tự” * Empty description: “Mô tả bài học không được bỏ trống” * Description is less than 6 characters: “Mô tả bài học không được ít hơn 6 kí tự” |   **Relationships:** N/A  **Business Rules:**   * Instructor creates a new interactive lesson to build course content. * Interaction lesson consists of many different chess moves and moves analyse, the instructor can freely set the chessboard. * A course can have many interactive lessons. | | | |

Table : Create Interactive Lesson specification

#### 5.2.3.4 <Instructor>  Create Uninteractive Lesson

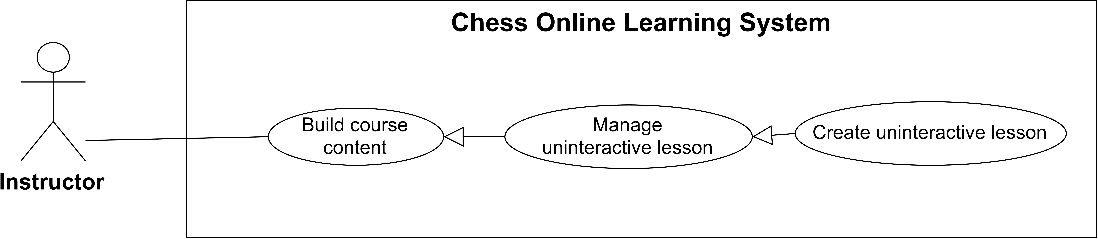


Figure : <Instructor> Create Uninteractive Lesson

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_IN.07** | | | |
| **Use Case No.** | UC\_IN.07 | **Use Case Version** | 2.0 |
| **Use Case Name** | Create Uninteractive Lesson | | |
| **Author** | Đặng Văn Hoàng | | |
| **Date** | 20/07/2019 | **Priority** | High |
| **Actor:**   * Instructor   **Summary:**   * This use case allows the instructor to create an uninteractive lesson.   **Goal:**   * Instructor can create an uinteractive lesson for the course.   **Triggers:**   * Instructor sends the create uninteractive lesson commands.   **Preconditions:**   * User signed in as an instructor successfully. * The instructor must have at least 1 course.   **Postconditions:**   * Success: An uninteractive lesson is added to the course. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Instructor navigates to creates an unteractive lesson | System displays a form to create the uninteractive lesson:   * Tên bài học: text field * Mô tả: text area * Nội dung bài học: WYSIWG Editor * Thêm: button * Close button   System required information:   * Tên bài học: min length 6, max length 255 * Mô tả: min length 6 * Nội dung: min length 6 | | 2 | Instructor inputs information. | [Exception 1] | | 3 | Instructor clicks on “Thêm” button.  [Alternative 1] | System create uninteractive lessons and displays message “Tạo bài đọc học thành công”. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Instructor clicks on the close icon. | System closes the form without  changing. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **Step** | **Cause** | **System Response** | | 1 | Instructor inputs invalid information or not fill in WYSIWYG Editor. | System shows error message match with fields:   * Empty name: “Tên bài học không được bỏ trống” * Name is less than 6 characters: “Tên bài học không được ít hơn 6 kí tự” * Name is more than 225 characters: “Tên bài học không được nhiều hơn 255 kí tự” * Empty description: “Mô tả bài học không được bỏ trống” * Description is less than 6 characters: “Mô tả bài học không được ít hơn 6 kí tự” * Empty content: “Nội dung bài học không được bỏ trống” * Content is less than 6 characters: “Nội dung bài học không được ít hơn 6 kí tự” |   **Relationships:** N/A  **Business Rules:**   * Instructor can create uninteractive lessons to build course content. * Uninteractive lesson is a theoretical learning format including images, videos, document files. * A course can have many uninteractive lessons. | | | |

Table : Create Uniteractive Lesson specification

#### 5.2.3.5 <Instructor>  Create Exercise

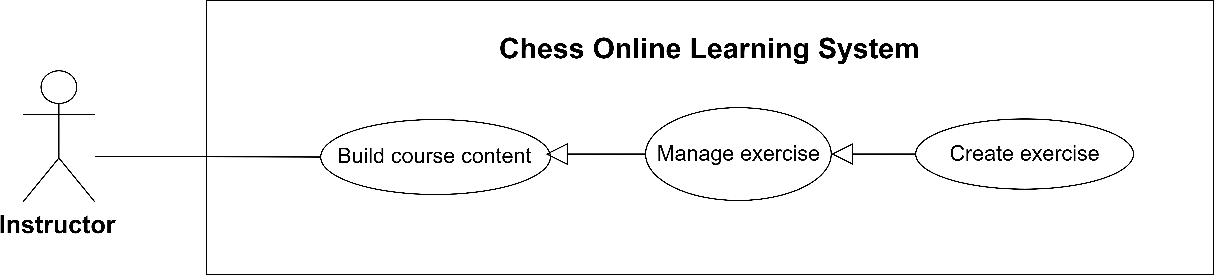


Figure : <Instructor> Create Exercise

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_IN.09** | | | |
| **Use Case No.** | UC\_IN.09 | **Use Case Version** | 2.0 |
| **Use Case Name** | Create Exercise | | |
| **Author** | Phạm Hoàng Tuyết Ngân | | |
| **Date** | 20/07/2019 | **Priority** | High |
| **Actor:**   * Instructor   **Summary:**   * This use case allows the instructor to create an exercise.   **Goal:**   * Instructor can create a new exercise.   **Triggers:**   * Instructor sends the create exercise command.   **Preconditions:**   * User signed in as an instructor successfully. * The instructor must have at least 1 course.   **Postconditions:**   * Success: An exercise is added to the course. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Instructor navigates to create a new exercise. | System displays a form to create an exercise:   * Tên bài tập: text field * Câu hỏi: text field * Mô tả: text area * Three of the answer:   + - Nước đi: move content-box     - Thực hiện đúng: text field     - Thực hiện sai: text field * Xóa toàn bộ: button * Chessboard * FEN Editing board button * Xem trước: button (disable/enable when information is valid) * Trở về: button (disable/enable when the user is in preview mode) * Close button * Dùng chế độ đánh tự động: checkbox   System required information:   * Tên bài tập: min length 6, max length 255 * Câu hỏi: min length 6 * Mô tả: min length 6 * Answer List is not empty or enable chess bot mode | | 2 | Instructor inputs exercise name, question and description. | [Exception 1] | | 3 | Instructor adds lesson content by performing a move.  [Alternative 1] [Alternative 2] | Move is auto-added to move content-box. | | 4 | Instructor can fill “Thực hiện đúng” and “Thực hiện sai” content for each move. | System saves the right response and wrong response for the move. | | 5 | Instructor clicks “Xem trước” button.  [Alternative 3] [Alternative 4] | System shows the new exercise in live mode:   * Lưu: button * Trở về: button. | | 6 | Instructor clicks “Lưu” button.  [Alternative 3] | System creates a new exercise and shows a message “Thêm bài tập thành công”. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1a | Instructor clicks FEN editing board button. | System shows form “Tạo thế cờ”   * Chessboard * Lưu thế cờ: button   Đóng: button. | | 1b | Instructor drags chess pieces to the chessboard and clicks the save button. | System shows a new chess position on the chessboard and all existing moves is removed. | | 2 | Instructor chooses bot answer. | Move answer form is disabled and the answer is noted in a bot answer. | | 3 | Instructor clicks on the close button. | System closes the for without changing. | | 4 | Instructor clicks on the remove all button. | System removes all the instructor information which were set up. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **Step** | **Cause** | **System Response** | | 1 | Instructor inputs invalid information in the text fields. | System shows error message match with fields:   * Empty name: “Tên bài tập không được bỏ trống” * Name is less than 6 characters: “Tên bài tập không được ít hơn 6 kí tự” * Name is more than 225 characters: “Tên bài tập không được nhiều hơn 255 kí tự” * Empty question: “Câu hỏi không được bỏ trống” * Question is less than 6 characters: “Câu hỏi bài học không được ít hơn 6 kí tự” * Empty is description: “Mô tả bài tập không được bỏ trống” * Description is less than 6 characters: “Mô tả bài tập không được ít hơn 6 kí tự” |   **Relationships:** N/A  **Business Rules:**   * Instructor can create exercises to build course content. * A course can have many exercises. * Exercise is a lesson, which provides a chess puzzle and learner must perform the right move to solve the problem. * Exercise contains 2 types of answer: using Bot answer and Manual answer. * There are as much as possible 3 answers in Manual answer. * Manual answer support right response and wrong response for each move. | | | |

Table : <Instructor> Create Exercise specification

### **5.2.4 <Admin> Overview Use Case**

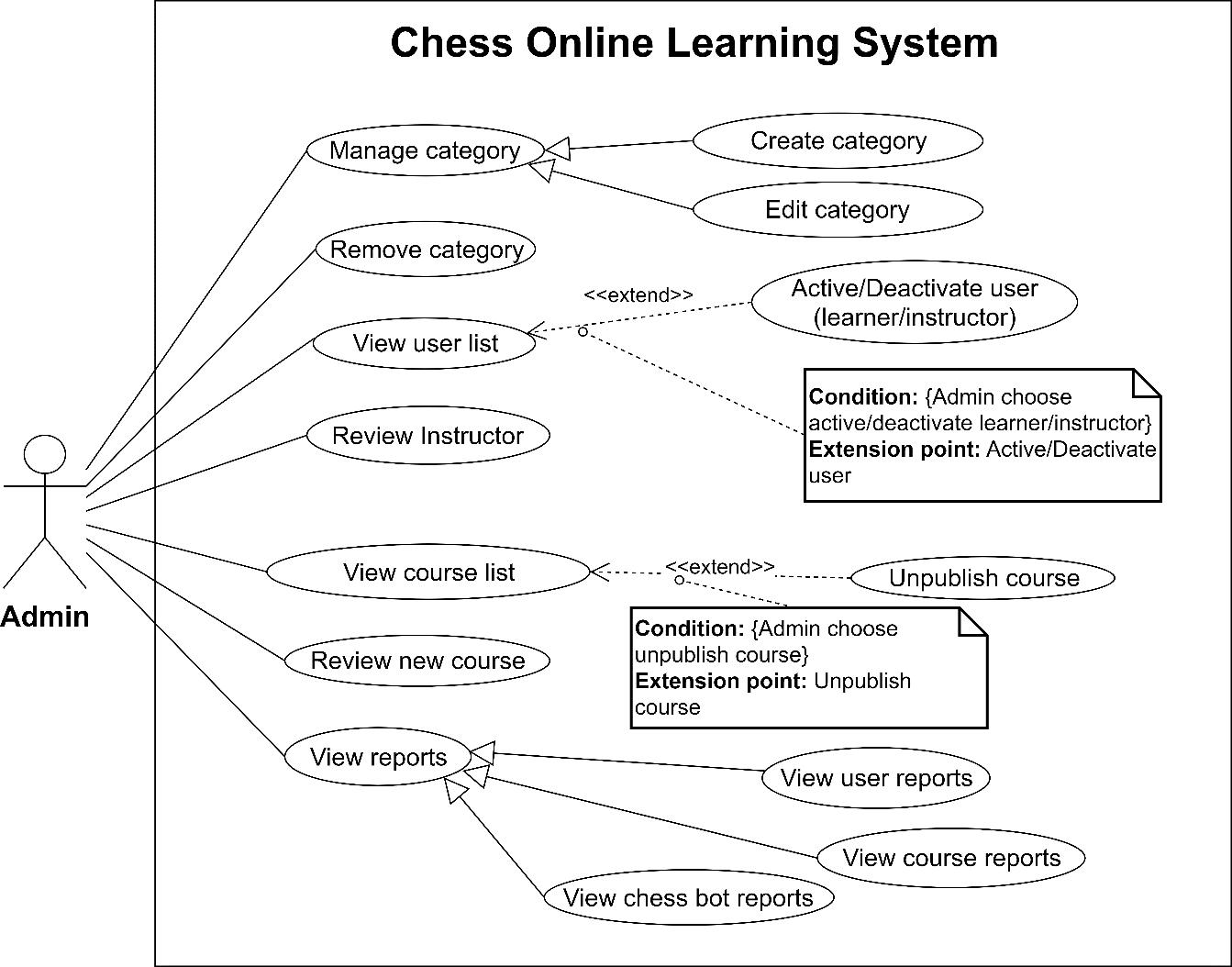


Figure : <Admin> Overview Use Case

#### 5.2.4.1 <Admin> Create Category

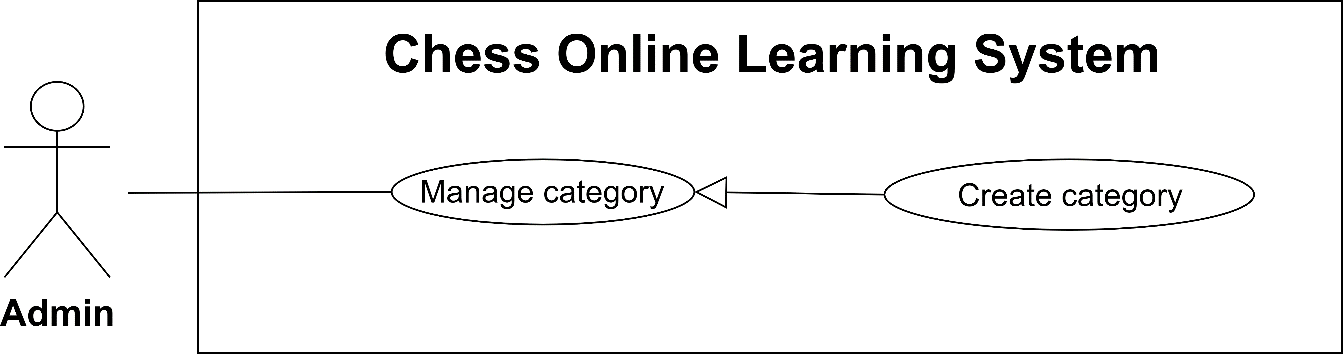


Figure : <Admin> Create Category

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_AD.01** | | | |
| **Use Case No.** | UC\_AD.01 | **Use Case Version** | 2.0 |
| **Use Case Name** | Create Category | | |
| **Author** | Đặng Văn Hoàng | | |
| **Date** | 20/7/2019 | **Priority** | Normal |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to create a category.   **Goal:**   * A new category is added to system.   **Triggers:**   * Admin sends the create category command.   **Preconditions:**   * User signed in as an admin successfully.   **Postconditions:**   * Success: A category is added to the system successfully. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin requests to create a new category. | System shows create category form:   * Danh mục: text field * Đồng ý: button * Hủy: button   System required information:   * Danh mục: min length 6, max length 255 | | 2 | Admin inputs text field. | [Exception 1] | | 3 | Admin clicks on “Đồng ý” button  [Alternative 1] | System shows the message “Tạo danh mục thành công” and a new category is added. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks on “Hủy” button | System returns to the category management page without changing. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Admin inputs invalid information. | System shows error message match with fields:   * Empty name: “Tên danh mục không được bỏ trống” * Name is less than 6 characters: “Tên danh mục không được ít hơn 6 kí tự” * Name is more than 225 characters: “Tên danh mục không được nhiều hơn 225 kí tự” |   **Relationships:** N/A  **Business Rules:**   * System category can be created by admins. | | | |

Table : Create Category specification

#### 5.2.4.2 <Admin> Review New Course

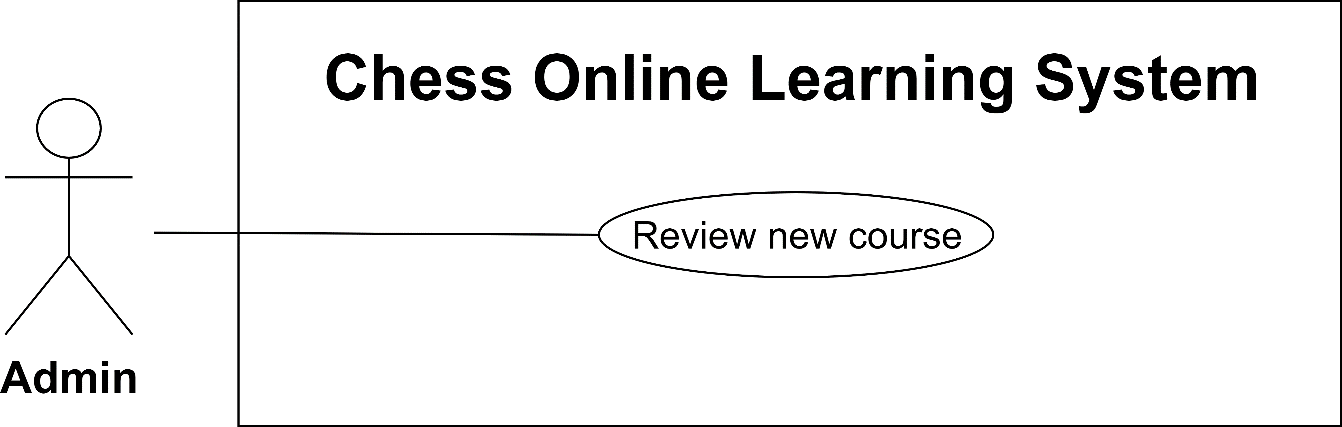


Figure : <Admin> Review New Course

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_AD.07** | | | |
| **Use Case No.** | UC\_AD.07 | **Use Case Version** | 2.0 |
| **Use Case Name** | Review New Course | | |
| **Author** | Đặng Văn Hoàng | | |
| **Date** | 20/7/2019 | **Priority** | Normal |
| **Actor:**   * Admin   **Summary:**   * This use case allows admin review new course. See details of the course: course information and course content.   **Goal:**   * A course is reviewed before publishing.   **Triggers:**   * Admin sends the review new course command.   **Preconditions:**   * User signed in as an admin successfully.   **Postconditions:**   * Success: A new course is published or rejected. * Failed: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin navigates to waiting course. | System displays the course details:   * Course information: name, description, skill level required, categories. * Course content: the number of lessons, lesson list. * Setting button | | 2 | Admin clicks on the setting button  [Alternative 1] | System show dialog “Thay đổi trạng thái khóa học”:   * Đồng ý: button * Từ chối: button * Đóng: button | | 3 | Admin click on “Đồng ý” button  [Alternative 1] [Alternative 2] | System displays a message “Khóa học đã được công khai” and change status course. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks on “Đóng” button | System closes dialog without changing in course. | | 2a | Admin clicks on “Từ chối” button | System displays dialog to enter rejected reason, after Admin confirm reason. | | 2b | Admin input rejected reson. | Course status is transformed to “Rejected” and system displays a message “Khóa học đã bị từ chối công khai”. |   **Exceptions:** N/A  **Relationships:** N/A  **Business Rules:**   * When the admin review is completed, the system sends a notifications and confirm email to the instructors. | | | |

Table : Review New Course specification

#### 5.2.4.3 <Admin> Review Instructor

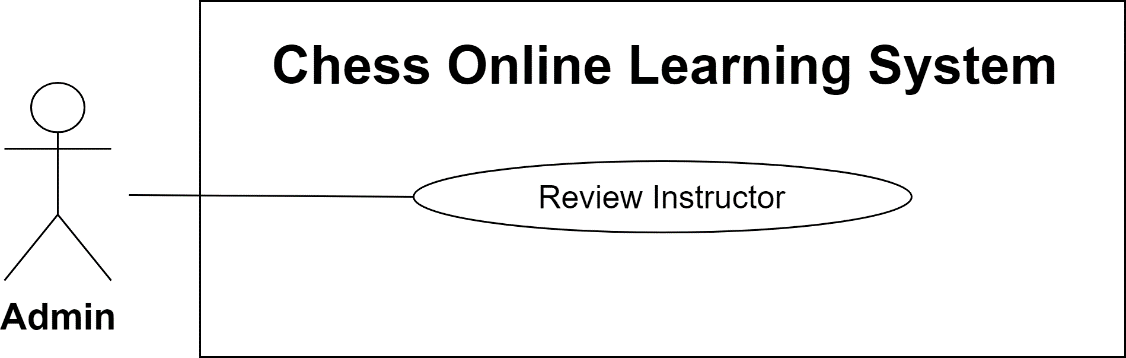


Figure : <Admin> Review Instructor

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_AD.10** | | | |
| **Use Case No.** | UC\_AD.10 | **Use Case Version** | 2.0 |
| **Use Case Name** | Review Instructor | | |
| **Author** | Đặng Văn Hoàng | | |
| **Date** | 20/7/2019 | **Priority** | Normal |
| **Actor:**   * Admin   **Summary:**   * This use case allows admin view the list of registered users to become instructors, view personal information of new instructor and approve or reject them.   **Goal:**   * Limit instructor for system.   **Triggers:**   * Admin sends the instructor review command.   **Preconditions:**   * User signed in as an Admin successfully. * New instructor apply to system.   **Post conditions:**   * Success: A new instructor is approved or rejected. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin navigates to view new instructor. | System shows new instructor information to the form and review buttons:   * Information fom:   + Full Name   + Email   + Achivement   + Certificate * Đồng ý: button * Từ chối: button | | 2 | Admin clicks “Đồng ý” button  [Alternative 1] | System shows a message “Người hướng dẫn mới đã được chấp nhận” and changes status of instructor. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1a | Admin click on “Từ chối” button | System shows rejected reason dialogue. | | 1b | Admin inputs rejected reason and confirm with the system. | System removes user and sends rejected email to them. |   **Exceptions:** N/A  **Relationships:** N/A  **Business Rules:**   * Admin can review all register. * When admin accepts, the user account can access the instructor's page. An accept notification is sent to the instructor. * When admin rejects, the instructor is removed in the database and they can apply again. | | | |

Table : Review Instructor specification

# 6. System Architecture Design2

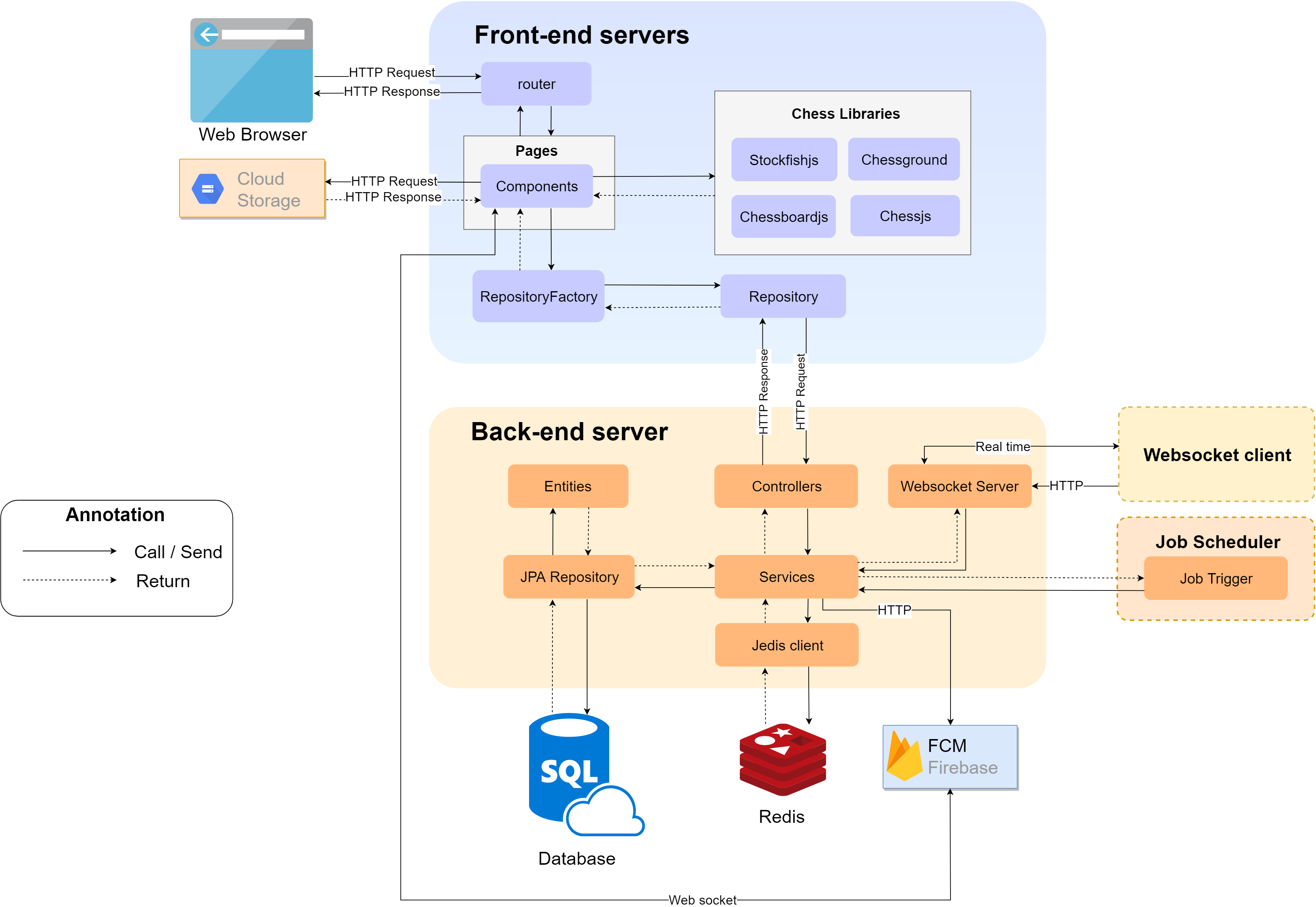


Figure : System architecture overview

This system is developed base on multi-tiers architecture with Database, Back-end server for process business logic and 2 front-end servers (for learning-app and admin-app) to process presentation.

**a. Data tier**

In Data tier, we use Relational Database Management System - MySQL to store system data.

**b. Business Logic tier**

In Business Logic tier, we use Spring Boot to develop RESTful APIs with MVC pattern.

In this system, the chess engine performs in the presentation tier, then we using websocket to synchronize game states between server and client.

We also use Redis, an in-memory data structure for caching data.

In order to schedule automatic tasks such as giving course suggestions to learners. We use Quartz Job 2.0 to schedule the task to run repeatedly. Besides, it is also used to trigger the timeout of the chess game by the cron expression.

**c. Presentation tier**

For Front-end applications, we use VueJS framework and Repository-pattern to manage data. We also apply chess libraries to solve our problems, which be related to user interaction, include:

* **Stockfishjs:** Stockfish is an opensource chess engine, which supports to analyse chess move and Stockfishjs is a pure JavaScript implementation of Stockfish. We can use UCI (Universal Chess Interface) to communicate with the chess engine. The chess engine performs as a worker thread, they receive UCI command by postMessage() method and perform tasks to get a result, and return the result by onmessage() method. We apply stockfishjs to build chess bot, which can autoplay with learner and support learner in exercises.
* **Chessjs:** Chessjs is a chess library which use to generate chess move and validate the client move. We use this library to validate user move and manage current status of chess game, such as turn colour, pieces position.
* **Chessboardjs and Chessground:** There 2 library also support to render Graphic User Interface (GUI), 2 libraries which be discussed above only support data in console, then we must use this libraries to display chessboard for the user. They can perform move and display pieces position from Chessjs and Stockfishjs by using FEN String, which will be discussed below.

**d. Cloud Services**

In this system, we use cloud services, which are Cloud Storage and Firebase Cloud Messaging (FCM). We use Cloud Storage of Google Cloud Platform to store images which are uploaded by the client and store the URL in Database. The FCM is used to push real-time notifications easily and conveniently.

# 7. Component Diagram1

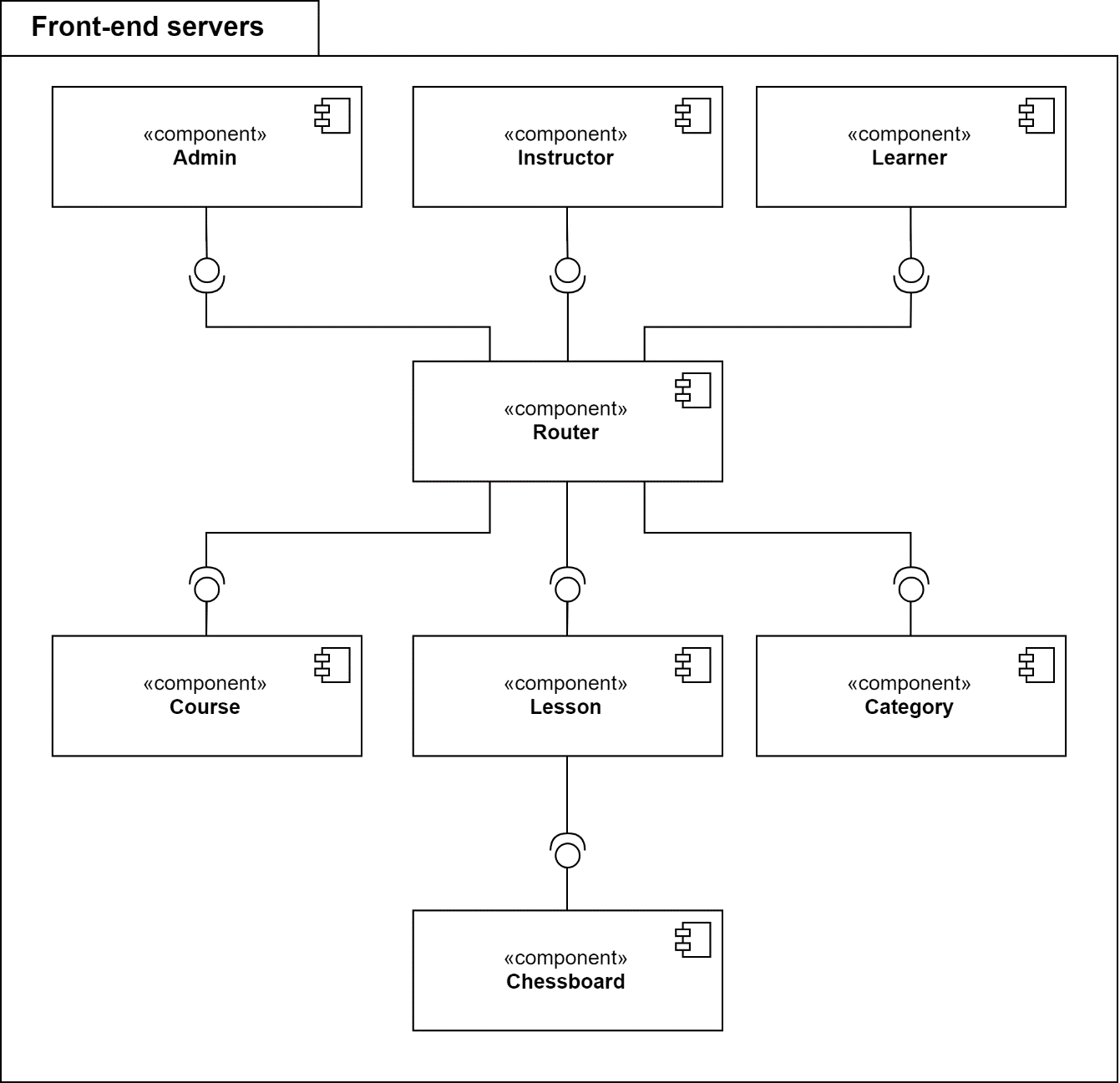


Figure : Front-end Servers Component diagram

|  |  |
| --- | --- |
| COMPONENT DICTIONARY: DESCRIBES COMPONENTS | |
| Router | Handle HTTP request, response |
| Course Component | Handle activity about Course |
| Lesson Component | Handle activity about Lesson |
| Chessboard Component | Handle activity about Chessboard |
| Category Component | Handle activity about Category |
| Admin Component | Handle activity of Admin |
| Instructor Component | Handle activity of Instructor |
| Learner Component | Handle activity of Learner |

Table : Front-end Servers Component Dictionary

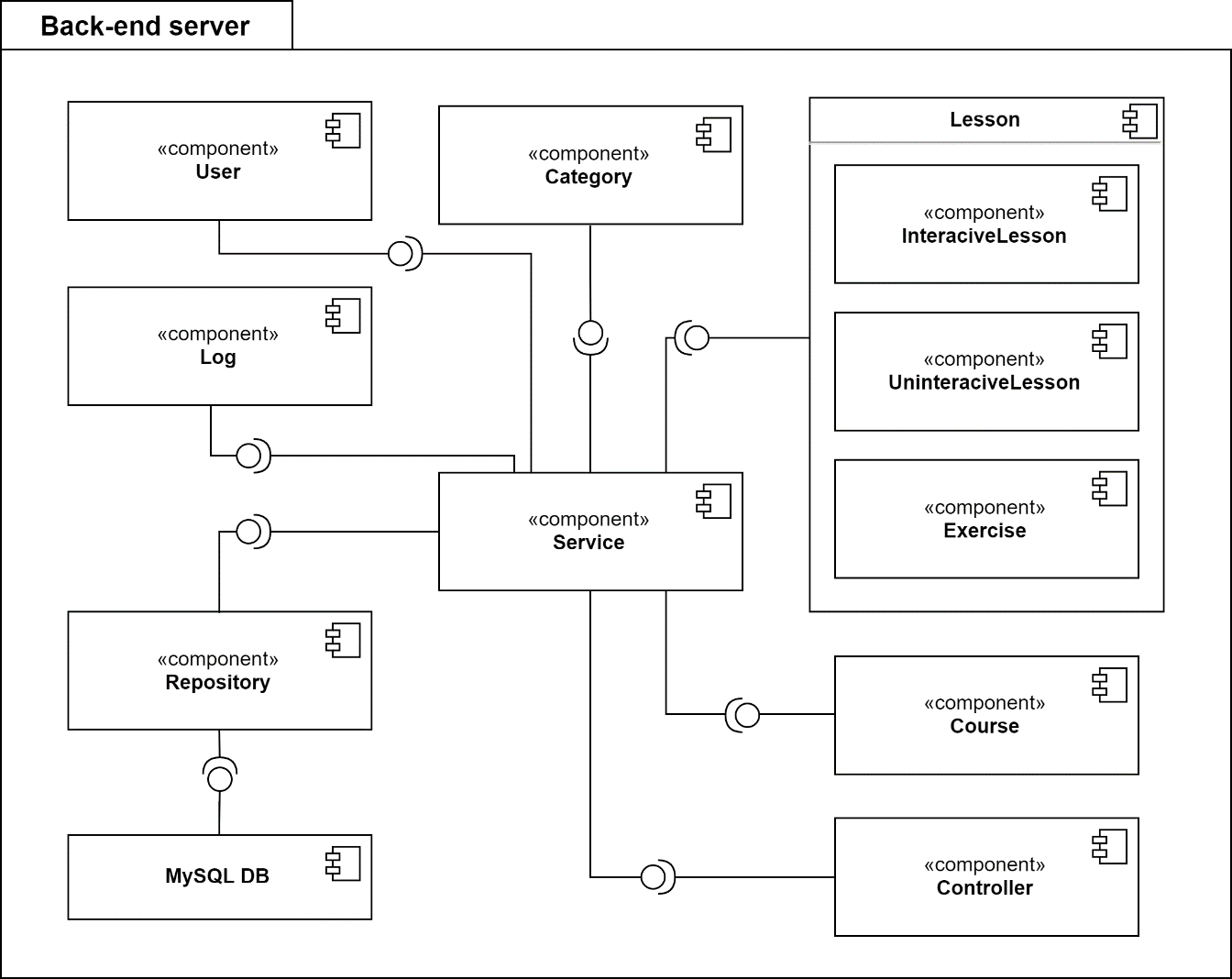


Figure : Back-end Server Component diagram

|  |  |
| --- | --- |
| COMPONENT DICTIONARY: DESCRIBES COMPONENTS | |
| Constroller | Handle HTTP request, response |
| Service Component | Process data |
| Repository Component | Access data from database |
| MySQL DB | Database component |
| Category Component | Handle activity about Category |
| User Component | Handle activity of User |
| Log Component | Handle log about system activity |
| Course Component | Handle activity about Course |
| Lesson Component | Handle activity about Lesson |
| Interactive Lesson Component | Handle activity about Lesson (type Interactive) |
| Uninteractive Lesson Component | Handle activity about Lesson (type Uninteractive) |
| Exercise Component | Handle activity about Exercise |

Table : Back-end Servers Component Dictionary

# 8. Class Diagram1

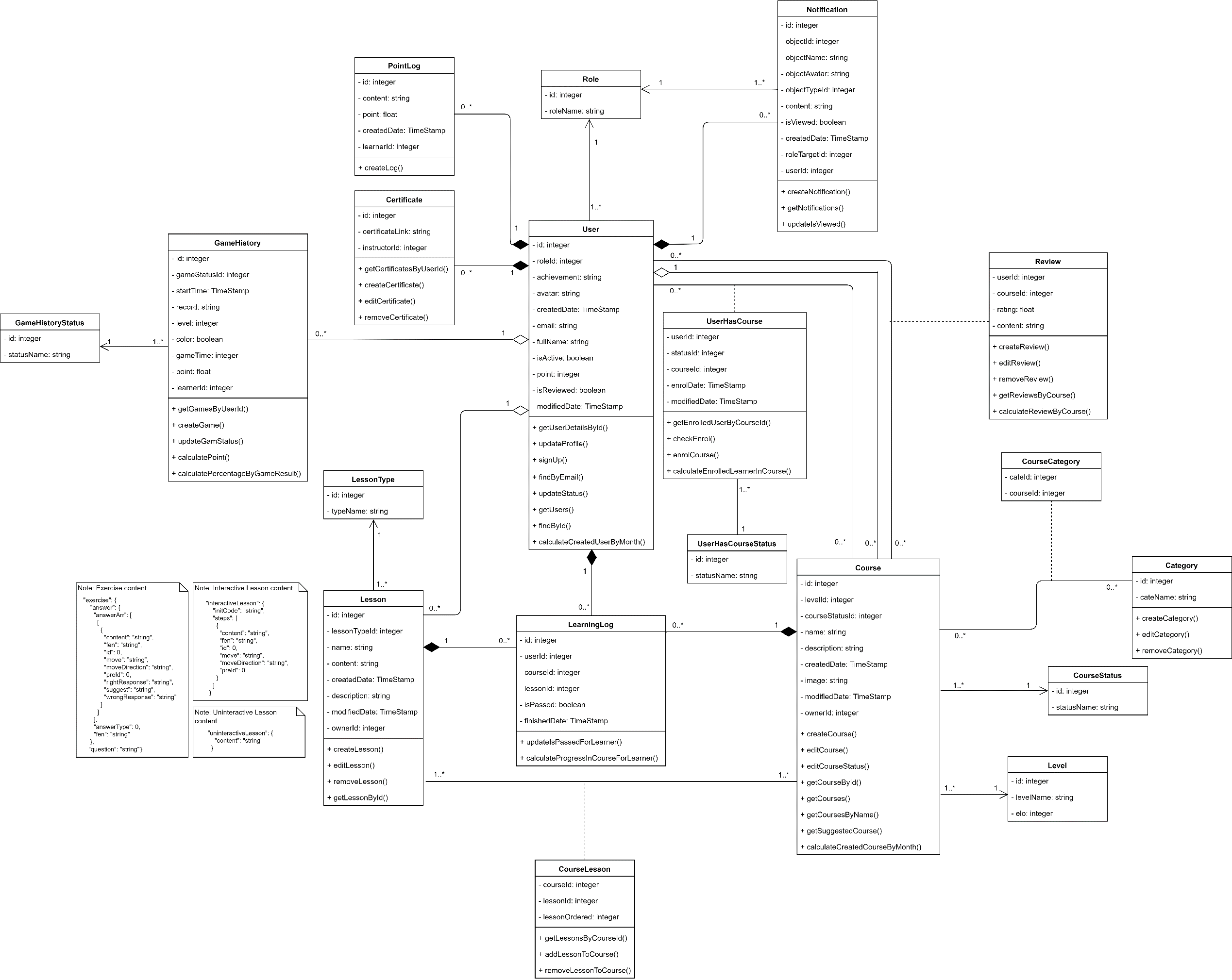


Figure : Class Diagram

|  |  |  |
| --- | --- | --- |
| **CLASS DICTIONARY: DESCRIBE CLASS** | | |
| **Class Name** | **Mapping column with Conceptual diagram** | **Description** |
| **User** | User | Contains the user information |
| **Role** | N/A | Not exist in Conceptual diagram but need this class to contain role name for user |
| **Notification** | Notification | Contains the notification for user |
| **Certificate** | Certificate | Contains the certificate information for instructor |
| **PointLog** | N/A | Not exist in Conceptual diagram but need this class to log point transaction for learner |
| **GameHistory** | GameHistory | Contain the game history information for learner |
| **GameHistoryStatus** | N/A | Not exist in Conceptual diagram but need this class to contain the status name of game history |
| **Course** | Course | Contains the course information |
| **CourseStatus** | N/A | Not exist in Conceptual diagram but need this class to contain status name of course |
| **CourseLesson** | N/A | Not exist in Conceptual diagram but need this class to contain the course which lesson belongs to |
| **Review** | Review | Contains the review for each course by learner |
| **UserHasCourse** | N/A | Not exist in Conceptual diagram but need this class to contain status for each course of learner |
| **UserHasCourseStatus** | N/A | Not exist in Conceptual diagram but need this class to contain status name for each course of learner |
| **Category** | Category | Contains the category information |
| **CourseCategory** | N/A | Not exist in Conceptual diagram but need this class to contain the relationship between courses and categories |
| **Lesson** | Lesson | Contains the lesson information |
| **LessonType** | N/A | Not exist in Conceptual diagram but need this class to contain the type name for each lesson |
| **LearningLog** | LearningLog | Contain learning logs of learners |

Table : Class Dictionary

# 9. Entity Relationship Diagram3

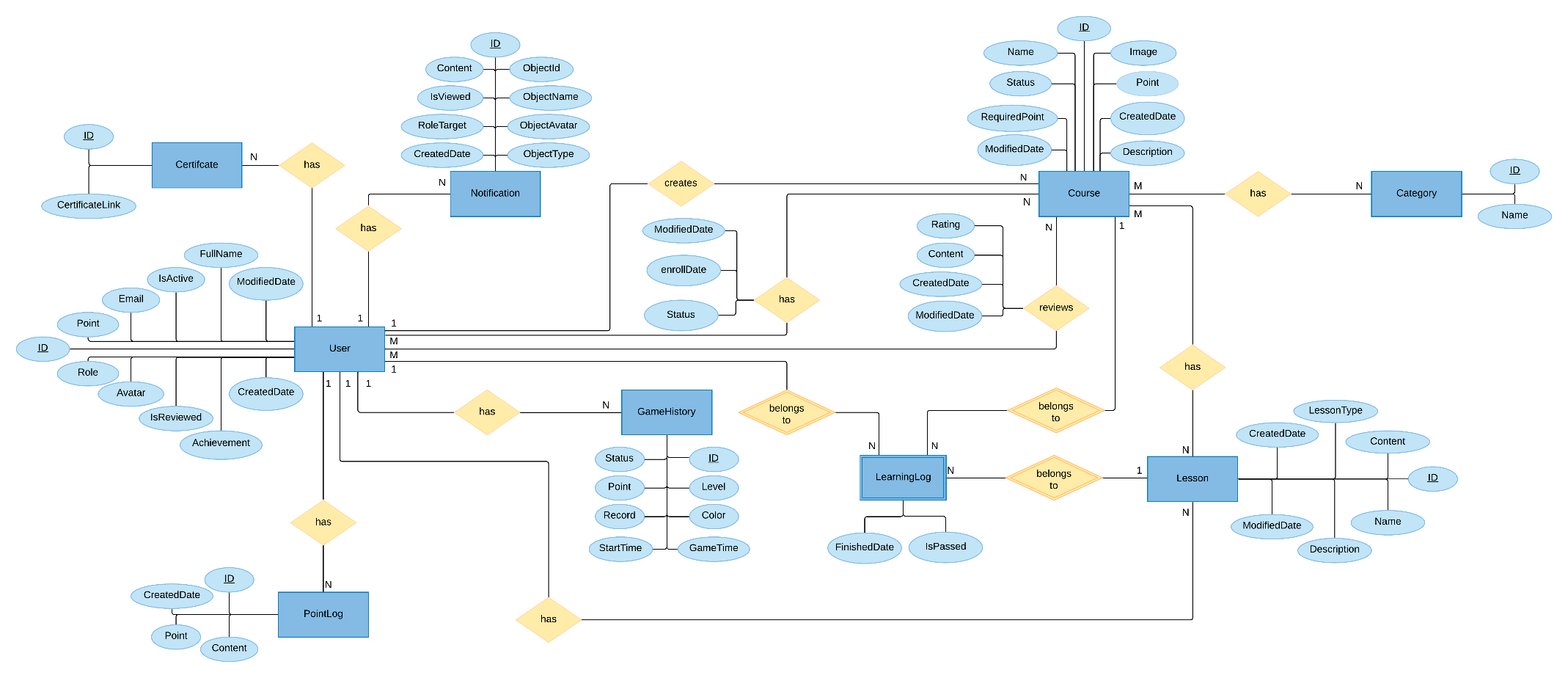


Figure : Entity relationship diagram (ERD)

|  |  |
| --- | --- |
| Entity Data Dictionary: describe content of all entities | |
| Entity name | **Description** |
| User | Contains the user’s information |
| Certificate | Contains certificates’s link for Instructor |
| Notification | Contains User’s notification |
| PointLog | Contains log about point transaction of Learner. |
| GameHistory | Contains the game history’s information. |
| Course | Contains the course’s information |
| Category | Contains the category’s information |
| Lesson | Contains the lesson’s information |
| LearningLog | Contains learning log for Learner |

Table : Entity data dictionary

**10. Interactive Diagram1**

## 10.1 Learner

### **10.1.1 Enrol Course**

Summary: this diagram shows the process to enrol a course

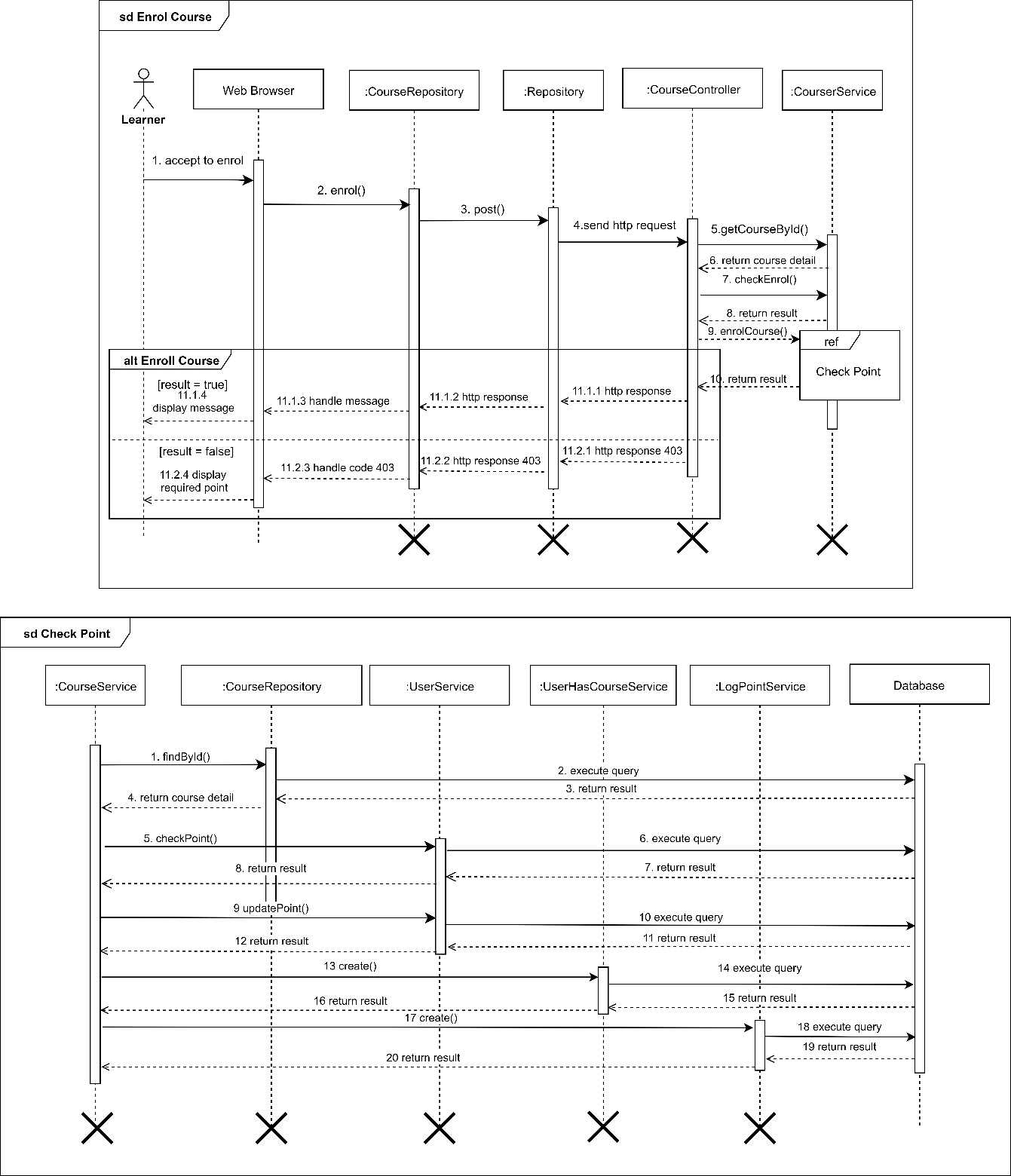


Figure 32: Sequence Diagram - <Learner> Enrol Course

### **10.1.2 Create Review**

Summary: this diagram shows process of create a review

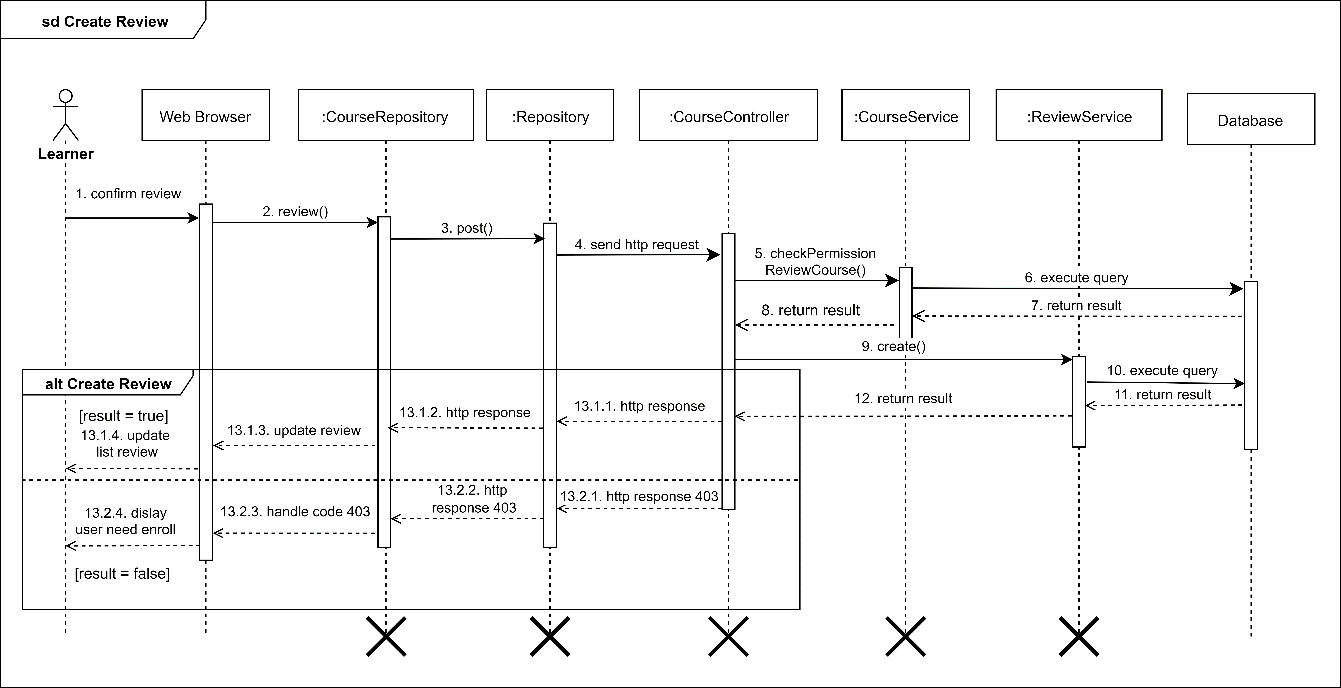


Figure 33: Sequence Diagram - <Learner> Create Review

### **10.1.3 View List Of Suggested Course**

Summary: this diagram shows process of view list of suggested course

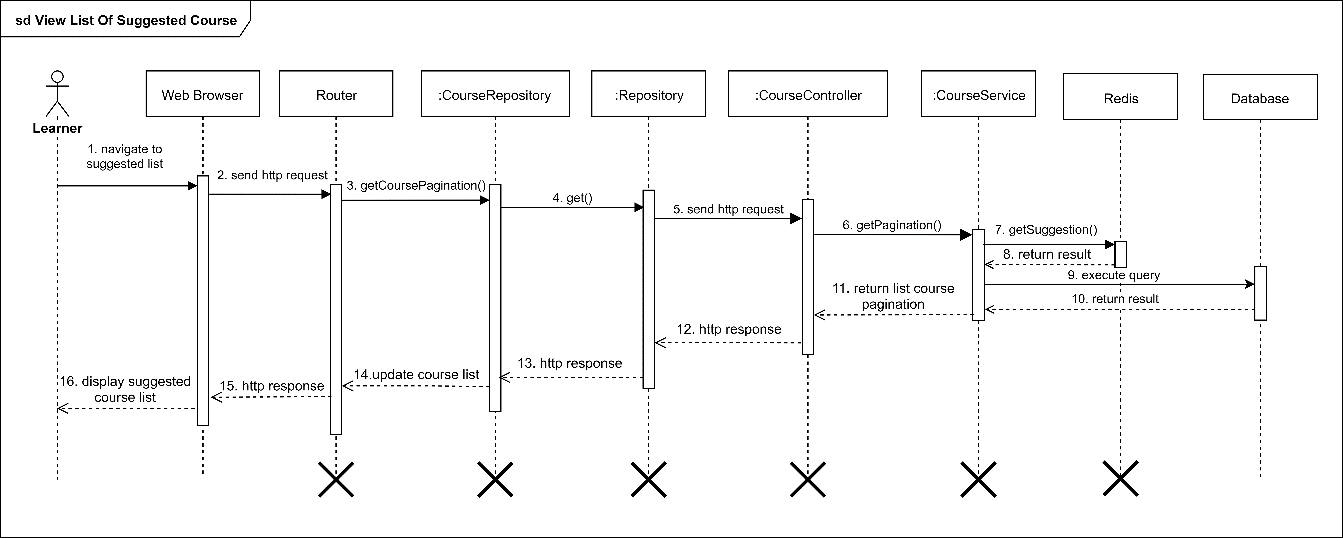


Figure 34: Sequence Diagram - <Learner> View List Of Suggested Course

## 10.2 Instructor

### **10.2.1 Create Course**

Summary: this diagram shows process of create a course

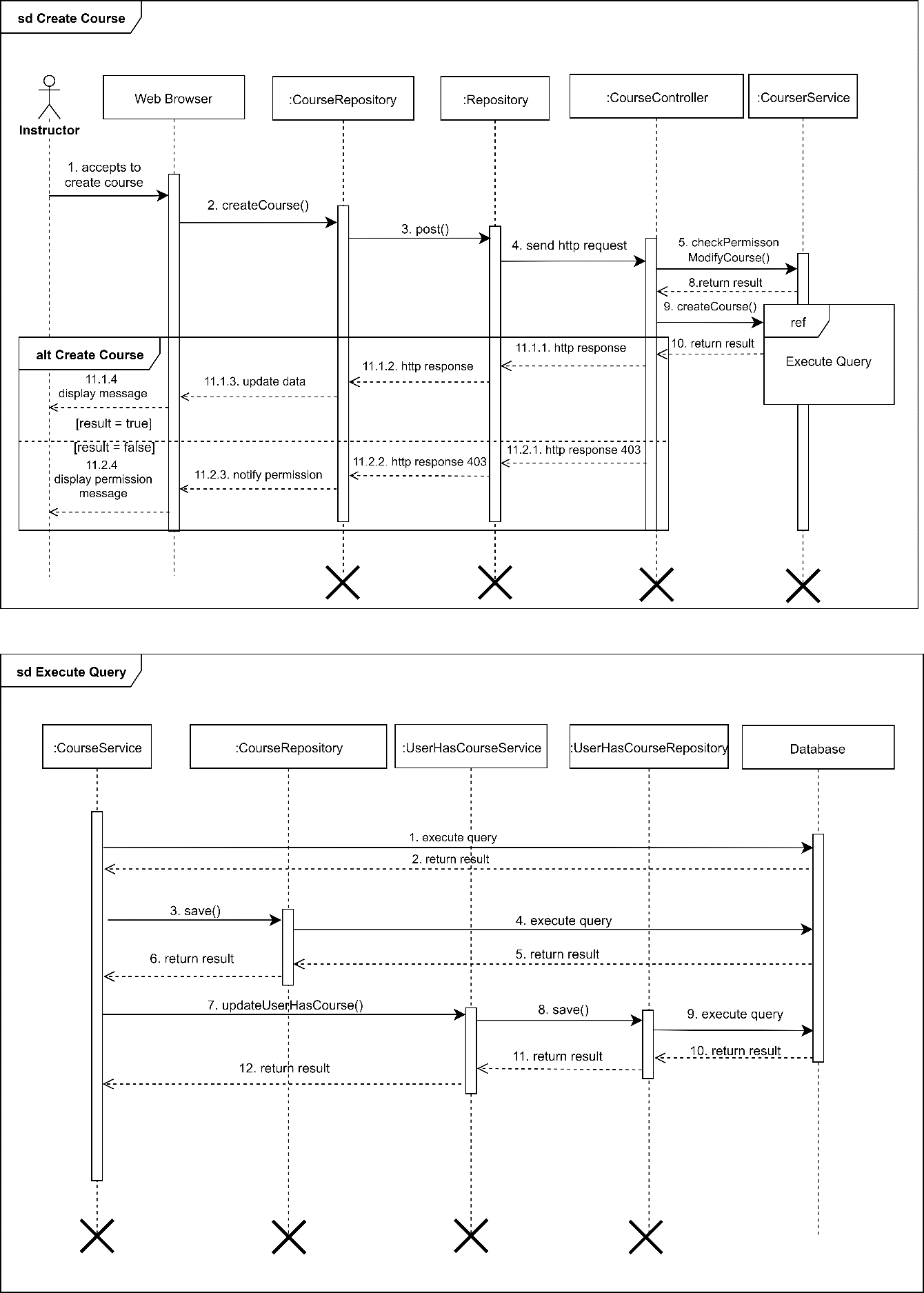


Figure 35: Sequence Diagram - <Instructor> Create Course

### **10.2.2 Submit Course**

Summary: this diagram shows process of submit course

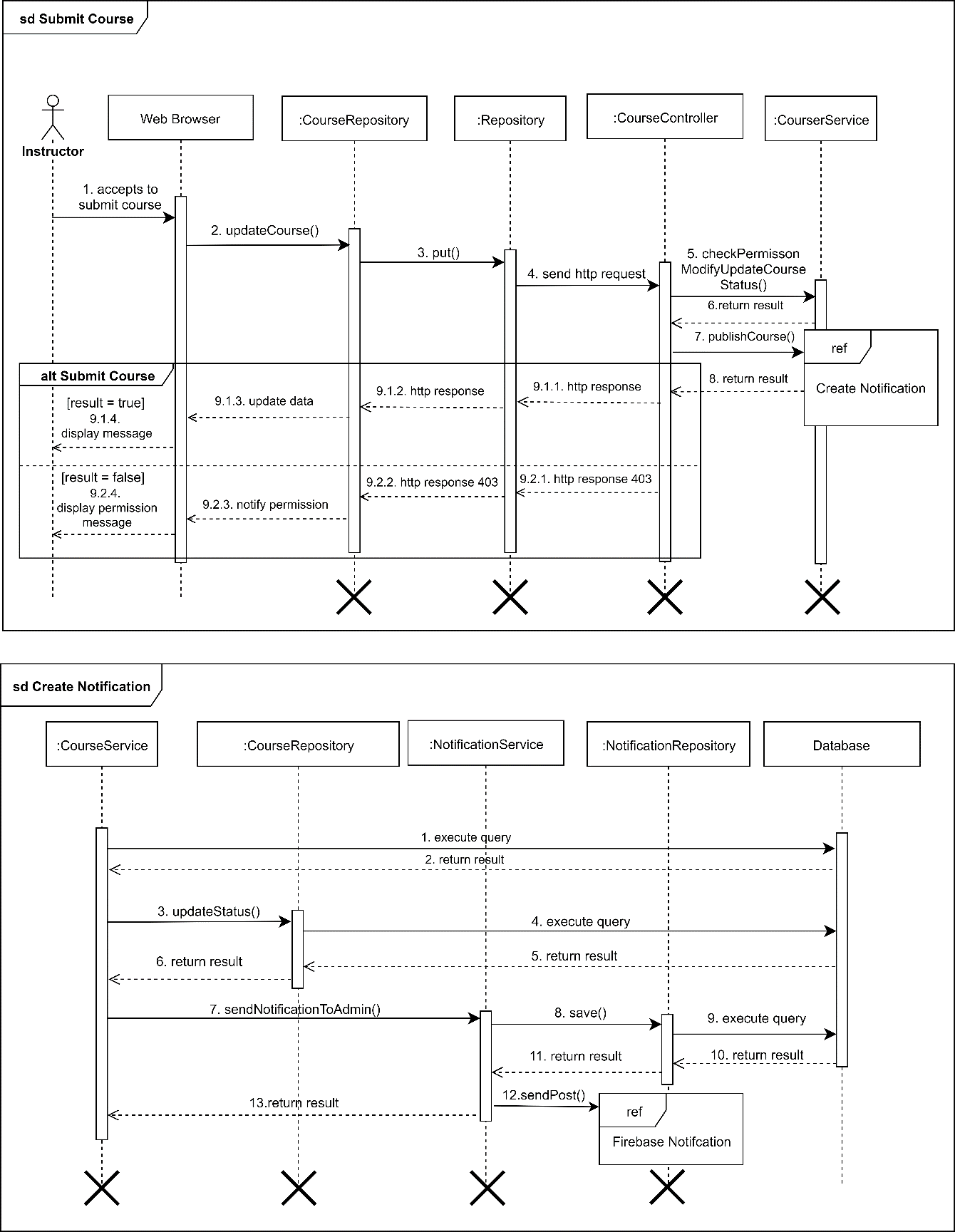


Figure 36: Sequence Diagram - <Instructor> Submit Course

### **10.2.3 Create Lesson**

Summary: this diagram shows process of creating lesson or exercise

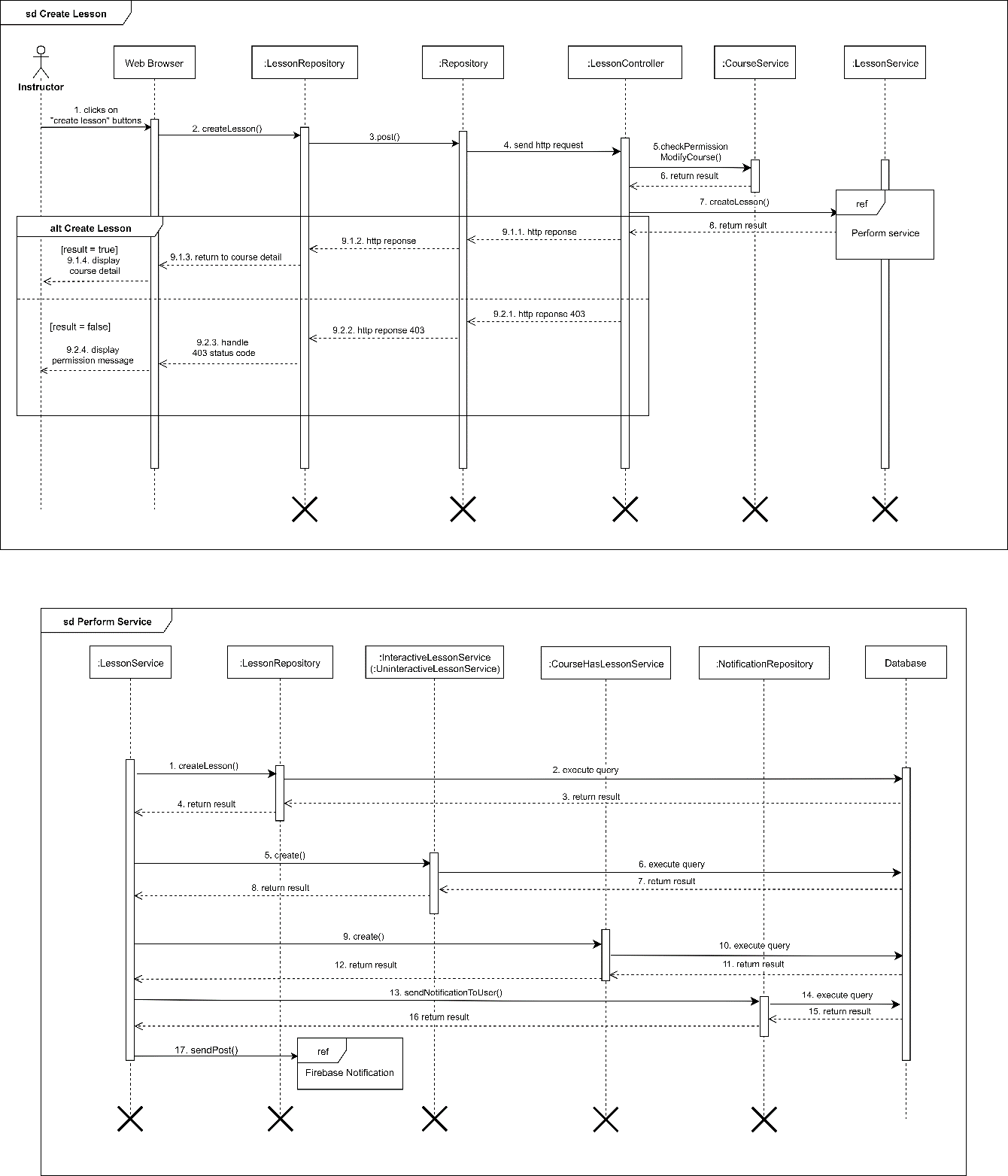


Figure 37: Sequence Diagram - <Instructor> Create Lesson

### **10.2.4 Edit Lesson**

Summary: this diagram shows process of editing lesson or exersise

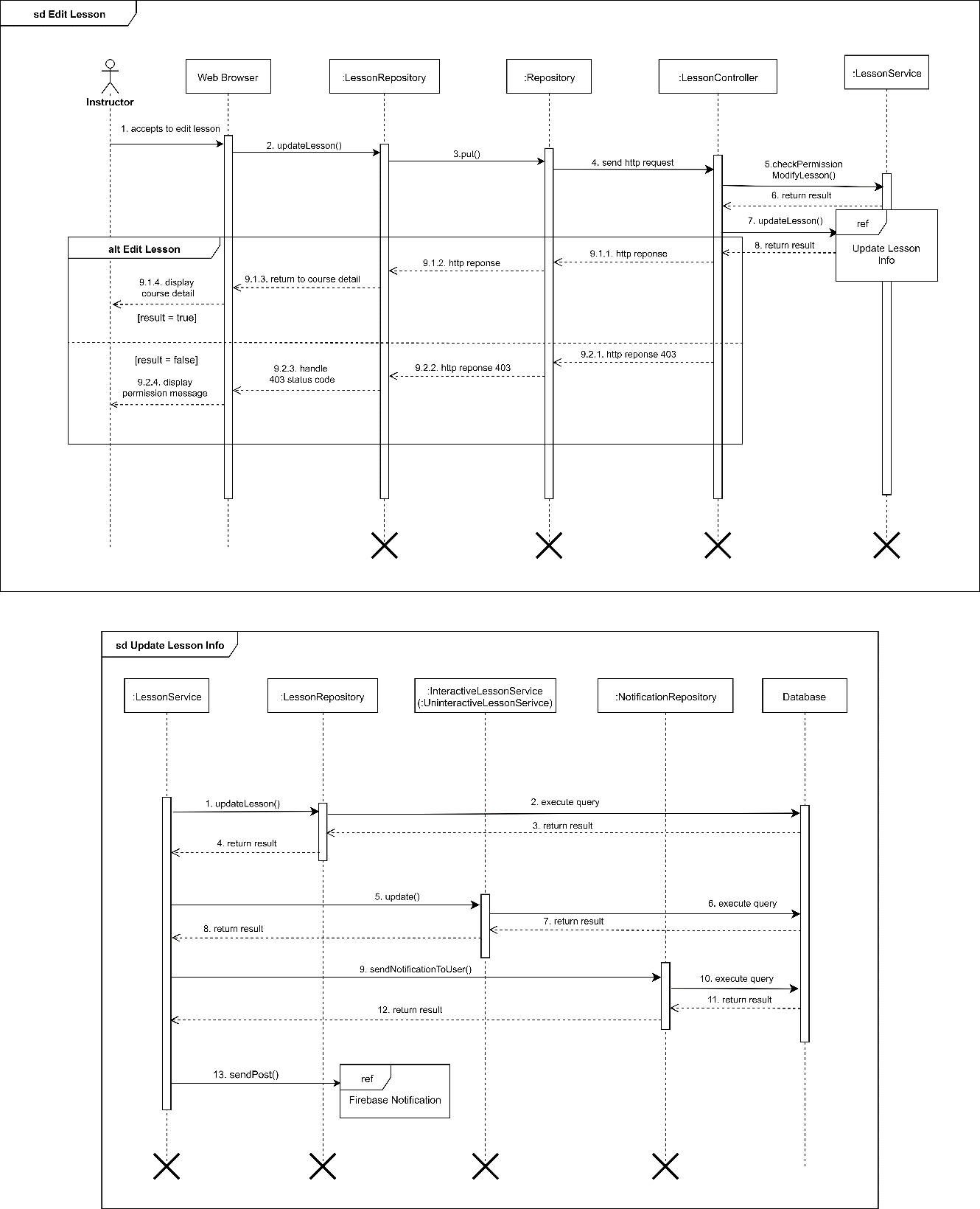


Figure 38: Sequence Diagram - <Instructor> Edit Lesson

## 10.3 Admin (Update Course Status)

Summary: this diagram shows process of update course status. This process is used for approve, reject or unpublish course.

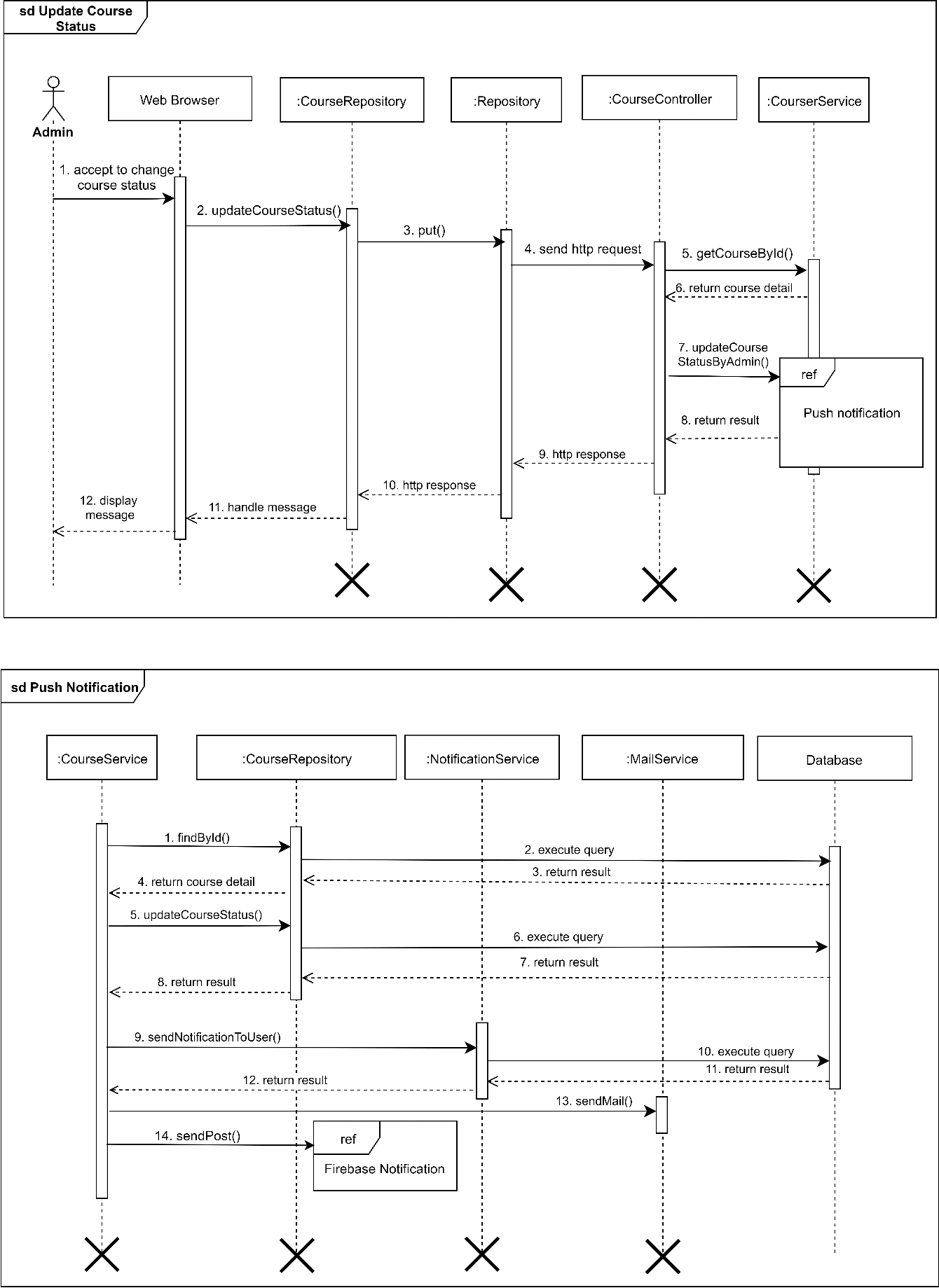


Figure 39: Sequence Diagram - <Admin> Update Course Status

## 10.4 System Bot (Play Chess With Learner)

Summary: this diagram shows process of play chess with learner.

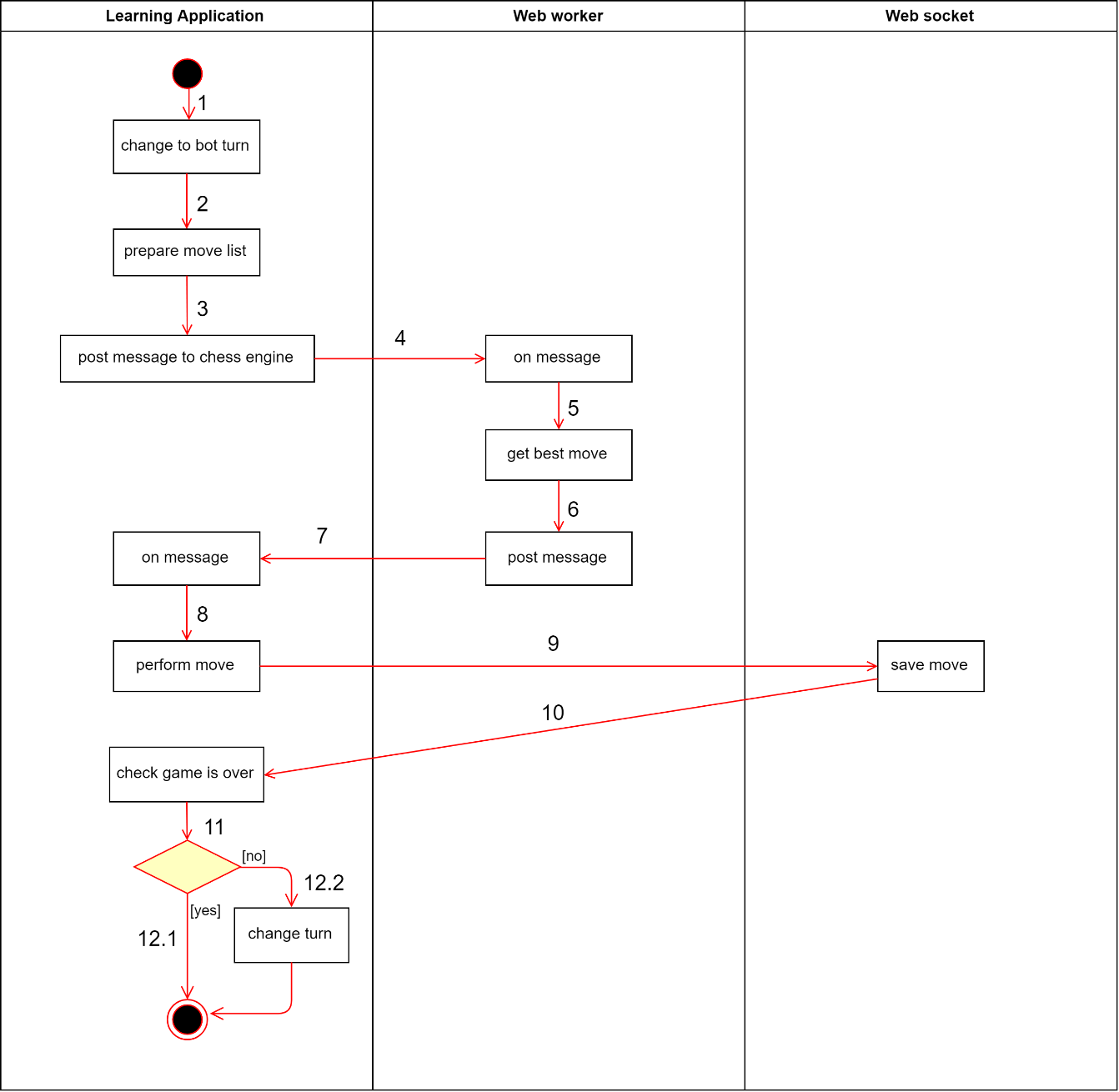


Figure 40: Activity Diagram - <Admin> Play Chess With Learner

**11. Database Relationship Diagram**

## 11.1 Physical Diagram

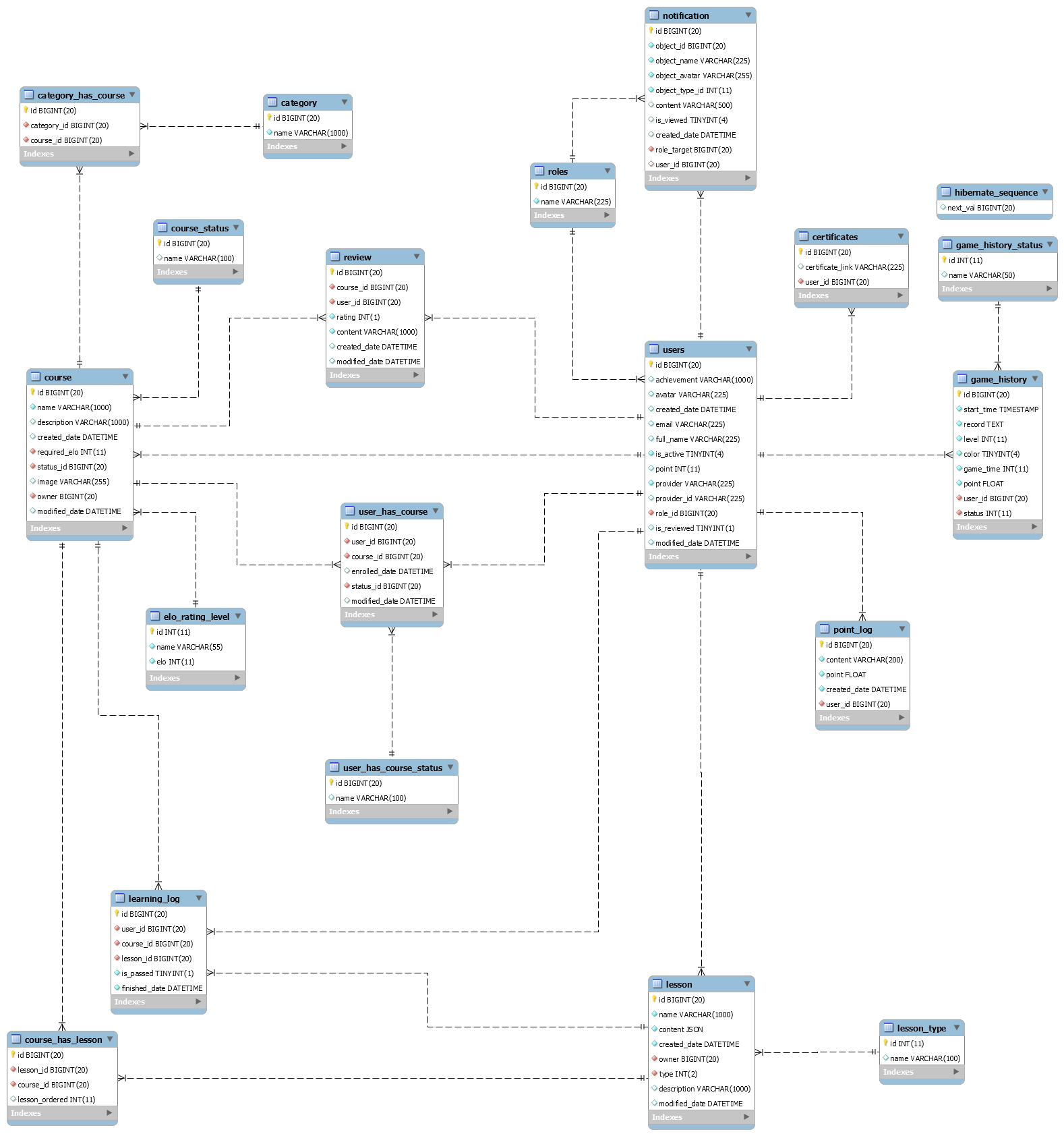


Figure : Physical diagram

## 11.2 Data dictionary

|  |  |
| --- | --- |
| **DATA DICTIONARY: DESCRIBE CONTENT OF ALL TABLES** | |
| **Table name** | **Description** |
| users | Contains the user information |
| roles | Contains the user role name |
| point\_log | Contains the log of user point transactions |
| notification | Contains the notification for each user |
| certificates | Contains the certificates for instructor |
| game\_history | Contains the game history for learner |
| game\_history\_status | Contains the status name of game |
| review | Contains the course review information |
| course | Contains the course information |
| course\_status | Contains the status name for a course |
| elo\_rating\_level | Contains the level name based on elo rating |
| category | Contains the category information |
| category\_has\_course | Contains the relationship between category and course |
| user\_has\_course | Contains the relationship between course and user |
| user\_has\_course\_status | Contains the status name of course for user |
| lesson | Contains the lesson information |
| lesson\_type | Contains the lesson type name |
| course\_has\_lesson | Contains the relationship between course and lesson |
| learning\_log | Contains the learning progress information of a learner |
| hibernate\_sequence | Table of Hibernate for tracking auto-increment ID |

Table : Data Table Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table Name | Attributes | Description | Domain | Null |
| user | id | Unique identifier of a user | BIGINT(20) | NO |
| achievement | Instructor’s achievement | VARCHAR(1000) | YES |
| avatar | User’s avatar URL | VARCHAR(225) | YES |
| created\_date | Created time of a user | DATETIME | YES |
| email | User’s email | VARCHAR(225) | YES |
| full\_name | User’s full name | VARCHAR(225) | YES |
| is\_active | Current user’s status | TINYINT(4) | NO |
| point | Learner’s point | FLOAT | YES |
| provider | User’s social network provider name | VARCHAR(225) | NO |
|  | provider\_id | User’s social network provider identifier | VARCHAR(225) | YES |
| role\_id | Identifier of the user’s role | BIGINT(20) | NO |
| is\_reviewed | Tracking instructor is reviewed or not | TINYINT(1) | NO |
| modified\_date | User information updated time | DATETIME | YES |
| roles | id | Unique identifier of a role | BIGINT(20) | NO |
| name | Name of role | VARCHAR(225) | NO |
| point\_log | id | Unique identifier of a point’s log | BIGINT(20) | NO |
| content | Point transaction’s reason | VARCHAR(200) | NO |
| point | Number of points | FLOAT | NO |
| created\_date | Log’s time | DATETIME | NO |
| user\_id | Point owner’s identifier | BIGINT(20) | NO |
| notification | id | Unique identifier of the notification | BIGINT(20) | NO |
| object\_id | Source object identifier | BIGINT(20) | NO |
| object\_name | Source object name | VARCHAR(225) | NO |
| object\_avatar | Source object avatar | VARCHAR(225) | NO |
| object\_type\_id | Source object type identifier | INT(11) | NO |
| content | Notification content | VARCHAR(500) | YES |
| is\_viewed | Tracking notification is viewed or not | TINYINT(4) | YES |
| created\_date | Notification time | DATETIME | YES |
| role\_target | Target role identifier | BIGINT(20) | NO |
| user\_id | Target user identifier | BIGINT(20) | NO |
| certificates | id | Unique indentifier of a certificate | BIGINT(20) | NO |
| certificate\_link | Instructor certificate image URL | VARCHAR(225) | YES |
| user\_id | Instructor’s identifier | BIGINT(20) | NO |
| game\_history | id | Unique identifier of a game history | BIGINT(20) | NO |
| start\_time | Game’s start time | TIMESTAMP | NO |
| record | Game’s PGN | TEXT | NO |
| color | Learner’s colour | TINYINT(4) | NO |
| game\_time | Playing time for each player | INT(11) | NO |
| point | Bonus point when ending game | FLOAT | NO |
| user\_id | Learner’s identifier | BIGINT(20) | NO |
| status | Game’s status | INT(11) | NO |
| game\_history\_status | id | Unique identifier of a game status | INT(11) | NO |
| name | Game status name | VARCHAR(50) | YES |
| review | id | Unique identifier of a review | BIGINT(20) | NO |
| course\_id | Course’s identifier which contains the review | BIGINT(20) | NO |
| user\_id | Reviewer’s identifier | BIGINT(20) | NO |
| rating | Review’s rating point | INT(1) | NO |
| content | Review’s content | VARCHAR(1000) | NO |
| created\_date | Review’s time | DATETIME | YES |
| modified\_date | Review’s updated time | DATETIME | YES |
| course | id | Unique identifier of a course | BIGINT(20) | NO |
| name | Course’s name | VARCHAR(1000) | NO |
| description | Course’s description | VARCHAR(1000) | YES |
| created\_date | Course’s created time | DATETIME | YES |
| required\_elo | Course’s required level identifier | INT(11) | NO |
| status\_id | Current course’s status identifier | BIGINT(20) | NO |
| image | Course’s cover image URL | VARCHAR(255) | YES |
| owner | Course’s owner identifier | BIGINT(20) | NO |
| modified\_date | Course’s updated time | DATETIME | YES |
| course\_status | id | Unique identifier of a course’s status | BIGINT(20) | NO |
| name | Course’s status name | VARCHAR(100) | YES |
| category | id | Unique identifier of a category | BIGINT(20) | NO |
| name | Category’s name | VARCHAR(1000) | NO |
| category\_has\_course | id | Unique identifier of a relationship between course and category | BIGINT(20) | NO |
| category\_id | Category identifier | BIGINT(20) | NO |
| course\_id | Course identifier | BIGINT(20) | NO |
| elo\_rating\_level | id | Unique identifier of a level | INT(11) | NO |
| name | Level’s name | VARCHAR(55) | NO |
| elo | Level’s required Elo | INT(11) | NO |
| user\_has\_  course | id | Unique identifier of a relationship between user and course | BIGINT(20) | NO |
| user\_id | User identifier | BIGINT(20) | NO |
| course\_id | Course identifier | BIGINT(20) | NO |
| enrolled\_date | Enrol’s time | DATETIME | YES |
| status\_id | Status identifier between course and user | BIGINT(20) | NO |
| modified\_date | Relationship’s status updated time | DATETIME | YES |
| user\_has\_  course\_status | id | Unique identifier of a status name for the relationship between user and course | BIGINT(20) | NO |
| name | Status’s name | VARCHAR(100) | YES |
| lesson | id | Unique identifier of a lesson | BIGINT(20) | NO |
| name | Lesson’s name | VARCHAR(1000) | NO |
| content | Lesson’s content | JSON | NO |
| created\_date | Lesson’s created time | DATETIME | NO |
| owner | Lesson’s owner identifier | BIGINT(20) | NO |
| type | Lesson’s type identifier | INT(2) | NO |
| description | Lesson’s description | VARCHAR(1000) | YES |
| modified\_date | Lesson’s updated time | DATETIME | YES |
| lesson\_type | id | Unique identifier of a lesson type | INT(11) | NO |
| name | Lesson type’s name | VARCHAR(100) | YES |
| course\_has\_  lesson | id | Unique identifier of a relationship between course and lesson | BIGINT(20) | NO |
| lesson\_id | Lesson identifier | BIGINT(20) | NO |
| course\_id | Course identifier | BIGINT(20) | NO |
| lesson\_ordered | Lesson ordered in course | INT(11) | YES |
| learning\_log | id | Unique identifier of a learning log | BIGINT(20) | NO |
| user\_id | Learner identifier | BIGINT(20) | NO |
| course\_id | Course identifier | BIGINT(20) | NO |
| lesson\_id | Lesson identifier | BIGINT(20) | NO |
| is\_passed | Tracking lesson is passed by the learner | TINYINT(1) | NO |
| finished\_date | Lesson finished time | DATETIME | NO |
| hibernate\_  sequence | next\_val | Next value of auto-increment key | BIGINT(20) | YES |

Table : Data Table Description

**12. Algorithms**

## 12.1 Convert chess move list from database to display list

### **12.1.1 Definition**

In our interactive lesson or exercise, the content is a chess game with existing move list, we need to display all chess move for the learner, and learner can click on each move to view the analysis of that chess puzzle. Moreover, chess game maybe contains more than 1 move for certain chess puzzle. Our system approves that 1 puzzle, a learner can view 2 moves, game move and another move from analyst, and we limit this action in 2 times, it means, in another move, we only have 1 another move again or the move depth is 3.

### **12.1.2 Define problem**

From above requirement, we used an array to display chess move for the learner, each object in array is a turn include: turn index, depth of move turn, if depth is 1, object will contain white move and black move and chess move information, such as: move direction, FEN string,… is store in every move, if the depth is 2, the object will contain an array of depth 2 move, and it is same with depth is 3. Then we use linked list to store chess move with JSON format in the database, and we need to convert move linked list to displayed list. For example:

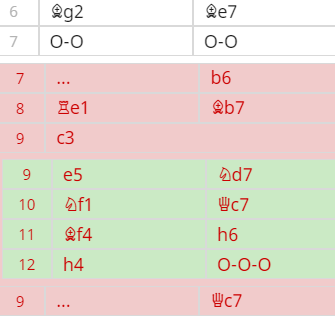


Figure : Target Move List

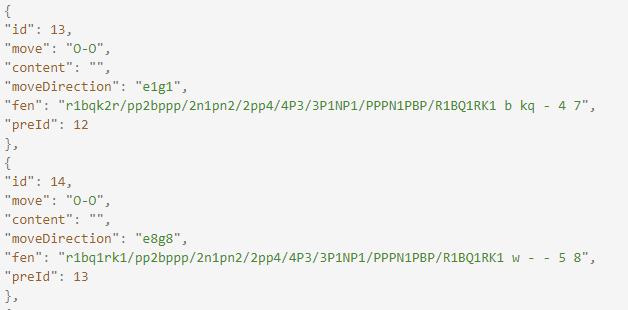


Figure : Sample move list from database

|  |  |
| --- | --- |
| Move list dictionary | |
| Key | Description |
| id | Move Identifier |
| move | Chess Move in SAN, use for display move to the learner |
| content | Description for puzzle follow with a move |
| moveDirection | Move direction include source square, target square and maybe it contains promotion piece name. Chess programming needs to move directly to generate a new position. |
| fen | FEN String to present for current a chess position |
| preId | Previous move identifier |

Table : Move list dictionary

### **12.1.3 Solution**

To display chess move history in array list to the client. We need to take 3 steps:

* We will implement an array list to simple linked list by adding 1 attribute is the next step to each element.
* After we have a simple linked list, we will use Depth First Search (DFS) Algorithm.
* We will save this tree in array again but with our format for rendering

Firstly, we need to implement a simple linked list because we need to know what is the next steps of any step.

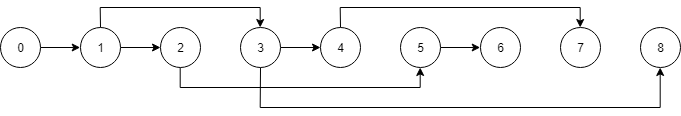
****

Figure : Example simple linked list

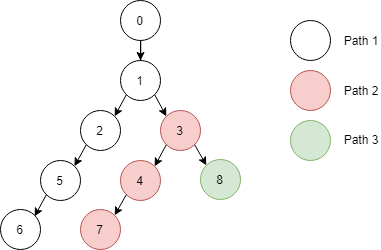


Figure : Example tree

With this tree, we have to find each path of the tree by using Recursion to implement Depth First Search (DFS) Algorithm. Of course, Breath First Search (BFS) is not suitable in this case.

Finally, while using DFS, we will save each node to element in the array with our format for rendering like this:

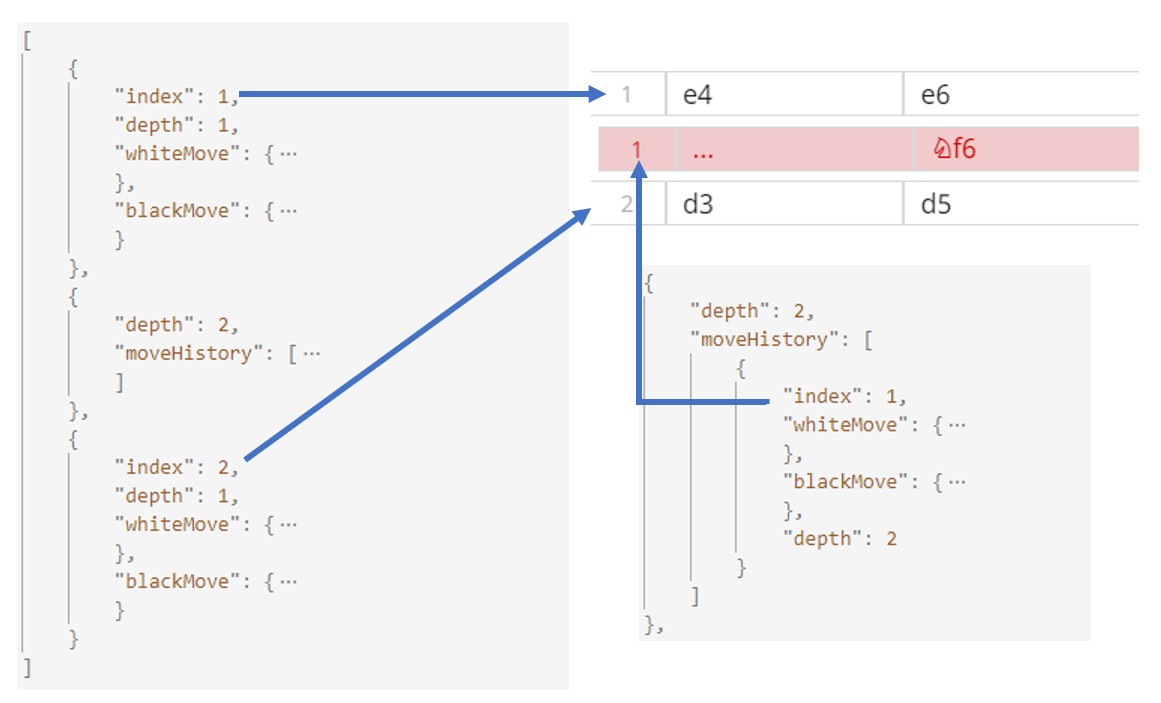


Figure : Move list result

### **12.1.4 Complexity**

The complexity of this algorithm is: O(N)

## 12.2 Building chess bot

### **12.2.1 Definition**

In COLS, we must build a chess bot, which can autoplay chess with the learner to improve their skills. We also use this bot to play with the learner in exercises.

### **12.2.2 Define problem**

In a chess game, when a move is performed, there will have more possible game have existed. The number of game will be increased after more moves. Then we cannot use exhausted method to get the best move, it will take the memory very much.

When human play chess, they also calculate move with some of the material properties, such as:

* Controlling centre squares are better than controlling unoccupied squares.
* Double pawns are not good, they cannot defend each other.

Nowadays, there are more chess programming, they combine the chess material properties and computed methods to analyse chess move in computer. In COLS, we also use existing chess engine to get the best move for a specific chess position, which is called Stockfishjs.

### **12.2.3 Solution**

In building play chess feature, we use stockfishjs chess engine to calculate the best move, Chessjs to handle the game and validates the player move. We also use Chessground library to show Chessboard for the learner.

There are some concepts in chess programming, then we will discuss an overview of them.

Chessboard: A chessboard will have 64 squares represents an 8x8 grid. In chess programming, they call file for each board column and rank for each board row. It means a chessboard contains 8 ranks: 1-8 and 8 columns: a-h. We also use rank and file to record a square, such as a1, b3, e6,…, it is called Algebraic notation (AN).

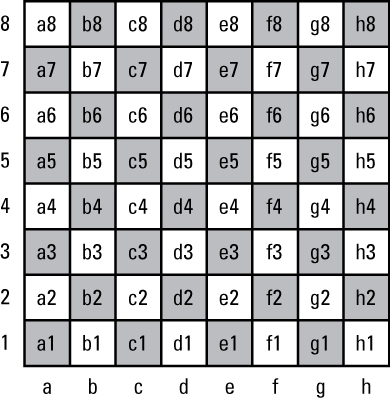


Figure : A chessboard with square in AN

FEN String: A FEN string is used to mark current position in a chess game, which will contain 6 parts – pieces position, move turn, castle tracking, en passant target, half move clock to handle 50 moves rule, and full move number.

* The first part in FEN String will use to represent chess pieces position by their name: **K**ing, **Q**ueen, **R**ook, **B**ishop, K**N**ight and **P**awn. The black pieces will be noted by lowercase (kqrbnp) and uppercase for white pieces. They will note chess pieces from rank 8 to rank 1 for each rank, and ranks will be joined at “/”. If a square which contains pieces, it will be noted by pieces name and empty square will be noted by number. When a rank has adjacent empty squares, the number will be incremental until a piece square, for example:

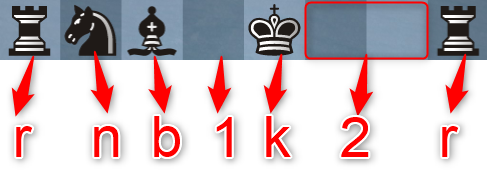


Figure : Sample rank in FEN String

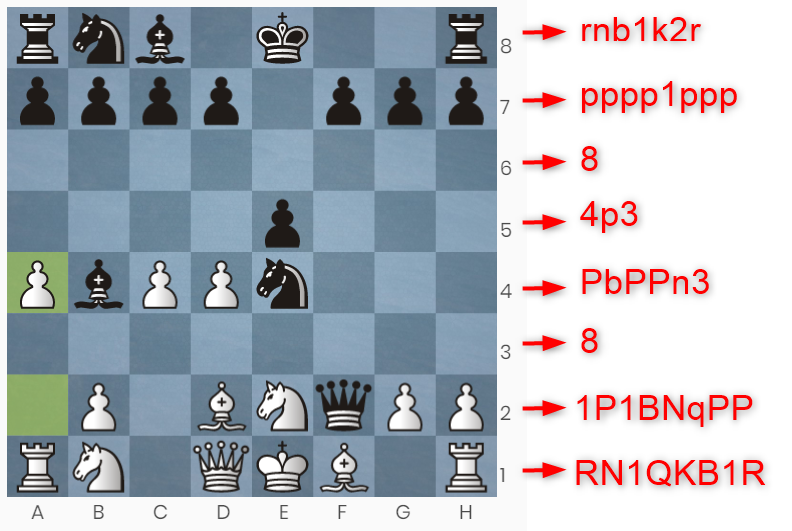


Figure : Sample Position in FEN String

(The above chess position will be noted: rnb1k2r/pppp1ppp/8/4p3/PbPPn3/8/1P1BNqPP/RN1QKB1R)

* The second part in FEN String will represent for move turn: “w” for the white turn in next move and “b” for the black turn.
* The third part is castling tracking, each character is the wing which player can use castling move: K (White King), Q (White Queen), k (Black King), q (Black Queen).
* The fourth part will display an en passant target square if a player performs a two squares move or “-“ for other moves. It means an en passant target move will be a move in rank 3 or 6.
* The fifth part is the total of half move since the last capture or pawn advance. It will be returned to 0 after a capture or pawn advance move and it is used to check 50 moves rule.
* The last part is the number of a full move in-game. It is count after a black’s move.

FEN String is important data, which can represent all states in the current game. In our chess bot, we use all above chess libraries follow this flow:

* When the game is started, the all chess libraries: chessjs, chessground, stockfishjs will have initial FEN String: rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1
* For player turn:
  + When a move is performed in the chessboard, Chessground will generate a move, which contains source square and target square (and promotion pieces for promotion move in lowercase), for example e2e4 or f7f8q (f7 to f8, promote to Queen).
  + The move will be sent to chessjs and chessjs will generate new FEN String by it. Chessground will render a new position from new FEN String. User can see new game position.
* For bot turn, we use UCI command to communicate with Stockfishjs to get the best move, the UCI command:

position startpos moves [string of moves from start game]

If the string of moves is empty, the engine will calculate from start game, else it will calculate game from the game position in the last move of string. The result will be the best move with move direction (source square, target square and maybe promotion piece name). We also use this command, it will use FEN String for analyzing in custom position:

position fen <fen string> moves [string of moves from start game]

* With the above UCI command, we need to add move after each half move.
* We also use chessjs to generate new FEN String and re-render new position.
* All the above step will repeat until the game is over. Chessjs lib will check FEN String to evaluate game is playing or over. After half move, chessjs also check possible moves and we will use possible moves to validate player move, if the move is not included, the player needs to perform again.

For building different bot level, we call different engine-level by this command: setoption name Skill Level value <skill\_level>

### **12.2.4 Flow chart**

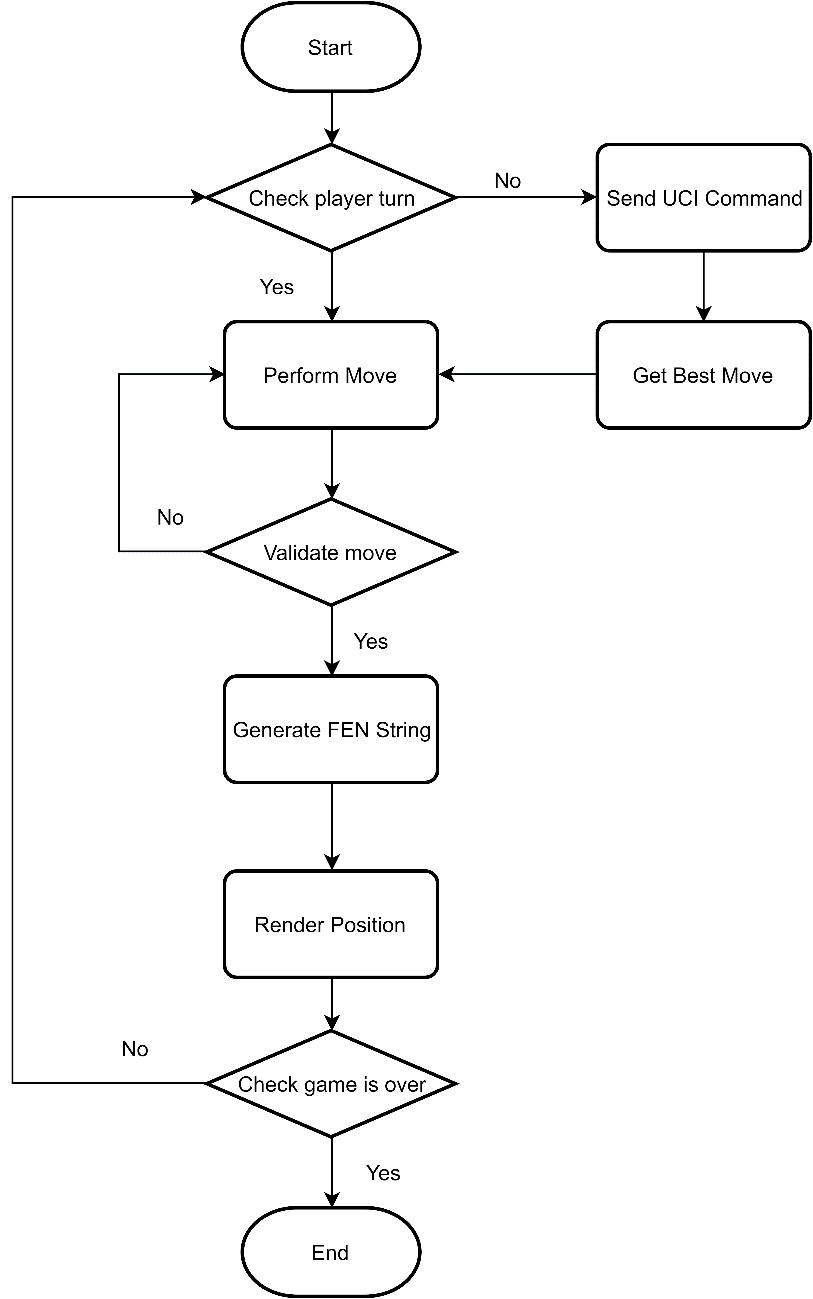


Figure : Building Chess Bot Flow chart

## 12.3 Building Elo rating system

### **12.3.1 Definition**

In COLS, we must evaluate the learner’s level by playing chess with the bot. Therefore, we need a standard rating system, which approved by FIDE (Fédération internationale des échecs).

### **12.3.2 Define problem**

In COLS, the learner will start with specially point based on their chosen level. Whenever learner complete playing chess with the bot, the system will update point of learner depend on a final result chess game.

Each level of stockfishjs chess bot will have a specially Elo point and learner point will be calculated following the Elo rating system.

### **12.3.3 Solution**

In the learner's rankings, we use the International Chess Association (FIDE) rankings15 as the standard and readjust it to match the ranking of the system. Each rank will correspond to an Elo range:

* Beginner (0 ~ 1000 Elo points)
* Minor (1000 ~ 1200 Elo points)
* Intermediate (1200 ~ 1400 Elo points)
* Major (1400 ~ 1600 Elo points)
* Master( > 1600 Elo points)

When users create an account will allow them to choose their current level. Then the system will fill in the corresponding points for them.

In classifying each level of the bot in the Elo system. We use the CCRL 4040 calculation to help evaluate the bot's Elo level when learners play:

Rank Name                    Elo + - games score oppo. draws

   1 Stockfish 10 x64 1T20  3495 102 102 400 99%  2804 2%

   2 Stockfish 10 x64 1T19  2855 32 32 400 45%  2965 26%

   3 Stockfish 10 x64 1T18  2825 32 32 400 41%  2972 27%

   4 Stockfish 10 x64 1T17  2793 32 32 400 36%  2980 27%

   5 Stockfish 10 x64 1T16  2745 23 23 800 50%  2788 24%

   6 Stockfish 10 x64 1T15  2677 30 30 400 60%  2602 26%

   7 Stockfish 10 x64 1T14  2652 30 30 400 56%  2608 26%

   8 Stockfish 10 x64 1T13  2552 31 31 400 39%  2633 21%

   9 Stockfish 10 x64 1T12  2458 24 24 800 50%  2444 14%

  10 Stockfish 10 x64 1T11  2370 33 33 400 65%  2253 14%

  11 Stockfish 10 x64 1T10  2269 32 32 400 49%  2279 12%

  12 Stockfish 10 x64 1T09  2181 33 33 400 36%  2301 13%

  13 Stockfish 10 x64 1T08  2105 25 25 800 49%  2108 8%

  14 Stockfish 10 x64 1T07  2050 35 35 400 66%  1910 7%

  15 Stockfish 10 x64 1T06  1950 34 34 400 51%  1935 5%

  16 Stockfish 10 x64 1T05  1862 34 34 400 39%  1957 4%

  17 Stockfish 10 x64 1T04  1722 29 29 800 52%  1683 3%

  18 Stockfish 10 x64 1T03  1589 37 37 400 68%  1408 2%

  19 Stockfish 10 x64 1T02  1427 36 36 400 48%  1449 2%

  20 Stockfish 10 x64 1T01  1302 38 38 400 32%  1480 1%

  21 Stockfish 10 x64 1T00  1181 42 42 400 19%  1510 0%

The realistic part of the ellipse after the learner completes the game of chess with the bot. It includes the probability of making ellipse commands between learners and bots and finding K-factors.

Probability of ellipse between learner and bot In which learners are A and bots are B:

: expected score of A

: expected score of B

: rating of A (current elo point)

: rating of B (current elo point)

**Note that:**

: new rating of A

: actually scored (1 for win , 0 for lose and drawn for 0.5)

K: k-factors

FIDE used the following ranges for K15:

* K = 40 for a player new to the rating list until he has completed events with at least 30 games.
* K = 20 as long as a player's rating remains under 2400.
* K = 10 once a player's published rating has reached 2400 and remains at that level subsequently, even if the rating drops below 2400.

## 12.4 Finding Similar User

### **12.4.1 Definition**

In order to make a recommendation for learners based on existing data, we use Collaborative filtering technique by memory-based. Firstly, we need to find similar users with the current user.

### **12.4.2 Define problem**

The existing data is a rating of learners for their courses. We would calculate similarity based on the interest of learners with courses.

### **12.4.3 Solution**

We apply cosin similarity to calculate similar between 2 users:

: Rating in course of user A

: Rating in course of user B

Result range: [0,1] with 1 is the most similar

### **12.4.4 Example**

There are ratings of 4 users for course 1 to 5:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Course** | **User A** | **User B** | **User C** | **User D** |
| 1 | 4.0 | 5.0 | 4.5 | 5.0 |
| 2 | 5.0 | null | 4.0 | 4.5 |
| 3 | 4.0 | 4.0 | null | 5.0 |
| 4 | 3.0 | 3.5 | 4.0 | null |
| 5 | null | 3.0 | 3.0 | 4.5 |

Similarity between User A and other User:

### **12.4.5 Complexity**

The Complexity is: O(

## 12.5 Course Rating Prediction (User-based filtering)

### **12.5.1 Definition**

From the above result, we perform next step is predicting ratings for courses which user has not been learned based on similar users.

### 12.5.2 Define problem

Each user has different courses, but if they have large similarity, we can predict the rating for courses which they have not been learned.

### 12.5.3 Solution

Calculating the weighted average based on rating and similarity of top n user:

: Predicted rating of course

: Rating in course of user

: Similarity between current user with user

### **12.5.4 Example**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Course** | **User A** | **User B** | **User C** | **User D** |
| 1 | 4.0 | 5.0 | 4.5 | 5.0 |
| 2 | 5.0 | null | 4.0 | 4.5 |
| 3 | 4.0 | 4.0 | null | 5.0 |
| 4 | 3.0 | 3.5 | 4.0 | null |
| 5 | null | 3.0 | 3.0 | 4.5 |

We will use above data and assump that User B, C, D is top 3 similar user of A. The similar between A and them are sim(A,B) = 0.78, sim(A,C) = 0.7864 and sim(A,D) = 0.809.

The predicted rating of A for course 5 is:

### **12.5.5 Complexity**

The Complexity is: O(

## 12.6 Item-based filtering

### **12.6.1 Definition**

In COLS, we also calculate similar between system courses and user enrolled courses.

### **12.6.2 Define problem**

When the user views a specific course, the system will find similar courses with current course to suggest for the user. The suggested course must be the same with learner skill level.

### **12.6.3 Solution**

We also using cosin similarity with special parameter to calculate similar course.

Courses in the system will have similar attributes:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Course A (current course)** | **Course B** |
| Same instructor | 1 |  |
| Same category | 1 |  |
| Course rating |  |  |

x: value is 0 if course B has different instructor and 1 if course B has same instructor

y =

: Average rating of course A

: Average rating of course B

Result range: [0,1] with 1 is the most similar

### **12.6.4 Example**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Course A (current course)** | **Course B** | **Course C** |
| Same instructor | 1 | 0 | 1 |
| Same category | 1 | 0.5 | 0.33 |
| Course rating | 4 | 4.5 | 4 |

Similarity between Course A and other courses:

Course C is almost identical to A than the other course because the similarity is larger.

### **12.6.5 Complexity**

The Complexity is: O(

**13. Future plan**

Current system also has problems and it will be developed in the future:

* Detect cheating in game to improve point reliability.
* Support chess playing peer to peer to extend practice opportunity for learner.
* Develop money collecting features for instructor.
* Automatively administrator working.
* Develop course suggestion feature based on learner profile.

**14. Appendix**

1. **The Unified Modeling Language** *[Online]* Available: <https://www.omg.org/spec/UML/2.0/About-UML/> Accessed on: Sept. 2019.
2. **Multitier Architecture** *[Online]* Available: <https://www.ibm.com/support/knowledgecenter/en/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/covr_3-tier.html> Accessed on: Sept. 2019.
3. Chen, P.P., "The Entity-Relationship Model: Toward a Unified View of Data". *ACM Trans. on Database Systems*.
4. **MySQL** *[Online]* Available: <https://dev.mysql.com/doc/> Accessed on: Sept. 2019.
5. **Spring Framework** *[Online]* Available: <https://spring.io/docs> Accessed on: Sept. 2019.
6. **VueJS** *[Online]* Available: <https://vuejs.org/v2/guide/> Accessed on: Sept. 2019.
7. **Vuetify** *[Online]* Available:

<https://vuetifyjs.com/en/getting-started/quick-start> Accessed on: Sept. 2019.

1. **Stockfish** *[Online]* Available: <https://stockfishchess.org/> Accessed on: Sept. 2019.
2. **Stockfishjs** *[Online]* Available: <https://github.com/nmrugg/stockfish.js/> Accessed on: Sept. 2019.
3. **chessjs** *[Online]* Available: <https://github.com/jhlywa/chess.js> Accessed on: Sept. 2019.
4. **chessground** *[Online]* Available: <https://github.com/ornicar/chessground> Accessed on: Sept. 2019.
5. **chessboardjs** *[Online]* Available: <https://chessboardjs.com/docs> Accessed on: Sept. 2019.
6. **FEN String** *[Online]* Available: <https://www.chessprogramming.org/Forsyth-Edwards_Notation> Accessed on: Sept. 2019.
7. **Chessprogramming wikipedia** *[Online]* Available: <https://www.chessprogramming.org/Main_Page> Accessed on: Sept. 2019.
8. **Elo Rating System** *[Online]* Available: <https://www.fide.com/fide/handbook> Accessed on: Sept. 2019.