

**SELLING ONLINE COURSE SYSTEM PROJECT**

**Software Design Specification**

– SUMMER 2023 –

**Record of changeS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **A\* M, D** | **In charge** | **Change Description** |
| 26/07/2023 | A | Phạm Hà Ngọc Mai | Draw additional Sequence Diagram |
| 11/07/2023 | M | Phạm Hà Ngọc Mai | Modify the font, Redraw the Sequence Diagram |
| 16/07/2023 | M | Phạm Hà Ngọc Mai | Modified class diagram |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

\*A - Added M - Modified D - Deleted

**Table of Contents**

[I.Introduction 4](#_heading=h.gjdgxs)

[1. Purpose 4](#_heading=h.30j0zll)

[2. System Overview 5](#_heading=h.1fob9te)

3[. Design Map 5](#_heading=h.1fob9te)

[II. Overview 4](#_heading=h.gjdgxs)

[1. Code Packages 4](#_heading=h.30j0zll)

[2. Database Design 5](#_heading=h.1fob9te)

[a. Database Schema 5](#_heading=h.3znysh7)

[b. Table Description 5](#_heading=h.2et92p0)

3[.](#_heading=h.30j0zll)  Development tools and standards5

[a.](#_heading=h.3znysh7) Development tools [5](#_heading=h.3znysh7)

b[.](#_heading=h.3znysh7) Development standards [5](#_heading=h.3znysh7)

I[II. Code Designs 8](#_heading=h.tyjcwt)

[1. Public User 8](#_heading=h.3dy6vkm)

[a. Class Diagram 8](#_heading=h.1t3h5sf)

[b. Class Specifications 9](#_heading=h.4d34og8)

[c. Sequence Diagram(s) 9](#_heading=h.17dp8vu)

[2. Public Order 17](#_heading=h.3rdcrjn)

[a. Class Diagram 17](#_heading=h.26in1rg)

[b. Class Specifications 17](#_heading=h.lnxbz9)

[c. Sequence Diagram(s) 18](#_heading=h.35nkun2)

[3.Public Course 20](#_heading=h.1ksv4uv)

[a. Class Diagram 20](#_heading=h.44sinio)

[b. Class Specifications 20](#_heading=h.2jxsxqh)

[c. Sequence Diagram(s) 21](#_heading=h.z337ya)

[4. Public Flashcards 22](#_heading=h.3j2qqm3)

[a. Class Diagram 22](#_heading=h.1y810tw)

[b. Class Specifications 22](#_heading=h.4i7ojhp)

[c. Sequence Diagram(s) 23](#_heading=h.2xcytpi)

[5. Public News 25](#_heading=h.1ci93xb)

[a. Class Diagram 25](#_heading=h.3whwml4)

[b. Class Specifications 26](#_heading=h.2bn6wsx)

[c. Sequence Diagram(s) 26](#_heading=h.qsh70q)

# I. Introduction

## 1. Purpose

The purpose of the Online Course Selling System is to provide a platform for selling and managing online courses. This system allows course creators to create and upload their courses, and users to browse and purchase the courses according to their interests and needs.

## 2. System Overview

The Online Course Selling System aims to enhance the process of selling and accessing educational content in a digital format. It provides a user-friendly interface for both course creators and users. Course creators can upload their courses, set pricing, and manage course content, while users can browse through available courses, make purchases, and access course materials.

The system includes the following key features:

* Course Creation and Management: Course creators can create and upload their courses, including videos, documents, quizzes, and other educational materials. They can also organize course content into modules and lessons for better structure.
* User Registration and Authentication: Users can create accounts, log in, and manage their profiles. This enables personalized experiences, such as tracking course progress and accessing purchased courses.
* Course Catalog and Search: Users can explore the course catalog, browse through various categories, and search for specific courses based on keywords, topics, or instructors.
* Shopping Cart and Payment Processing: Users can add courses to their shopping cart and proceed to checkout for secure payment processing. Various payment methods, such as credit cards or online payment gateways, can be integrated into the system.
* Course Reviews and Ratings: Users can provide feedback and ratings for the courses they have taken, helping other users make informed decisions.
* Course Progress Tracking: The system allows users to track their progress within each course, marking completed lessons and modules. This feature enables users to easily resume their learning from where they left off.
* Notifications and Reminders: Users can receive notifications and reminders about upcoming course deadlines, new course releases, or important announcements from course creators.

## 3. Design Map

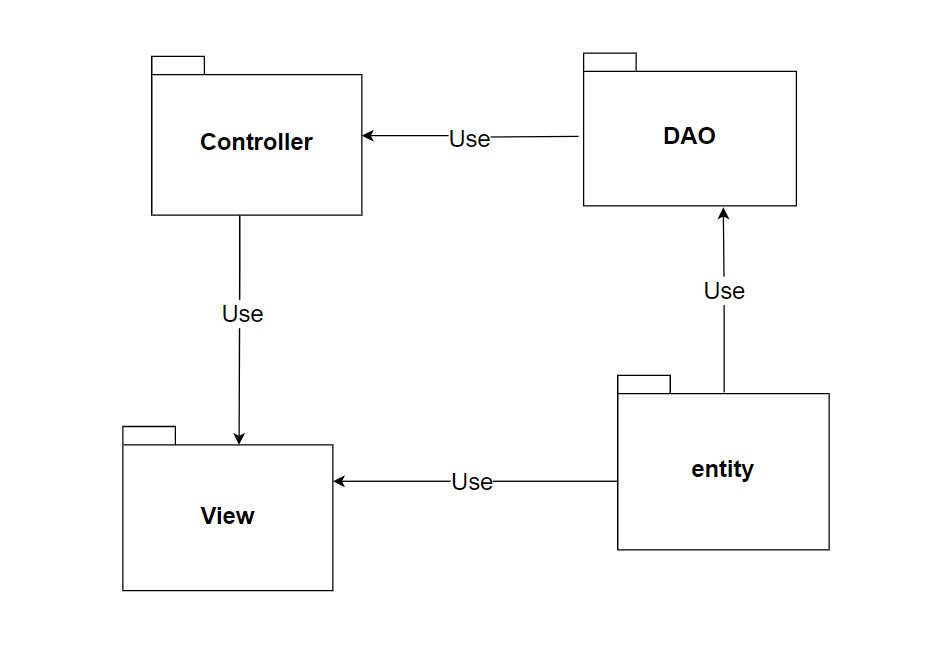
The design artifacts for the Online Course Selling System include:

* User Interface Design: This section describes the visual and interactive design of the system, including wireframes, mockups, and user flow diagrams.
* Database Design: It outlines the database schema and relationships necessary to store course information, user data, transactions, and other relevant data.
* Course Creation and Management Workflow: This artifact details the process and steps involved in creating and managing courses within the system.
* User Registration and Authentication Workflow: It explains the workflow for user registration, login, and authentication processes.
* Payment Processing Integration: This section describes the integration of payment gateways or services for secure transaction processing.
* Course Catalog and Search Implementation: It outlines how the course catalog will be populated, organized, and searchable within the system.
* Course Progress Tracking and Notifications: This artifact explains how course progress tracking and notifications will be implemented to enhance user experience.

These design artifacts collectively contribute to the development and implementation of the Online Course Selling System, ensuring a seamless user experience for both course creators and learners

# II. Overview

## 1. Code Packages

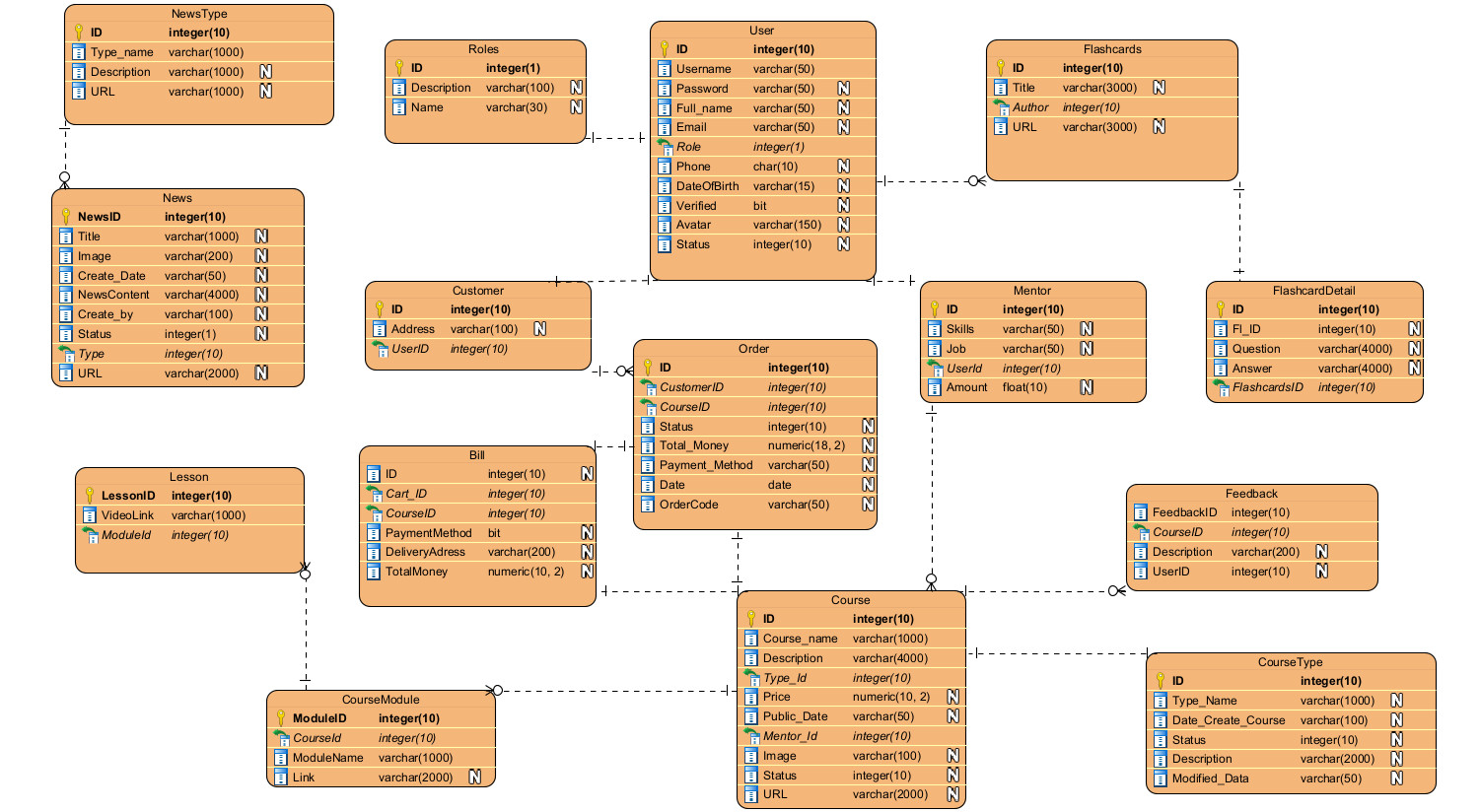


***Package descriptions***

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| 1 | controller | Contains components related to the management of user requests, including navigation agents, control functions, routing methods, and error handling. |
| 2 | DAO | Contains components related to data processing, including methods to access the database, edit data in the database. |
| 3 | entity | Contains objects that represent tables in the database. |
| 4 | View | View is responsible for displaying information to the user and interacting with the user. |

## 2. Database Design

### a. Database Schema



### b. Table Description

**News:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | NewsID | int |  | x | x | x | PK |
| 2 | Title | nvarchar | 1000 |  | x |  |  |
| 3 | Image | nvarchar | 200 |  | x |  |  |
| 4 | Create\_Date | nvarchar | 50 |  | x |  |  |
| 5 | NewsContent | nvarchar | 4000 |  | x |  |  |
| 6 | Create\_By | nvarchar | 100 |  | x |  |  |
| 7 | Status | int |  |  | x |  |  |
| 8 | Type | int |  |  | x | x | FK |

**NewsType:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | ID | int |  | x | x | x | PK |
| 2 | Type\_name | nvarchar | 100 0 |  | x |  |  |
| 3 | Description | nvarchar | 1000 |  | x |  |  |

**Quiz:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | ID | int |  | x | x | x | PK |
| 2 | ModuleID | int |  |  |  | x | FK |
| 3 | VideoLink | nvarchar | 2000 |  |  |  |  |

**CourseModule:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | ModuleID | int |  | x | x | x | PK |
| 2 | Question | nvarchar | MAX |  |  |  |  |
| 3 | Answer | nvarchar | 10 |  |  |  |  |

**Course:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | ID | int |  | x | x | x | PK |
| 2 | Course\_name | nvarchar | 1000 |  |  |  |  |
| 3 | Type\_ID | int |  |  |  | x | FK |
| 4 | Image | nvarchar | 1000 |  |  |  |  |
| 5 | Status | int |  |  |  |  |  |
| 6 | Description | nvarchar | 4000 |  |  |  |  |
| 7 | Mentor\_ID | int |  |  |  | x | FK |
| 8 | Price | numeric | (10,2) |  |  |  |  |
| 9 | Public\_Date | nvarchar | 50 |  |  |  |  |
| 10 | URL | nvarchar | 2000 |  |  |  |  |

**Feedback:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | FeedbackID | int |  | x | x | x | PK |
| 2 | CourseID | int |  |  | x | x | FK |
| 3 | Description | nvarchar | 2000 |  | x |  |  |
| 4 | UserID | int |  |  |  | x | FK |

**User:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | ID | int |  | x | x | x | PK |
| 2 | Username | nvarchar | 50 |  |  |  |  |
| 3 | Password | nvarchar | 50 |  |  |  |  |
| 4 | Full\_name | nvarchar | 50 |  |  |  |  |
| 5 | [E-mail] | nvarchar | 50 |  |  |  |  |
| 6 | Role | Int |  |  |  | x | FK |
| 7 | Phone | char | 10 |  |  |  |  |
| 8 | [Date of Birth] | nvarchar | 15 |  |  |  |  |
| 9 | Verified | bit |  |  |  |  |  |
| 10 | Avatar | nvarchar | 150 |  |  |  |  |
| 11 | Status | int |  |  |  |  |  |

**Mentor:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | ID | int |  | x | x | x | PK |
| 2 | Skills | nvarchar | 50 |  |  |  |  |
| 3 | Job | nvarchar | 50 |  |  |  |  |
| 4 | UserID | int |  |  | x | x | FK |
| 5 | Amount | float |  |  |  |  |  |

**Roles:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | ID | int |  | x | x | x | PK |
| 2 | Description | nvarchar | 100 |  | x |  |  |
| 3 | Name | nvarchar | 30 |  | x |  |  |

**CourseType:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | ID | int |  | x | x | x | PK |
| 2 | Type\_Name | nvarchar | 1000 |  | x |  |  |
| 3 | Date\_Create\_Course | nvarchar | 100 |  | x |  |  |
| 4 | Status | int |  |  | x |  |  |
| 5 | Description | nvarchar | 2000 |  | x |  |  |
| 6 | Modified\_date | nvarchar | 50 |  |  |  |  |

**Vouchers:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | VoucherID | int |  | x | x | x | PK |
| 2 | Discount | int |  |  | x |  |  |
| 3 | Quantity | int |  |  | x |  |  |
| 4 | VoucherCode | nvarchar | 10 |  |  |  |  |
| 5 | CourseID | int |  |  |  | x | FK |

**Order:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | ID | int |  | x | x | x | PK |
| 2 | CustomerID | int |  |  | x | x | FK |
| 3 | Status | int |  |  | x |  |  |
| 4 | Total\_Money | numeric | (18,2) |  |  |  |  |
| 5 | Payment\_Method | nvarchar | 50 |  |  |  |  |
| 6 | Date | date |  |  |  |  |  |
| 7 | OrderCode | nvarchar | 50 |  | x |  |  |

**Customer:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | ID | int |  | x | x | x | PK |
| 2 | Address | nvarchar | 100 |  |  |  |  |
| 3 | UserID | int |  |  | x | X | FK |

**Customer:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | ID | int |  | x | x | x | PK |
| 2 | OrderID | int |  |  | x |  |  |
| 3 | Course\_ID | int |  |  | x | X | FK |
| 4 | Price | numeric | (10,2) |  | x |  |  |

## 3. DEVELOPMENT TOOLS AND STANDARD

### a. Development Tools

#### HTML

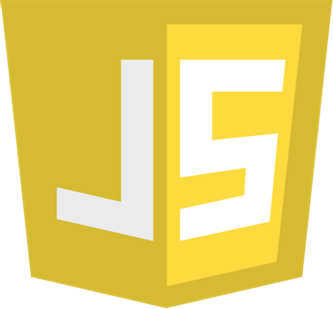


HTML stands for HyperText Markup Language. It is a standard markup language for web page creation. It allows the creation and structure of sections, paragraphs, and links using HTML elements (the building blocks of a web page) such as tags and attributes.

#### CSS

[CSS](https://developer.mozilla.org/en-US/docs/Glossary/CSS) (Cascading Style Sheets) allows you to create great-looking web pages, but how does it work under the hood? This article explains what CSS is with a simple syntax example and covers some key terms about the language.

#### JavaScript

JavaScript, often abbreviated as JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices. JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard.[10] It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

#### JSP/Servlet



JSP stands for JavaServer Pages which is a technology for developing dynamic web pages. JSP helps developers to insert java code into HTML pages using special JSP tags. JSP is a type of Java servlet designed to create the user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, JSP actions and commands.

The Java Servlet API allows software developers to add dynamic content to Web servers using the Java platform. It generates content that is usually HTML, sometimes other languages such as XML.

#### SQL

Microsoft SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product whose primary function is to store and retrieve data required by other software applications.

#### Java 8



Java 8 is a revolutionary release of the world’s #1 development platform. It includes a huge upgrade to the Java programming model and a coordinated evolution of the JVM, Java language, and libraries. Java 8 includes features for productivity, ease of use, improved polyglot programming, security and improved performance. Welcome to the latest iteration of the largest, open, standards-based, community-driven platform

#### Tomcat 10



This is the top-level entry point of the documentation bundle for the Apache Tomcat Servlet/JSP container. Apache Tomcat version 10.0 implements the Servlet 5.0 and JavaServer Pages 3.0 specifications from Jakarta EE, and includes many additional features that make it a useful platform for developing and deploying web applications and web services.

### b. Development Standards

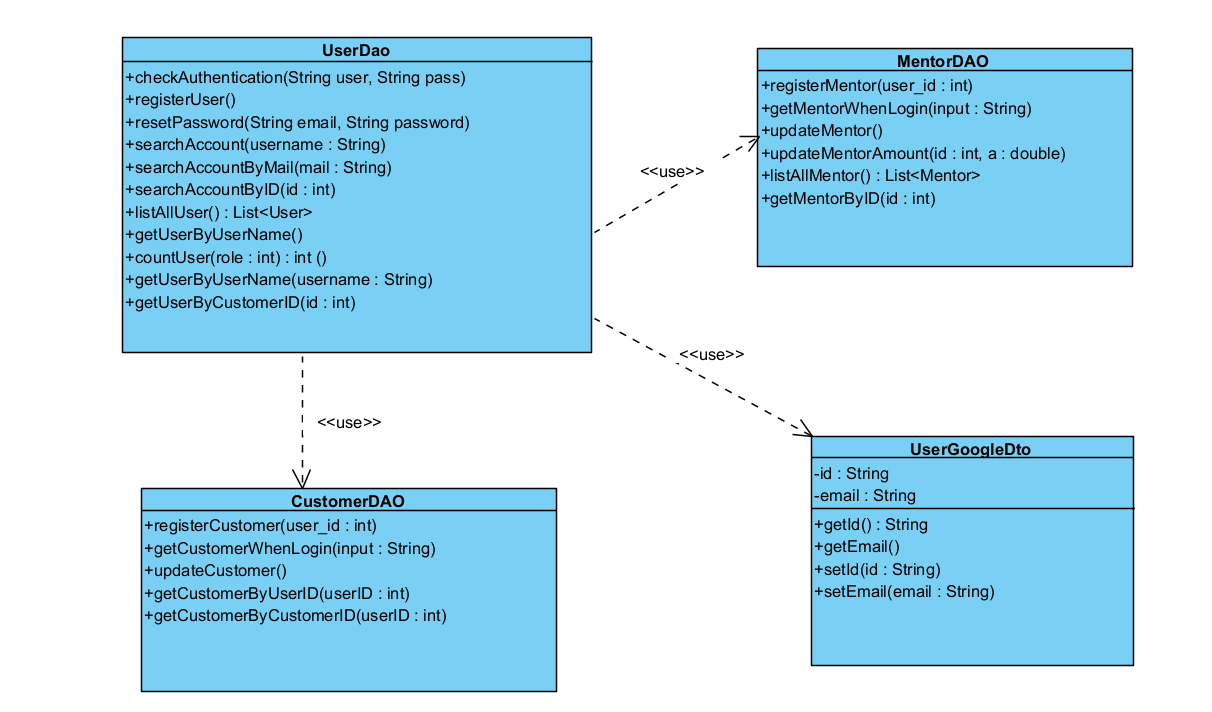
Tick the appropriate box to indicate the standards being followed for this application:

|  |  |
| --- | --- |
| **Standard** | **√ indicates compliance** |
| Database Design | √ |
| Cold Fusion | √ |
| Java | √ |
| Uportal Development | √ |
| Accessibility | √ |
| Web Style Standards | √ |
| Supported Web Browsers | √ |

# III. Code Designs

## 1. Public User

### a. Class Diagram



### b. Class Specifications

#### UserDao.java

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public User checkAuthentication(String user, String pass) | Handle request get Admin by email and password |
| 02 | public boolean registerUser(String username, String password, String fullName, String email, String phone, int role, int Verified, int status) | Handle request get register User |
| 03 | public void resetPassword(String email, String password) | Handle request get reset password |
| 04 | public User searchAccount(String username) | Handle request get account |
| 05 | public User searchAccountByMail(String mail) | Handle request get account by email |
| 06 | public List<User> listAllUser() | Handle request get List User |
| 07 | public int countUser(int role) | Handle request count User |
| 08 | public User getUserByUserName(String username) | Handle request get User by User Name |
| 09 | public void blockUser(int userID) | Handle request block an User |
| 10 | public void unblockUser(int userID) | Handle request unblock an User |
| 11 | public User getStatusByID(int id) | Handle request get status by ID |
| 12 | public String getAvatarbyID(int id) | Handle request get Avatar by ID |

#### CustomerDAO.java

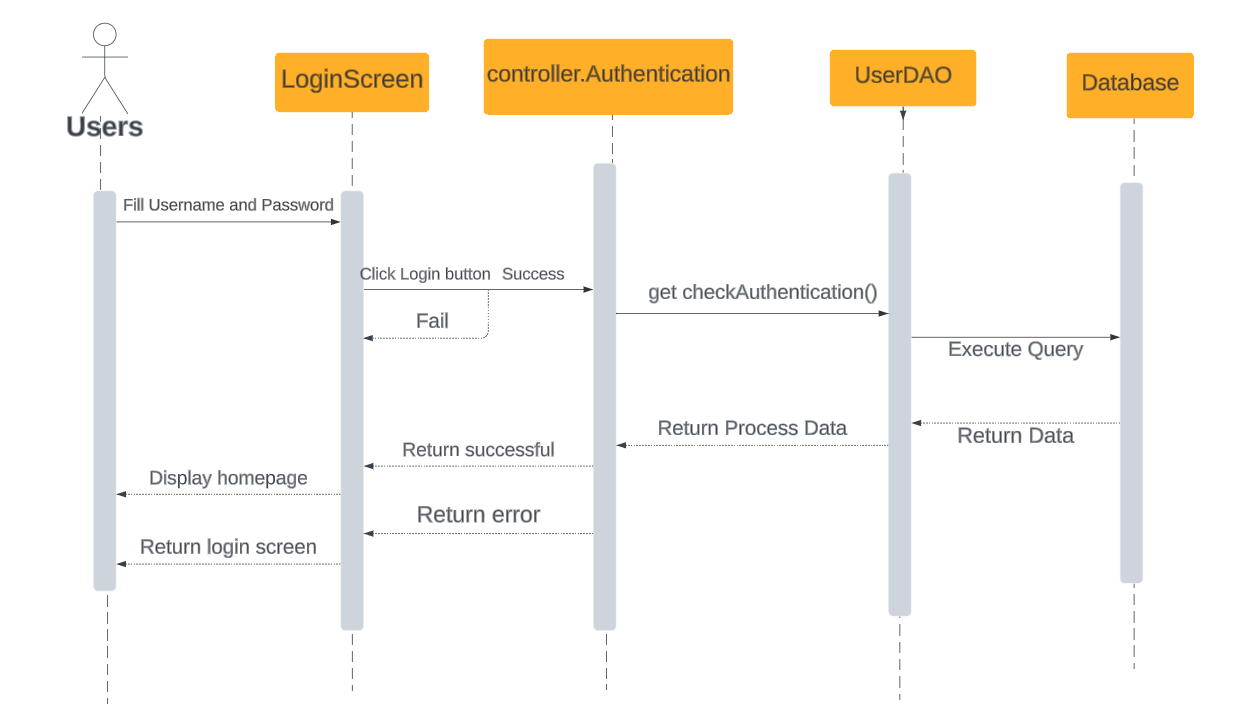
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
|  |  |  |
| 01 | public void registerCustomer(int user\_id) | Handle request get register Customer |
| 02 | public Customer getCustomerWhenLogin(String input) | Handle request get Customer when Login |
| 03 | public void updateCustomer(int id, String phone, String name, String dob, String address, String avatar) | Handle request Update Customer By UserID |
| 04 | public Customer getCustomerByUserID(int userID) | Handle request get Customer by UserID |

#### MentorDAO.java

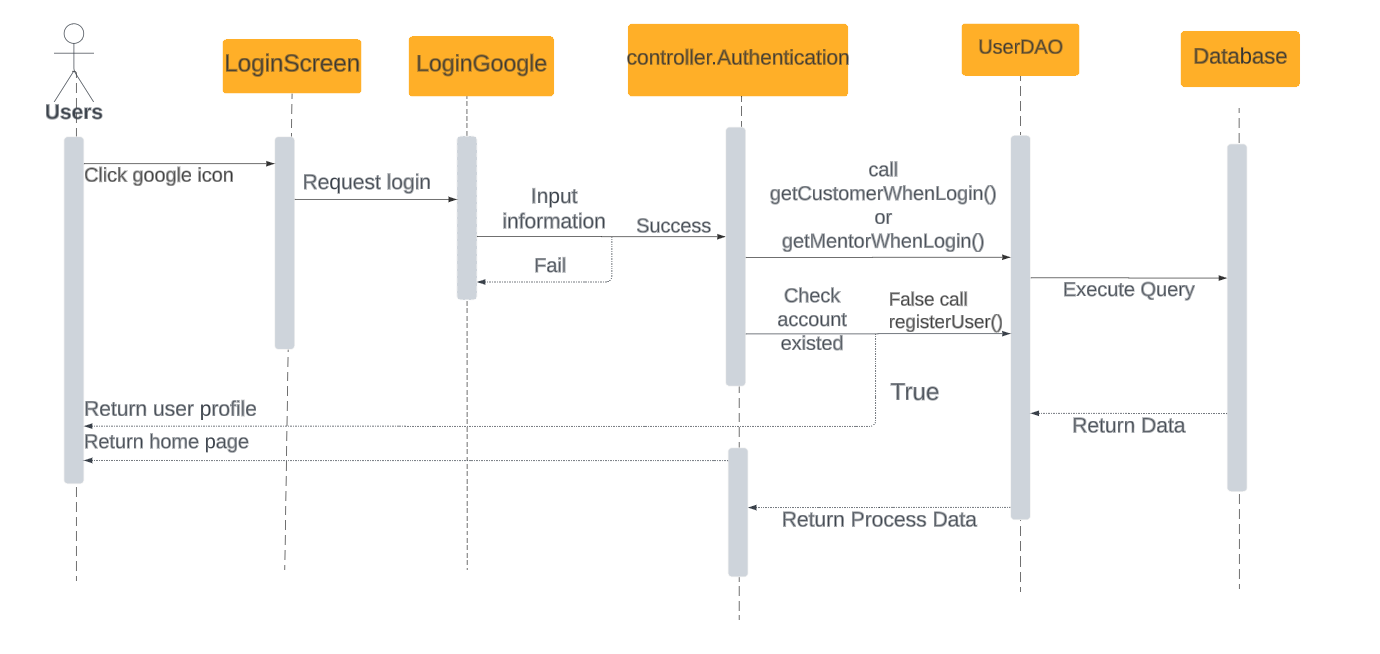
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public void registerMentor(int user\_id) | Handle request get register Mentor |
| 02 | public Mentor getMentorWhenLogin(String input) | Handle request get Mentor when Login |
| 03 | public void updateMentor(int id, String phone, String name, String dob, String skill, String job, String avatar) | Handle request Update Mentor By UserID |
| 04 | public List<Mentor> listAllMentor() | Handle request List All Mentor |
| 05 | public List<Mentor> top3MentorHasMostCourse() | Handle request get top 3 Mentor Has Most Course |
| 06 | public List<Mentor> mentorPage() | Handle request List Mentor in Mentor Page |
| 07 | public Mentor getMentorByID(int id) | Handle request get Mentor by ID |

### c. Sequence Diagram(s)

#### Login

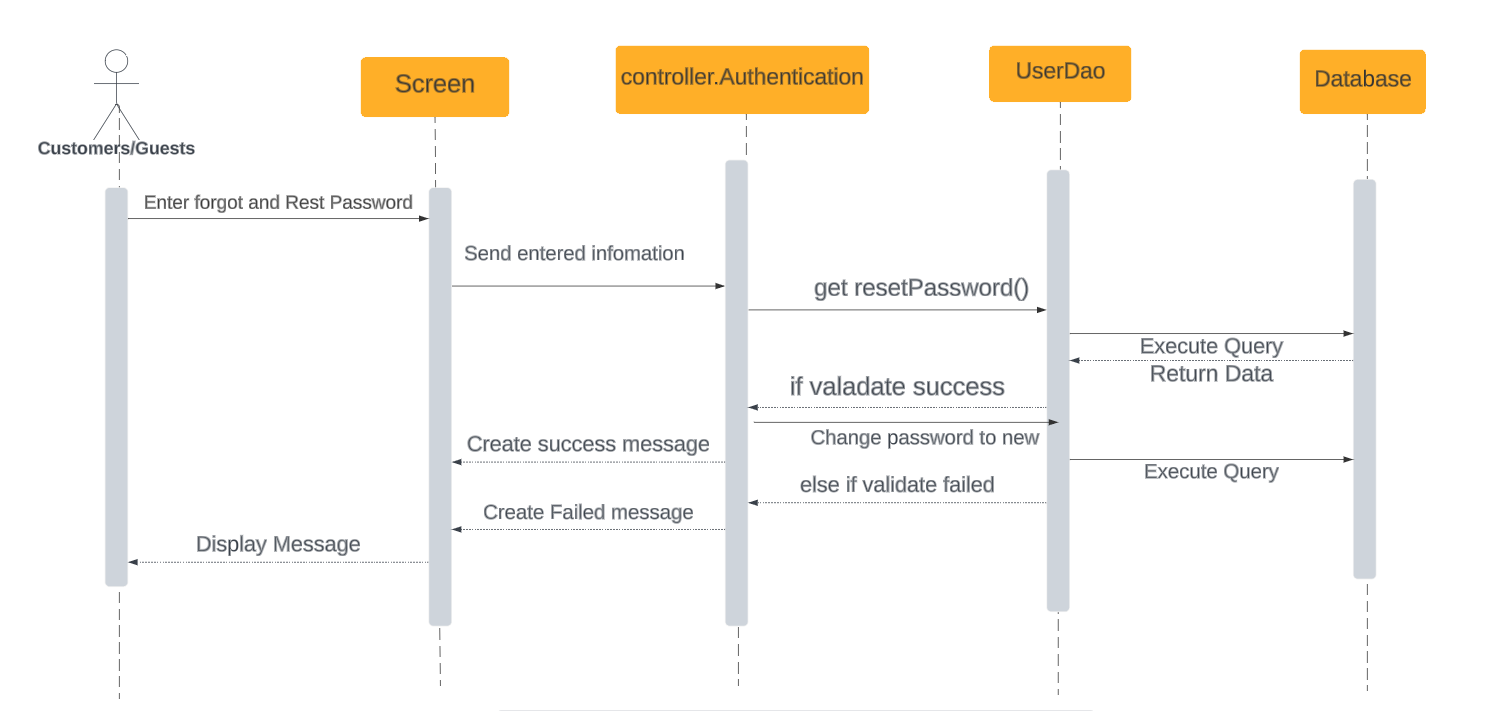


*Login by username and password*

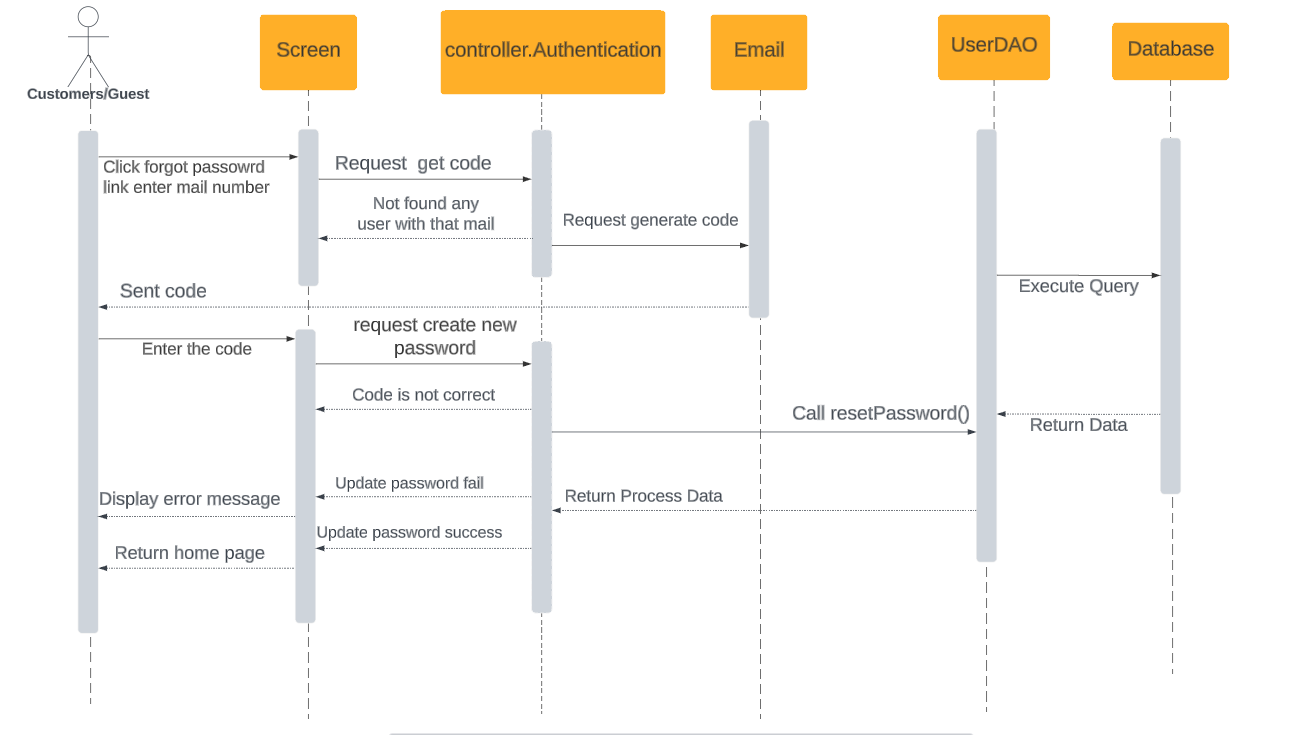
**

*Login with Google Account*

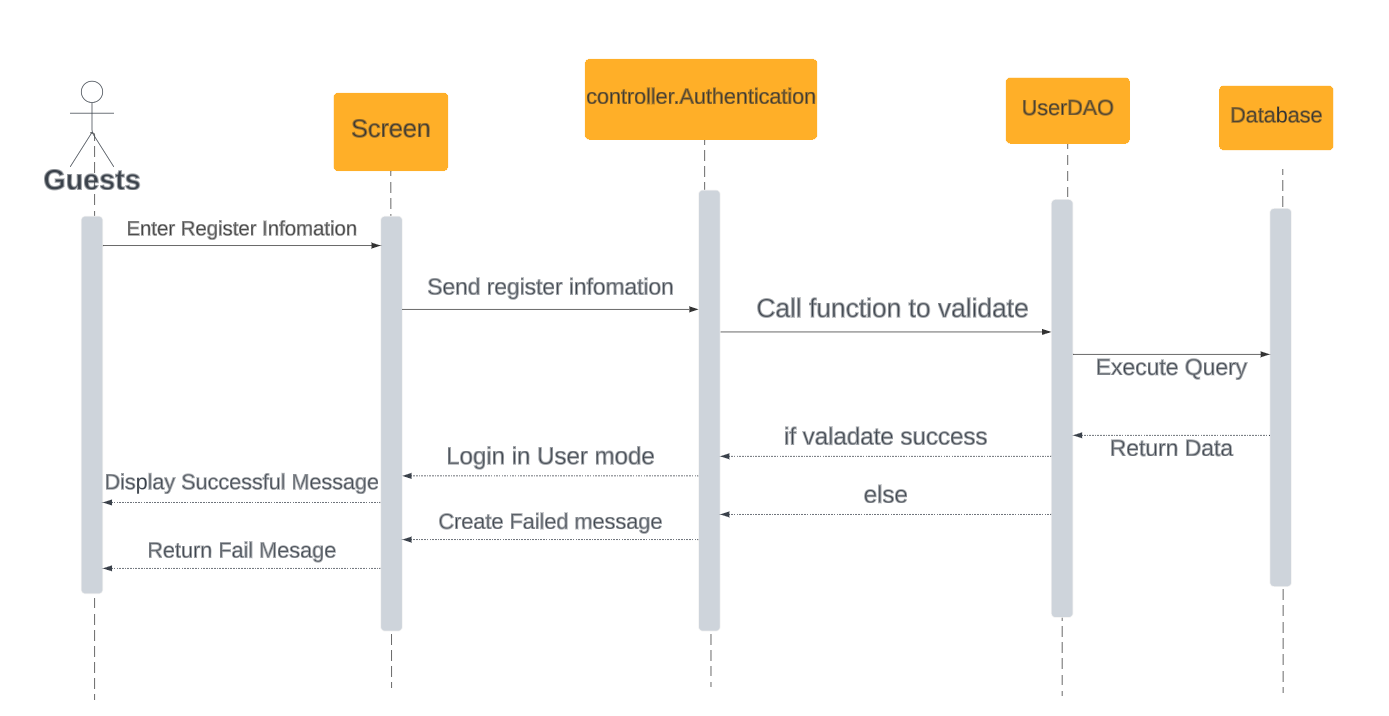
#### Change Password



#### Forgot Password

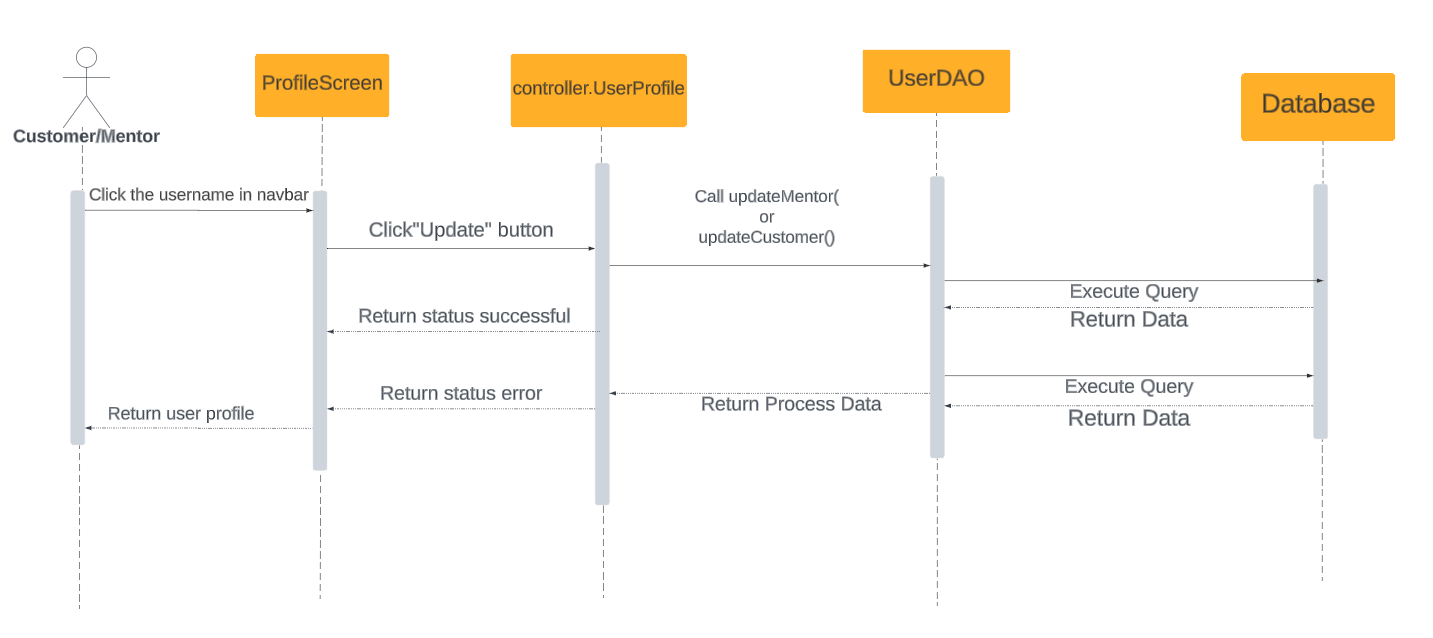


#### Register



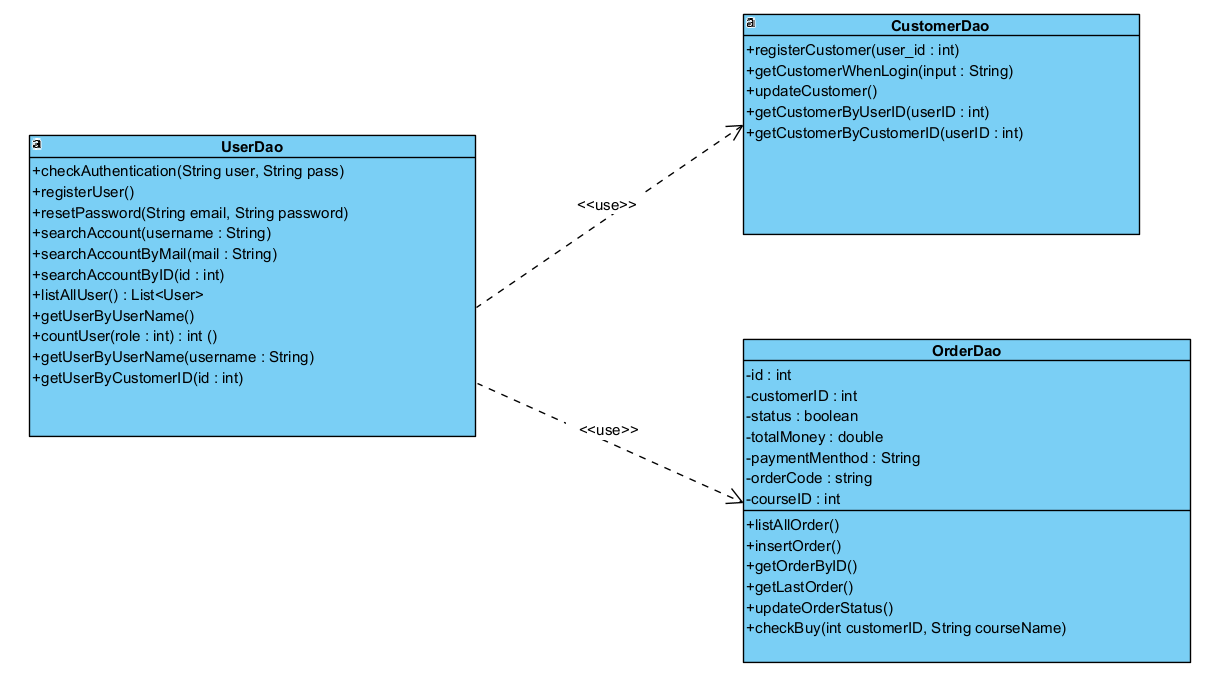
#### Display Profile

#### Update Profile



## 2. Public Order

### a. Class Diagram



### b. Class Specifications

#### UserDao.java

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public User checkAuthentication(String user, String pass) | Handle request get Admin by email and password |
| 02 | public boolean registerUser(String username, String password, String fullName, String email, String phone, int role, int Verified, int status) | Handle request get register User |
| 03 | public void resetPassword(String email, String password) | Handle request get reset password |
| 04 | public User searchAccount(String username) | Handle request get account |
| 05 | public User searchAccountByMail(String mail) | Handle request get account by email |
| 06 | public List<User> listAllUser() | Handle request get List User |
| 07 | public int countUser(int role) | Handle request count User |
| 08 | public User getUserByUserName(String username) | Handle request get User by User Name |
| 09 | public void blockUser(int userID) | Handle request block an User |
| 10 | public void unblockUser(int userID) | Handle request unblock an User |
| 11 | public User getStatusByID(int id) | Handle request get status by ID |
| 12 | public String getAvatarbyID(int id) | Handle request get Avatar by ID |

#### CustomerDAO.java

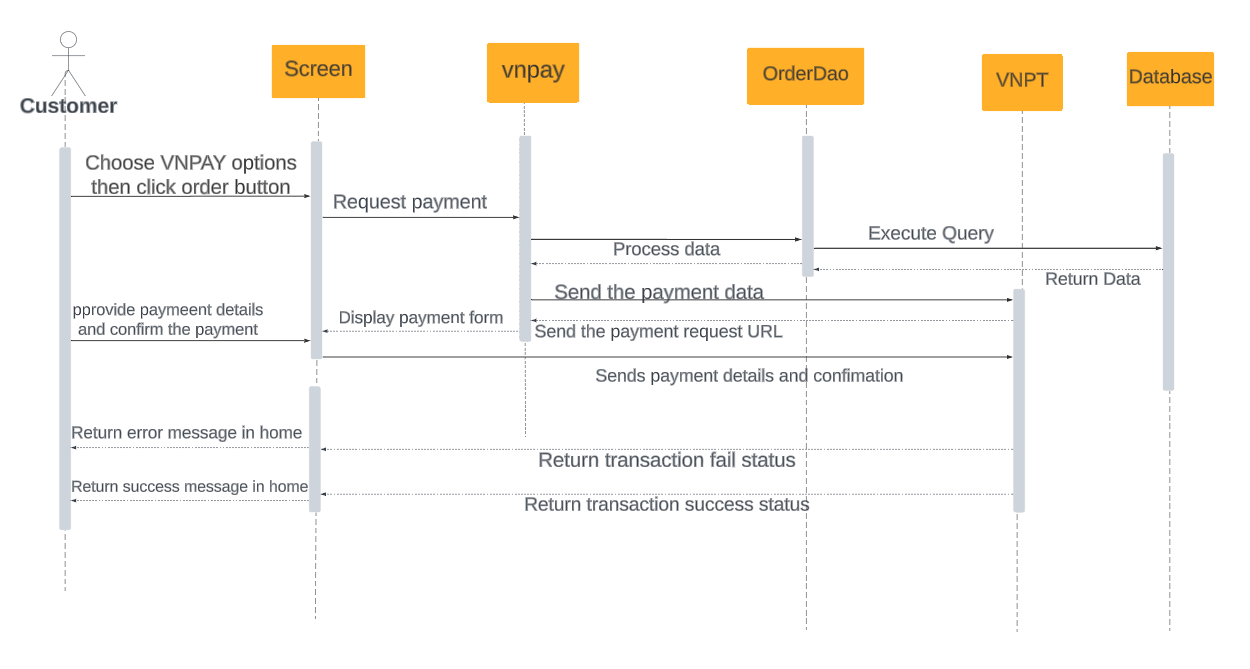
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public void registerCustomer(int user\_id) | Handle request get register Customer |
| 02 | public Customer getCustomerWhenLogin(String input) | Handle request get Customer when Login |
| 03 | public void updateCustomer(int id, String phone, String name, String dob, String address, String avatar) | Handle request Update Customer By UserID |
| 04 | public Customer getCustomerByUserID(int userID) | Handle request get Customer by UserID |

#### OrderDAO.java

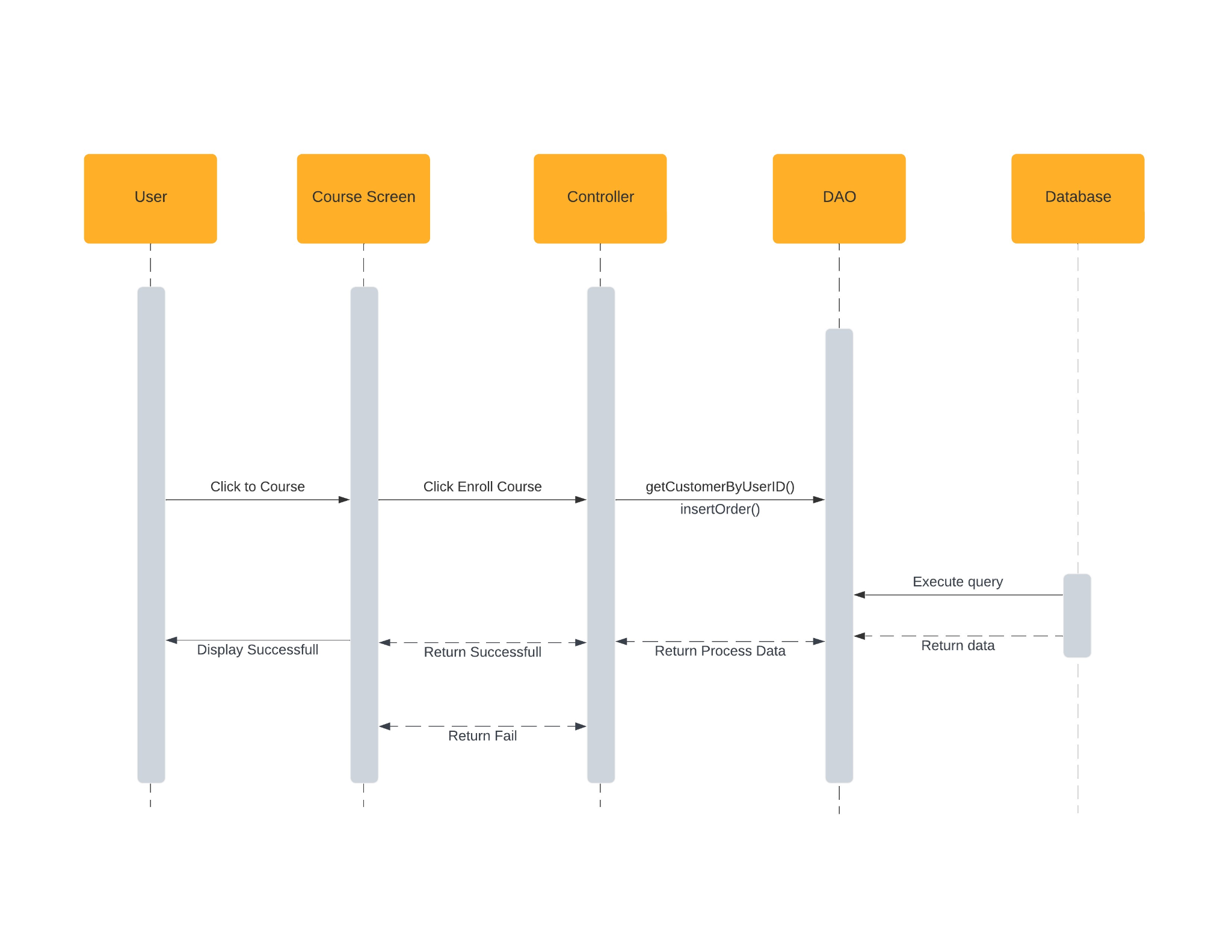
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public List<Order> listAllOrder() | Handle request get List All Order |
| 02 | public void insertOrder(Order order) | Handle request Insert Order |
| 03 | public Order getOrderByID(int orderID) | Handle request get Order by ID |
| 04 | public Order getLastOrder() | Handle request get Last Order |
| 05 | public void updateOrderStatus(int orderID, boolean newStatus) | Handle request Update Order Status |
| 06 | public boolean checkBuy(int customerId, String courseName) | Handle request check Buy |

### c. Sequence Diagram(s)

#### Payment by VNPay

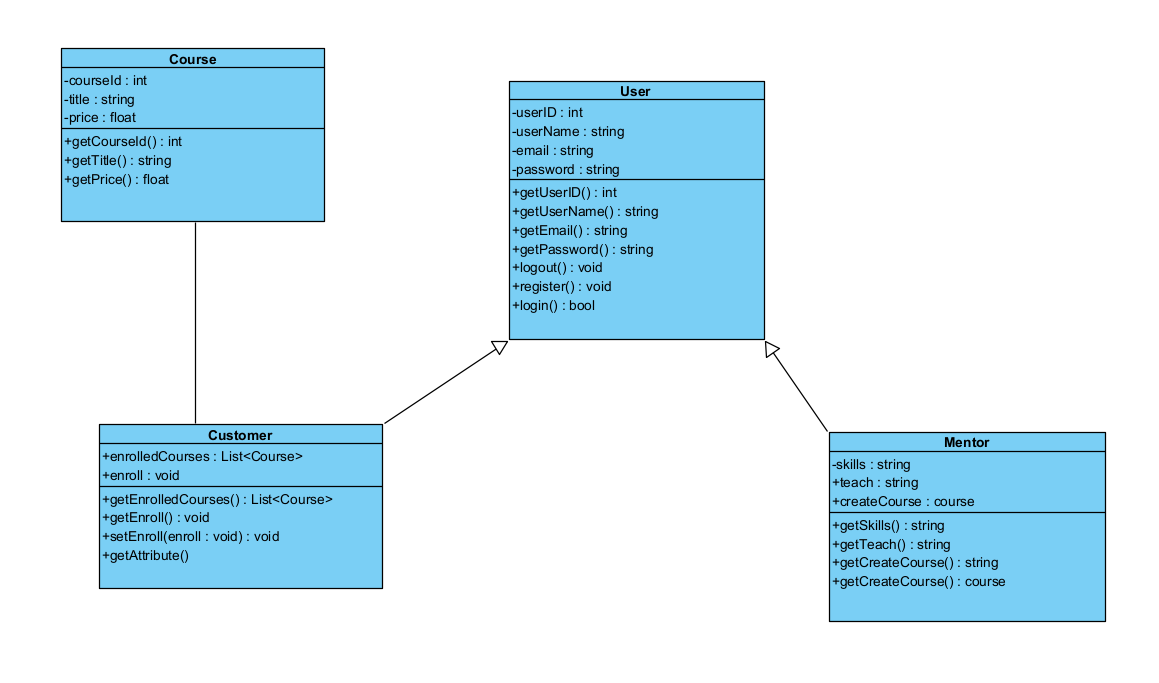


#### Insert Order



## 3. Public Course

### a. Class Diagram



### b. Class Specifications

#### UserDao.java

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public User checkAuthentication(String user, String pass) | Handle request get Admin by email and password |
| 02 | public boolean registerUser(String username, String password, String fullName, String email, String phone, int role, int Verified, int status) | Handle request get register User |
| 03 | public void resetPassword(String email, String password) | Handle request get reset password |
| 04 | public User searchAccount(String username) | Handle request get account |
| 05 | public User searchAccountByMail(String mail) | Handle request get account by email |
| 06 | public List<User> listAllUser() | Handle request get List User |
| 07 | public int countUser(int role) | Handle request count User |
| 08 | public User getUserByUserName(String username) | Handle request get User by User Name |
| 09 | public void blockUser(int userID) | Handle request block an User |
| 10 | public void unblockUser(int userID) | Handle request unblock an User |
| 11 | public User getStatusByID(int id) | Handle request get status by ID |
| 12 | public String getAvatarbyID(int id) | Handle request get Avatar by ID |

#### MentorDAO.java

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public void registerMentor(int user\_id) | Handle request get register Mentor |
| 02 | public Mentor getMentorWhenLogin(String input) | Handle request get Mentor when Login |
| 03 | public void updateMentor(int id, String phone, String name, String dob, String skill, String job, String avatar) | Handle request Update Mentor By UserID |
| 04 | public List<Mentor> listAllMentor() | Handle request List All Mentor |
| 05 | public List<Mentor> top3MentorHasMostCourse() | Handle request get top 3 Mentor Has Most Course |
| 06 | public List<Mentor> mentorPage() | Handle request List Mentor in Mentor Page |
| 07 | public Mentor getMentorByID(int id) | Handle request get Mentor by ID |

#### CourseDAO.java

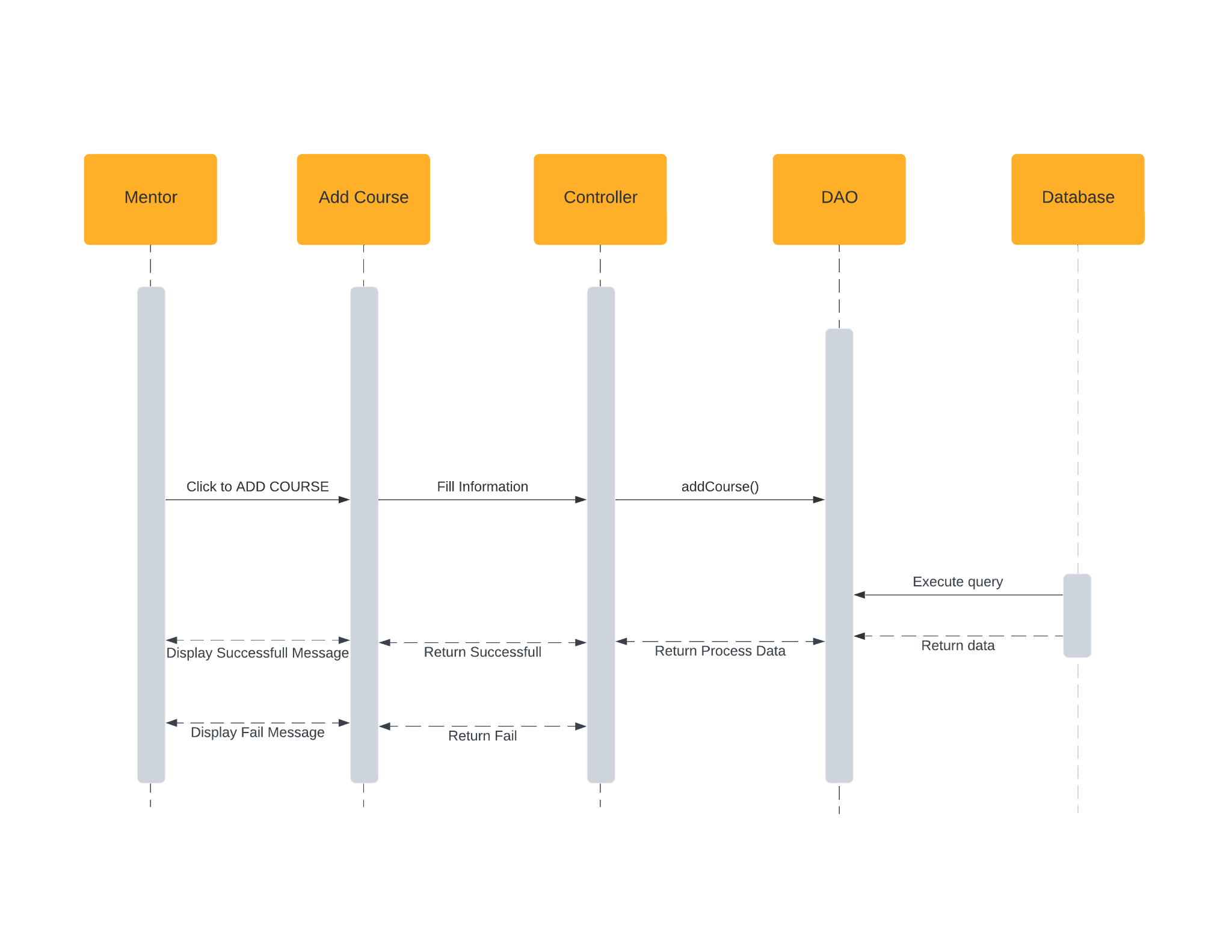
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public List<Course> listAllCourse() | Handle request get List All Course |
| 02 | public List<Course> listAllCourseClient() | Handle request get List All Course Client |
| 03 | public List<Course> get3NewCourse() | Handle request get 3 New Course |
| 04 | public void addCourse(String Course\_name, String Description, int Type\_ID, double price, String Public\_date, int MentorID, String Image, int Status) | Handle request Add Course |
| 05 | public int getLastestCourseID() | Handle request get Lastest Course ID |
| 06 | public void addModule(int courseId, String moduleName, String link) | Handle request Add Module |
| 07 | public int count() | Handle request Count |
| 08 | public List<Course> listCourseByMentor(int mentor\_id) | Handle request List Course By Mentor |
| 09 | public List<Course> listCourseByType(int type\_id) | Handle request get List Course By Type |
| 10 | public List<Course> listCourseClientSite() | Handle request get List Course Client Site |
| 11 | public Course getCourseByID(int course\_id) | Handle request get Course By ID |
| 12 | public void changeStatusType(int type\_id, int status) | Handle request Change Status Type |
| 13 | public List<CourseModule> listModuleByCourseID(int id) | Handle request get List Module By Course ID |
| 14 | public List<Course> listCourseBySearchPart(String txt) | Handle request List Course By SearchPart |
| 15 | public Course getCourseByURL(String url) | Handle request get Course By URL |
| 16 | public String convertYouTubeLinkToEmbed(String link) | Handle request Convert Youtube Link To Embed |

#### CourseTypeDAO.java

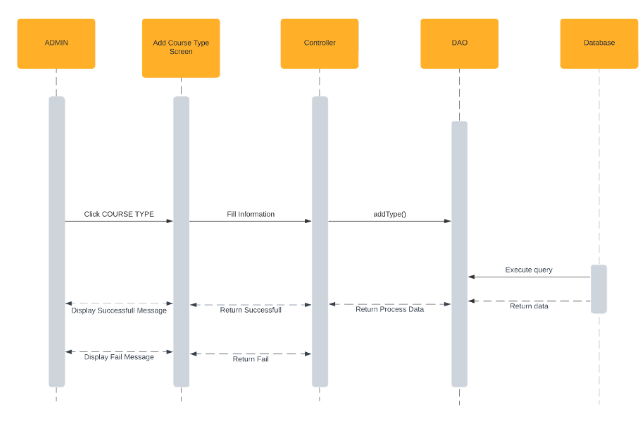
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public List<CourseType> listAllCourseType() | Handle request get List All Course Type |
| 02 | public List<CourseType> listCourseTypeClientSite() | Handle request get List Course Type Client Site |
| 03 | public int count() | Handle request Count |
| 04 | public void addType(String type, String created\_date, String description) | Handle request Add Type |
| 05 | public CourseType getTypeByName(String type) | Handle request get Type By Name |
| 06 | public void updateCourseTypeByID(int id, String name, String dateUpdate, String description) | Handle request Update Course Type By ID |
| 07 | public CourseType getTypeByID(int id) | Handle request get Type By ID |
| 08 | public void changeStatusType(int type\_id, int status) | Handle request change Status Type |
| 09 | public CourseType getStatusByID(int id) | Handle request get Status By ID |

### c. Sequence Diagram(s)

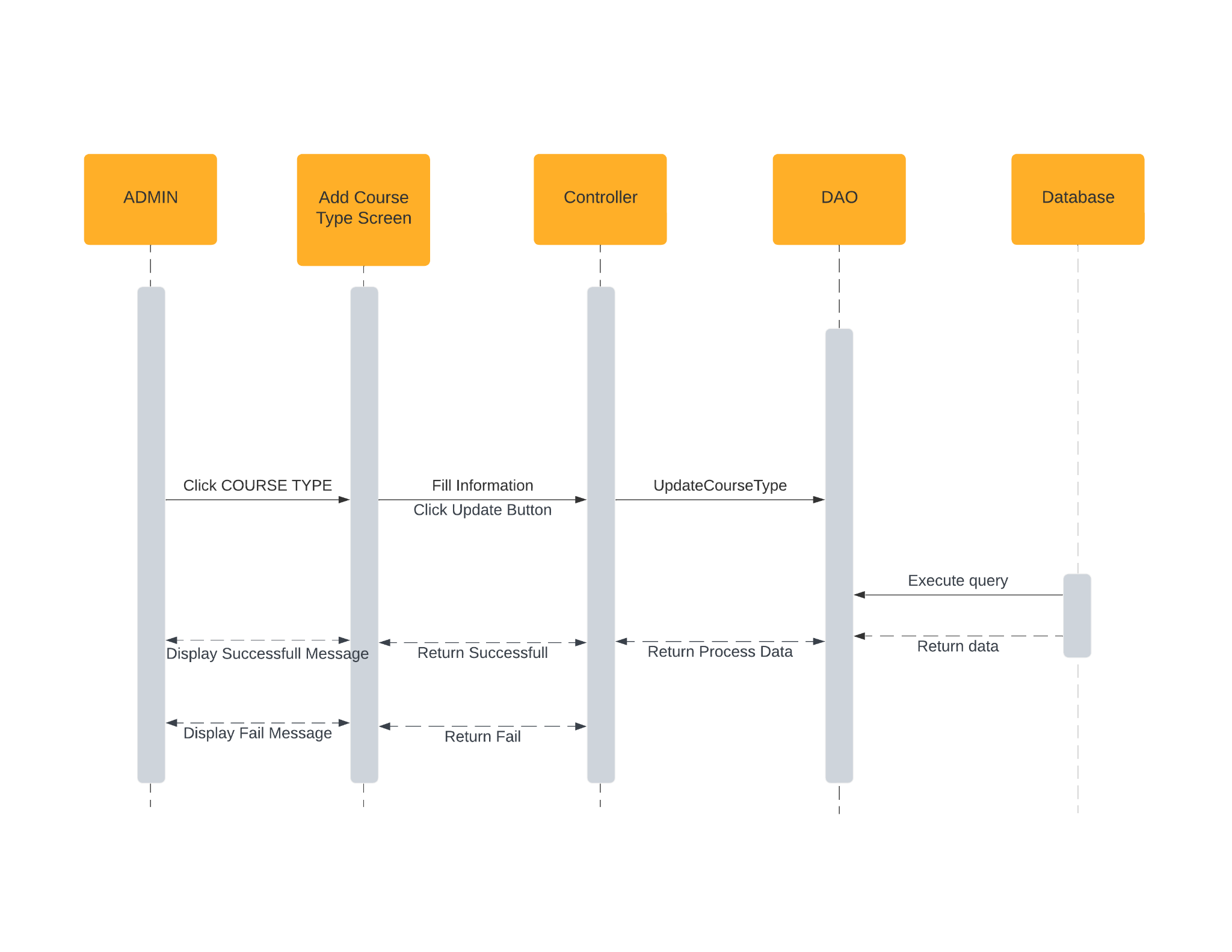
#### Add Course



#### Add CourseType

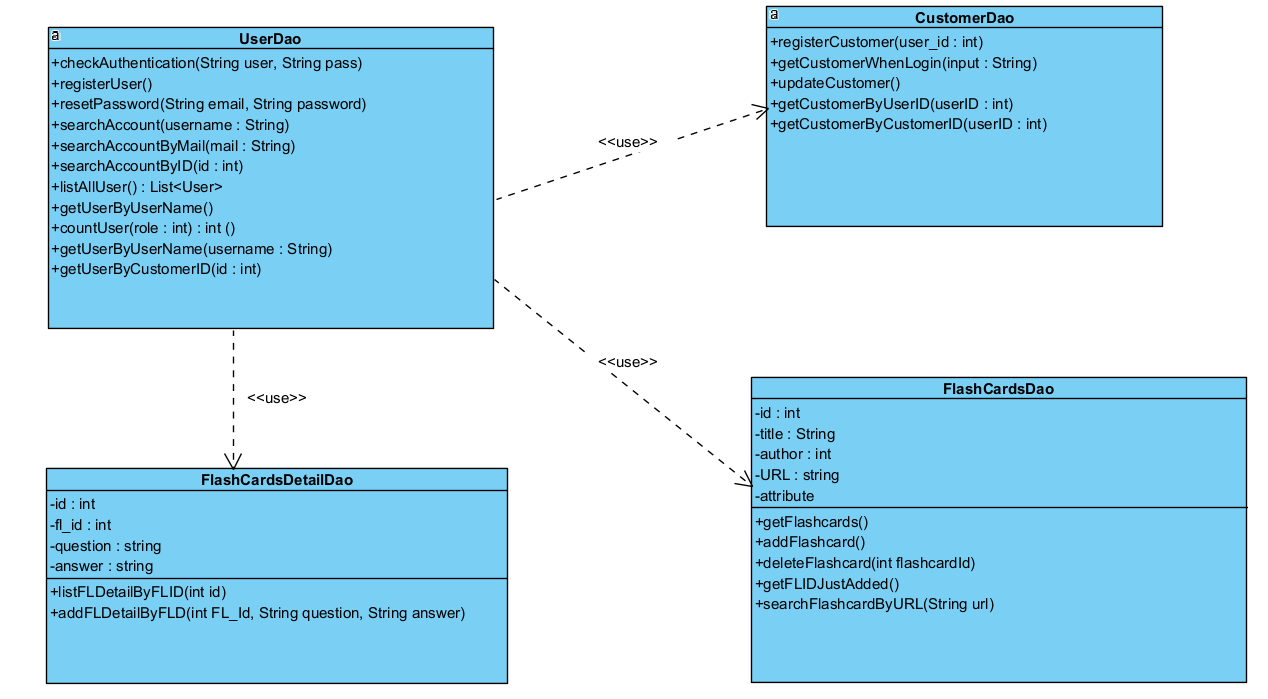


#### Update CourseType



## 4. Public FlashCards

### a. Class Diagram



### b. Class Specifications

#### UserDao.java

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public User checkAuthentication(String user, String pass) | Handle request get Admin by email and password |
| 02 | public boolean registerUser(String username, String password, String fullName, String email, String phone, int role, int Verified, int status) | Handle request get register User |
| 03 | public void resetPassword(String email, String password) | Handle request get reset password |
| 04 | public User searchAccount(String username) | Handle request get account |
| 05 | public User searchAccountByMail(String mail) | Handle request get account by email |
| 06 | public List<User> listAllUser() | Handle request get List User |
| 07 | public int countUser(int role) | Handle request count User |
| 08 | public User getUserByUserName(String username) | Handle request get User by User Name |
| 09 | public void blockUser(int userID) | Handle request block an User |
| 10 | public void unblockUser(int userID) | Handle request unblock an User |
| 11 | public User getStatusByID(int id) | Handle request get status by ID |
| 12 | public String getAvatarbyID(int id) | Handle request get Avatar by ID |

#### CustomerDAO.java

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public void registerCustomer(int user\_id) | Handle request get register Customer |
| 02 | public Customer getCustomerWhenLogin(String input) | Handle request get Customer when Login |
| 03 | public void updateCustomer(int id, String phone, String name, String dob, String address, String avatar) | Handle request Update Customer By UserID |
| 04 | public Customer getCustomerByUserID(int userID) | Handle request get Customer by UserID |

#### FlashCardsDAO.java

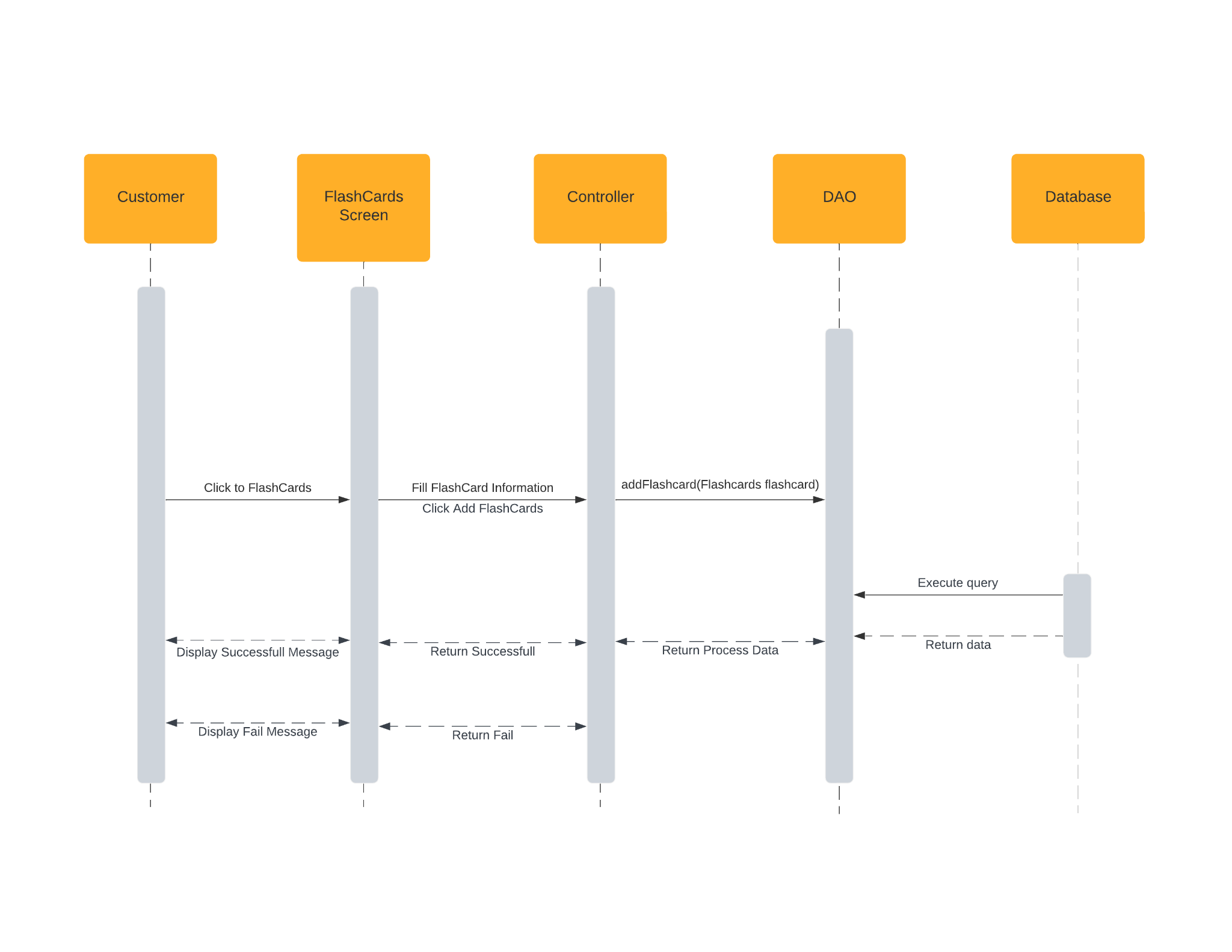
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public List<Flashcards> getFlashcards() | Handle request get FlashCards |
| 02 | public void addFlashcard(Flashcards flashcard) | Handle request Add FlashCards |
| 03 | public void deleteFlashcard(int flashcardId) | Handle request Delete FlashCards |
| 04 | public int getFLIDJustAdded() | Handle request get FlashCards ID just Added |
| 05 | public Flashcards searchFlashcardByURL(String url) | Handle request search FlashCard By URL |

#### FlashCardDetailDAO.java

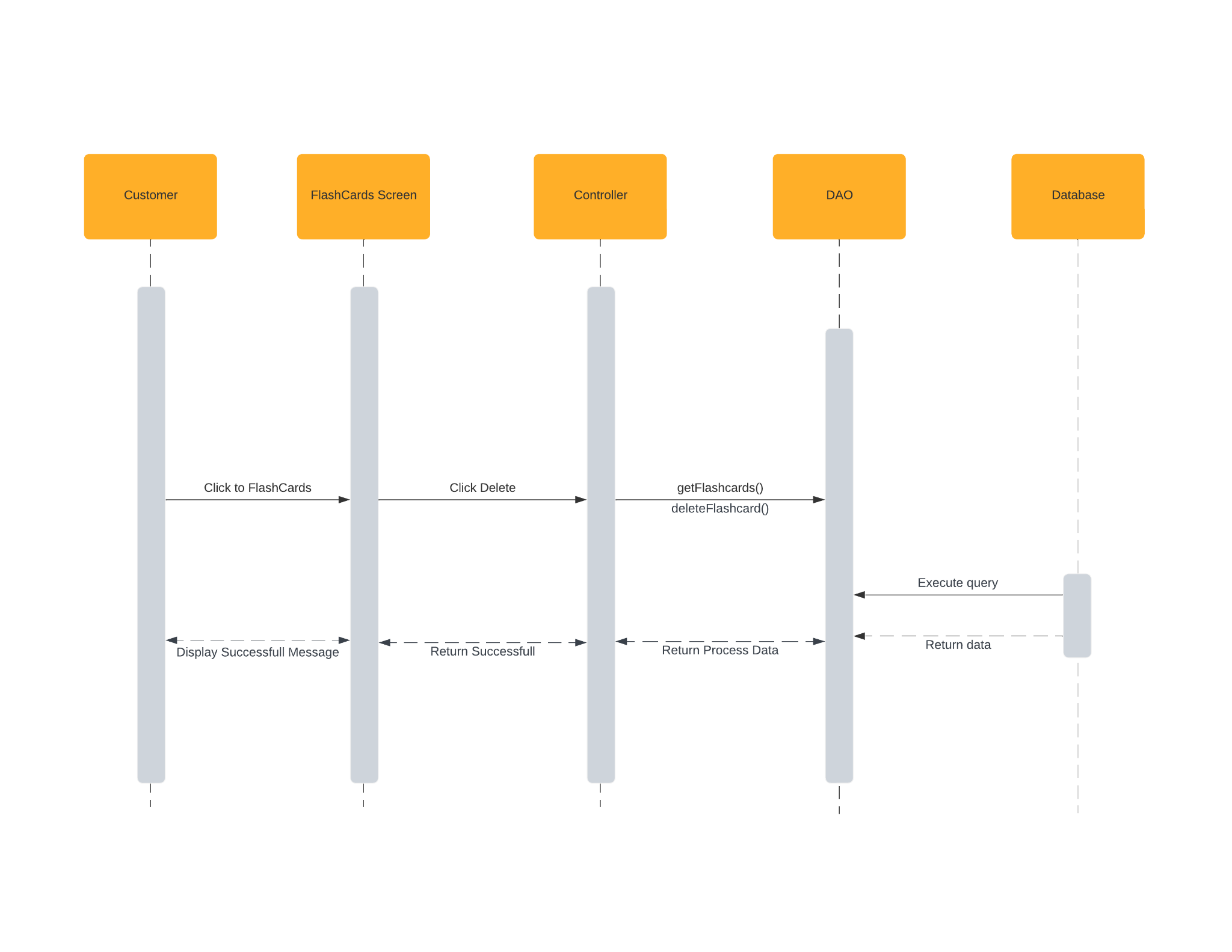
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public List<FlashCardDetail> listFLDetailByFLID(int id) | Handle request get list FlashCard Detail by FlashCard ID |
| 02 | public void addFLDetailByFLID(int Fl\_Id, String question, String answer) | Handle request add FlashCard Detail By FlashCard ID |

### c. Sequence Diagram(s)

#### Add FlashCards



#### Delete FlashCards



## 5. Public News

### a. Class Diagram

### b. Class Specifications

#### UserDao.java

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public User checkAuthentication(String user, String pass) | Handle request get Admin by email and password |
| 02 | public boolean registerUser(String username, String password, String fullName, String email, String phone, int role, int Verified, int status) | Handle request get register User |
| 03 | public void resetPassword(String email, String password) | Handle request get reset password |
| 04 | public User searchAccount(String username) | Handle request get account |
| 05 | public User searchAccountByMail(String mail) | Handle request get account by email |
| 06 | public List<User> listAllUser() | Handle request get List User |
| 07 | public int countUser(int role) | Handle request count User |
| 08 | public User getUserByUserName(String username) | Handle request get User by User Name |
| 09 | public void blockUser(int userID) | Handle request block an User |
| 10 | public void unblockUser(int userID) | Handle request unblock an User |
| 11 | public User getStatusByID(int id) | Handle request get status by ID |
| 12 | public String getAvatarbyID(int id) | Handle request get Avatar by ID |

#### NewsDao.java

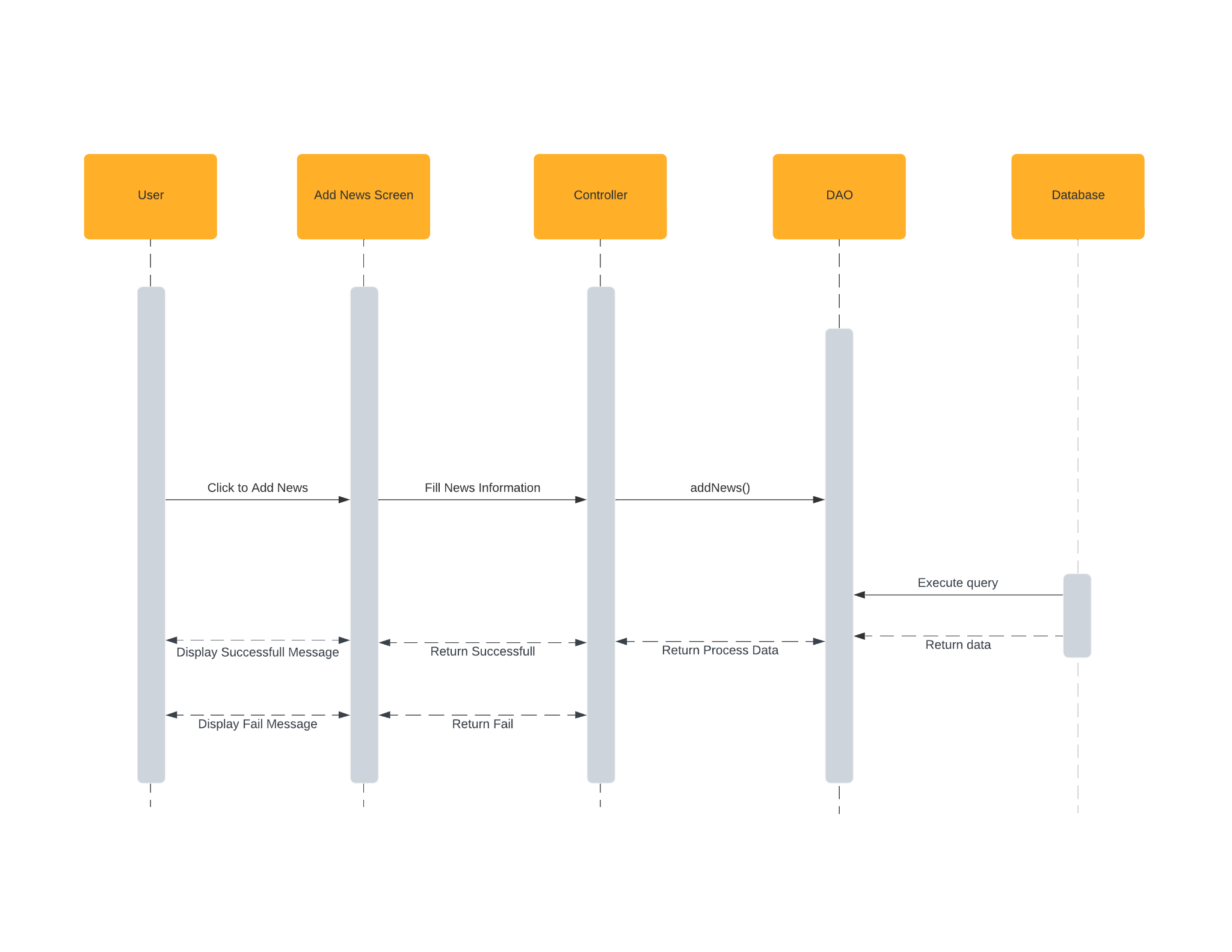
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public void updateNewsByID(int id, String title, String image, String newsContent, int type) | Handle request Update News By ID |
| 02 | public List<News> listAllNews() | Handle request get List All News |
| 03 | public int addNews(String title, String image, String createDate, String newsContent, String createBy, int type) | Handle request Add News |
| 04 | public List<News> listAllNewsByMentor(String username) | Handle request get List All News By Mentor |
| 05 | public List<News> listNewsAccepted(int status, int type) | Handle request get List News Accepted |
| 06 | public News getStatusByID(int id) | Handle request get Status News By ID |
| 07 | public void changeStatusType(int newsID, int status) | Handle request change Status News |
| 08 | public News getNewsByID(int id) | Handle request get News By ID |
| 09 | public News getNewsByURL(String url) | Handle request get News By URL |
| 10 | public List<News> listNewsRandom(int status, int type) | Handle request get List News Random |

#### NewsTypeDao.java

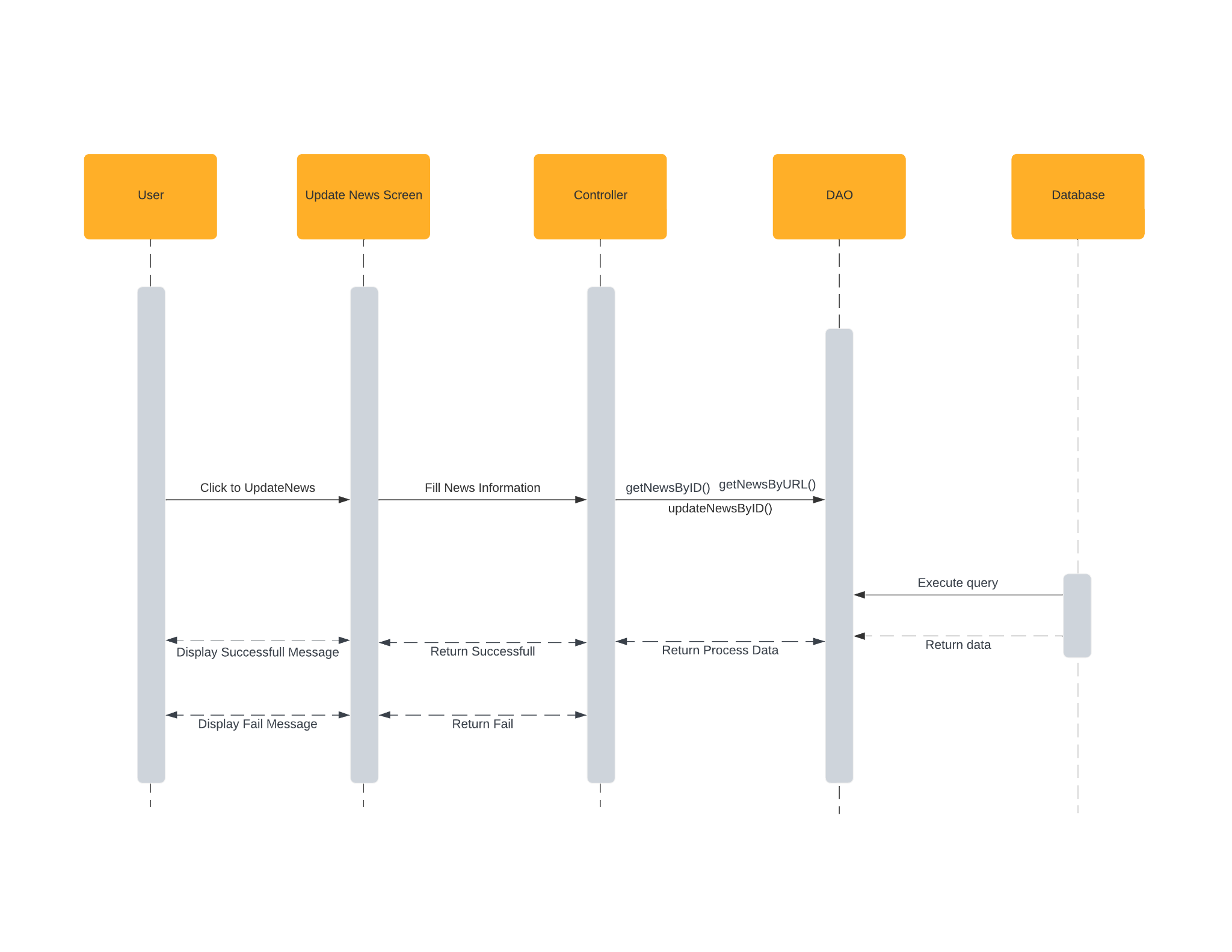
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public List<NewsType> listAllNewsType() | Handle request get List All News Type |
| 02 | public NewsType getTypeByID(int id) | Handle request get Type By ID |
| 03 | public void updateNews(String type, String description, int id) | Handle request Update News |
| 04 | public void addNews(String type, String description) | Handle request Add News Type |
| 05 | public NewsType checkTypeExist(String type) | Handle request check News Type Exist |

### c. Sequence Diagram(s)

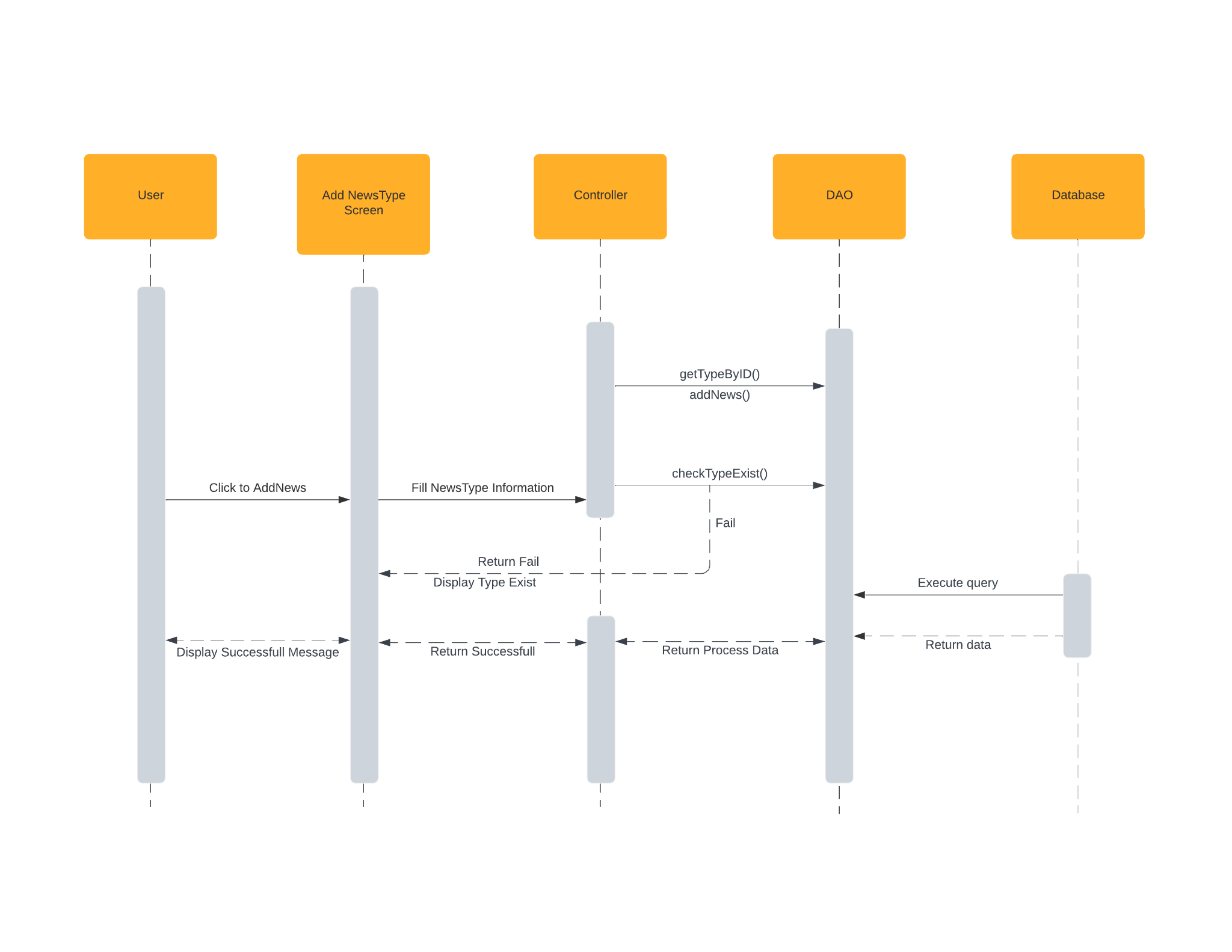
#### AddNews



#### Update News



#### Add NewsType



#### Update NewsType

