

---

# Architecture Document

**SWEN90007**

## Online Examination System

Team: A Team

In charge of:

Student Name	Username	Student Id	Email
Zihan Zhang	zhazz	963790	zhazz@student.unimelb.edu.au
Chengeng Liu	chengengl	813174	chengengl@student.unimelb.edu.au
Hao Ding	dingh2	825702	dingh2@student.unimelb.edu.au

---



---

SCHOOL OF  
**COMPUTING &  
INFORMATION  
SYSTEMS**

---

**Revision History** (this template's)

Date	Version	Description	Author
25/10/20	01.00-D01	Initial draft	Hao Ding
28/10/20	02.00-D01	Updated Concurrency	Hao Ding
29/10/20	03.00-D01	Added diagrams	Hao Ding

## Contents

### Table of Contents

<b>1. Introduction</b>	5
1.1 Proposal	5
1.2 Target Users	5
1.3 Target Users	5
<b>2. Architectural representation</b>	5
2.1 Class Diagram:	7
<b>3. Architectural Patterns</b>	7
Examples of Use:	8
3.2 < Authorization Enforcer>	9
Examples of Use:	9
3.3 < Secure Pipe >	10
Examples of Use:	10
3.4 < Pessimistic offline lock pattern >	11
<b>4. Source Code Directories Structure</b>	13
<b>5. Testing Instructions</b>	14

# 1. Introduction

---

## 1.1 Proposal

This document illustrates the high-level architecture of the online examination system, which can be seen from the class diagram. Moreover, this document records the implementations and justifications of the architectural patterns used to solve the concurrency and security issues.

## 1.2 Target Users

This document is mainly intended for SWEN90007 students and teaching team. In addition, the developing team and the system administrator are also encouraged to use this document as a reference to the system,

## 1.3 Target Users

Conventions, terms and abbreviations

This section explains the concept of some important terms that will be used throughout this document. These terms are detailed alphabetically in the following table.

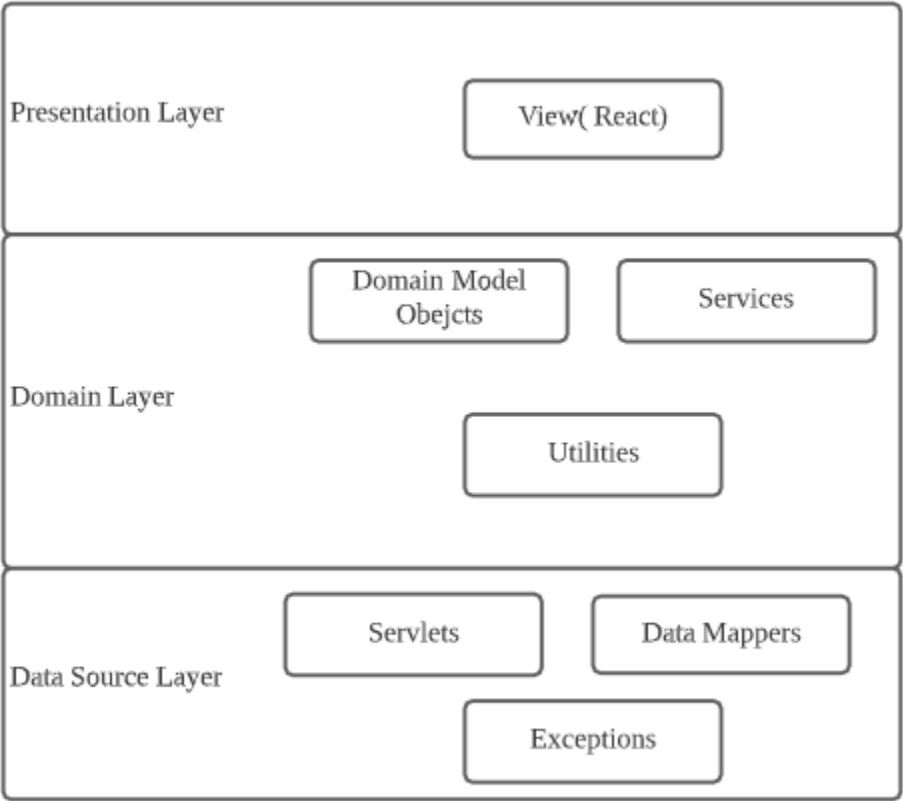
Term	Description
LMS	Learning Management System
SSL	Secure Sockets Layer

# 2. Architectural representation

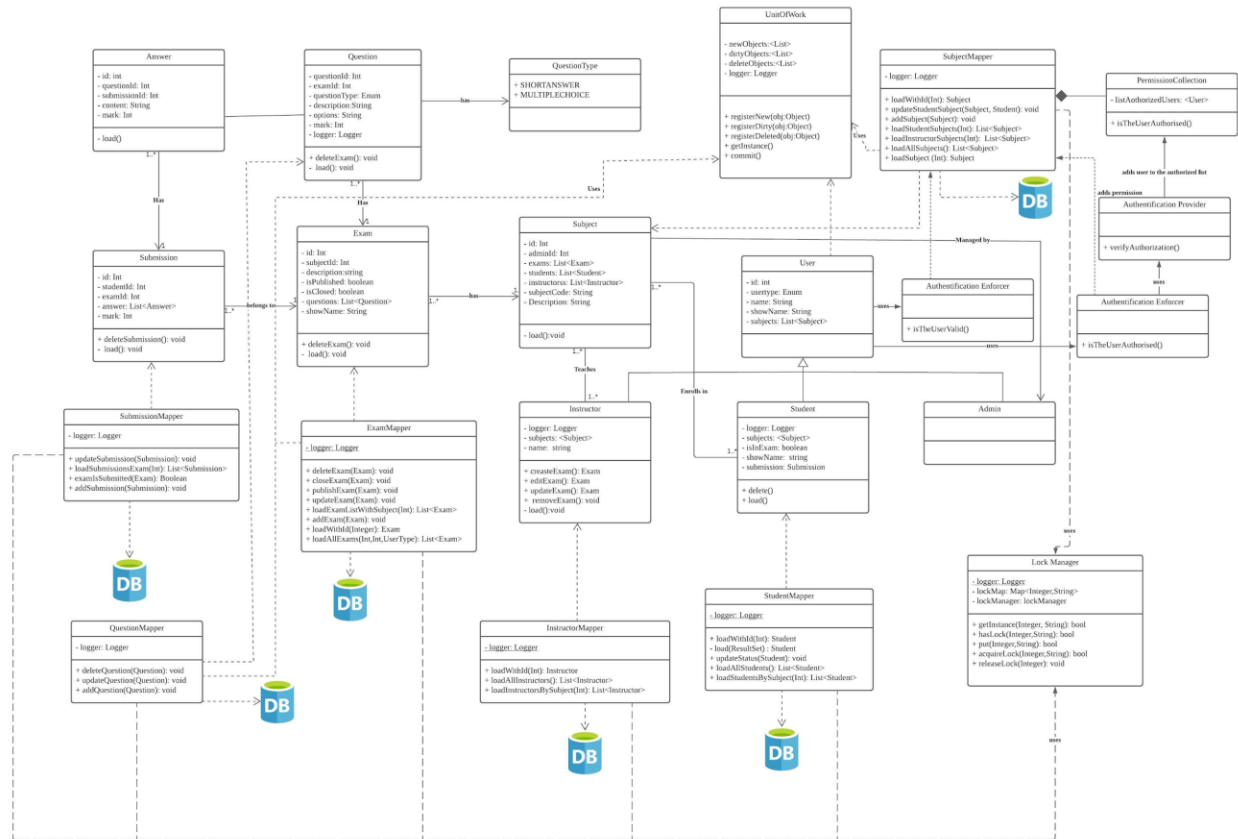
---

We designed the system according to three layers: presentation layer, domain layer, and data source layer.

- The presentation layer displays the system to the users, it is the interface between user interactions and the system. We used React as the frontend framework. All the user inputs are sent from here to the lower layers.
- The domain layer contains all the objects as well as their business logic. Furthermore, all the objects to relational behavior patterns are implemented in this layer. Although this layer is not connected with the database directly, we utilized the service and utility components to track the modifications on objects before sending the requests to the data source layer.
- The data source layer controls the requests sent in and out from the database. Only the authenticated and authorized requests will be passed to the lower layers. We used the data mapper as the gateway to manage the requests sent from the higher layers, this layer maps the domain layer objects to the corresponding tables and rows in the database. Nevertheless, all the objects to relational structural patterns are implemented in this layer.



## 2.1 Class Diagram:



## 3. Architectural Patterns

This section illustrates all the architectural patterns we implemented in Part 3 of the online examination system; we also gave the justifications for the intentions to use the patterns. Furthermore, we provided one example of use for each pattern and a sequence diagram is accompanied with each pattern to visual the example of use.

We can divide the patterns into two categories: security patterns and concurrency patterns.

The security patterns consist of Authentication Enforcer, Authorization Enforcer and the Secure Pipe patterns. Whereas the Pessimistic Offline Lock pattern with Read Lock is implemented to handle concurrent transactions.

### 3.1 < Authentication Enforcer>

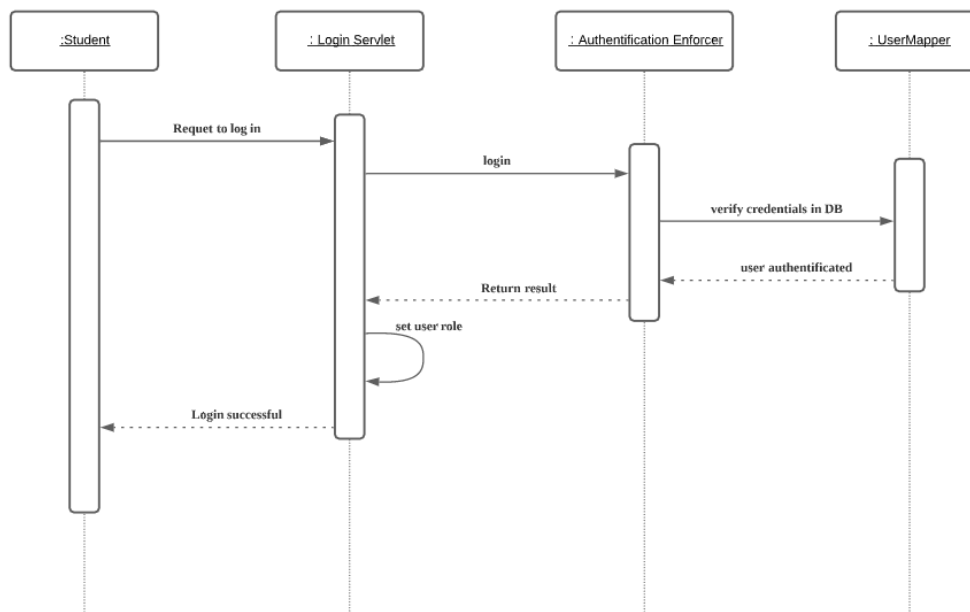
Pattern	Reason
Authentication Enforcer	The server can potentially receive HTTP requests sent from any clients. If the server accepts all the incoming requests, the server and the database information may leak or even be hacked. Therefore, we must ensure the end client is an authenticated entity, and only the right users are allowed to access the online examination system. This pattern only enhances security, but also increases maintainability. In addition, even with the authentication algorithms implemented, we have to write authentication codes for every functionality. Clearly, this approach is not scalable and manageable. Therefore, we could route all the requests to a single verifier, the centralized authentication enforcer would make the codes cleaner and reusable.

Implementation:

We implemented the authentication enforcer pattern when a user requests to sign in. The Shiro library is utilized to implement the underlying authentication mechanisms. First of all, the login servlet passes the user's credentials to the authentication enforcer, and then a token is generated from the credentials. The token is then sent to the database via the user mapper, if there exist the corresponding user details, this request is valid. The user information is fetched and returned with a successful message. Otherwise, the login is rejected, and the authentication fails.

Examples of Use:

Student Login into the system





### 3.2 < Authorization Enforcer>

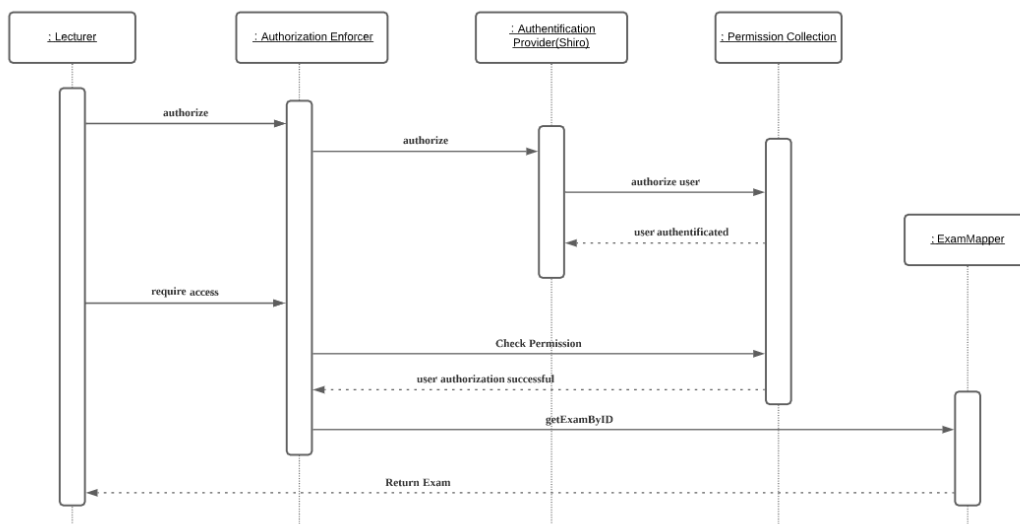
Pattern	Reason
Authorization Enforcer	There are different types of users in this online examination system, each user category has different restrictions. Even if an user is authenticated, it does not guarantee the user will have full accessibility to the system. For example, students are not allowed to create any exams. Therefore, we must ensure users can only get access to functionalities that are available to them. Furthermore, in order to avoid repeating authorizations codes in multiple places, we can have a centralized Authorization Enforcer class to handle all the authorizations. This approach not only increases maintainability of the authorization policies, but also separates the responsibilities between authorization and authentication.

#### Implementation:

We implemented the authorization enforcer pattern whenever there is a communication between the user and the server. For example, after a student logging in, all the student's requests are sent to a centralized authorization enforcer. The enforcer verifies the student's identity and determines whether the request should be granted. Moreover, the authorization enforcer pattern must be collaborating with the authentication enforcer pattern, if the request is verified and authorized, the request will be forwarded to the database via the mapper. On the other hand, if a student sends a POST request to modify an exam, the authorization will fail.

#### Examples of Use:

After successfully logged in, when a lecturer requires to modify an existing exam, the authorization enforcer first checks whether the lecture has the permission. If the lecturer is authorized, he/she will be added to the permission list. Therefore, the request can be passed to the database via the exam mapper, the exam will be returned to the lecturer for modifying.



### 3.3 < Secure Pipe >

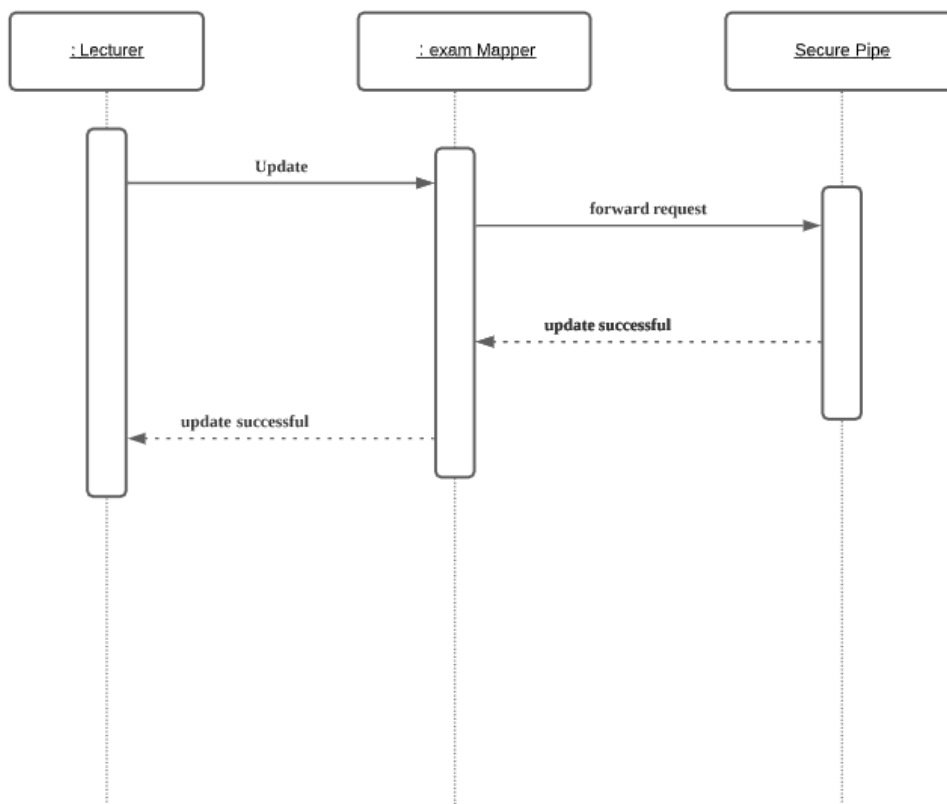
Pattern	Reason
Secure Pipe	<p>Even with strong authentication and authorization mechanisms implemented, the online examination system is not safe yet due to a lack of secure communication channels. For instance, attackers can use SQL injection to manipulate the data transferred between the end-users and the server. Therefore, we need to establish a secure medium by using cryptography to ensure confidentiality.</p> <p>Therefore, the secure pipe pattern can be applied here to prevent information leakage and increase system integrity.</p>

#### Implementation:

The online examination system will be using HTTPS to transfer any data between all the users and the server. Since the system is deployed on Heroku, which automatically includes SSL(secure sockets layer). Therefore, we do not need to implement any additional secure pipe to facilitate data security and integrity.

#### Examples of Use:

After modifying an existing exam, the lecturer commits the change. The request will be send to the database via the secure pipe channel, and the database will return a formation via the secure channel tpp.



### 3.4 < Pessimistic offline lock pattern >

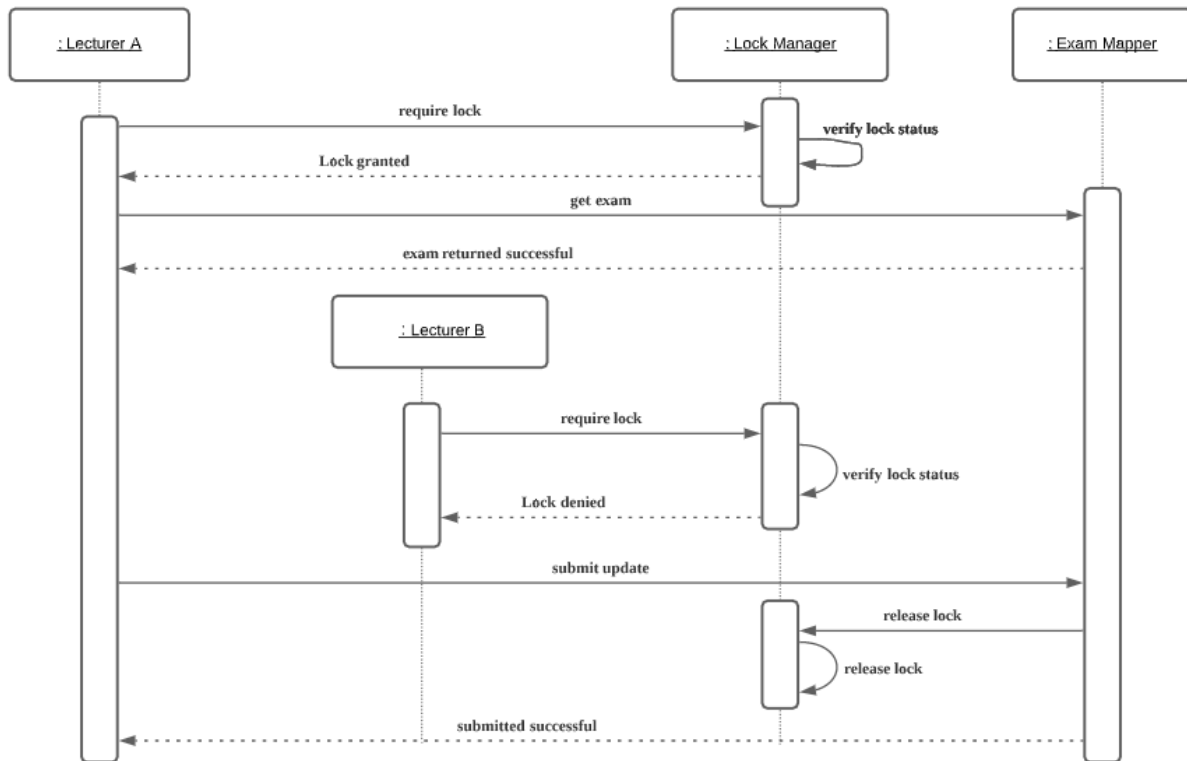
Pattern	Reason
Pessimistic offline lock	In real life, multiple lecturers may be updating the same exam simultaneously. The cost of conflicts can potentially be significant, for example, Lost Update can happen when one lecturer's transaction is overridden by another. Therefore, with the assistance of the pessimistic offline lock pattern, only one lecturer can update one exam at once, the lecture will be notified whether the update is successful. This pattern also ensures the correctness of the system. On the other hand, although the optimistic offline lock pattern ensures the liveness of the system, once an exam is created or marked, it is unlikely to be changed many times. Thus, liveness is not a big concern here because the concurrent transaction situations are rare. Furthermore, the optimistic offline lock pattern is more complicated and is harder to implement than the pessimistic lock pattern. As a result, we decided to use a pessimistic offline lock pattern to handle concurrency problems in the online examination system. The lectures can perform any operations they want to at any time, without having to worry about other lectures that may be operating in parallel.

#### Implementation:

We implemented one centralized lock manager for all the lock requests. We need to avoid a situation where two lecturers are attempting to modify an exam paper simultaneously. We implemented an exclusive read lock to prevent this situation. Specifically, when a lecturer is modifying the exam, other users are prohibited from loading the exam, thus, all lectures are guaranteed to see the latest exam information when it's their turn. One lecture requests to lock an exam, the lock manager first checks whether the exam is being accessed, if not, the lock manager locks the exam by its ID, and then authorizes the lecture the power to update the exam. When the lecturer successfully finishes updating, the centralized lock manager releases the lock the exam is available to the public again. As can be seen, the exclusive read lock ensures correctness and consistency.

#### Example of Use:

When two lecturers are required to modify the same exam simultaneously, only Lecturer A will be granted access to the exam. Lecturer B can only access the exam when Lecturer finished editing and released the lock.



## 4. Source Code Directories Structure

---

```
.
├── .github
│   └── workflows
├── docs
│   ├── deliverables
│   └── minutes
└── src
    ├── frontend
    └── java
```

<b>Heroku Link</b>	<a href="https://online-examination-app.herokuapp.com/">https://online-examination-app.herokuapp.com/</a>
<b>Release Tag</b>	SWEN90007_2020_Part3_A_Team
<b>LucidChart Link to the diagrams</b>	<a href="https://app.lucidchart.com/documents/edit/e477d18e-9290-45a3-83c9-17046ac7ee1c#?folder_id=home&amp;browser=icon">https://app.lucidchart.com/documents/edit/e477d18e-9290-45a3-83c9-17046ac7ee1c#?folder_id=home&amp;browser=icon</a>

## 5. Testing Instructions

---

### 5.1 Please note:

1. the register function has not been implemented
2. If a student cannot see any exams, it is either because no exams have been created for that subject, or the instructor has not published the exam. If it is the second case, after the instructor has published the exam, the student should go back to the previous page, then go into the subject page again to be able to see the newly published exam. Refreshing the page will not work because refreshing the page will not send requests to the server.

Login Credentials:

For your convenience, we have some predefined users in the database, they teach different subjects, feel free to choose which any of them to login:

Username	Password	User Type
maria	Maria123	Instructor
toby	Toby123	Instructor
stuart	Stuart	Student

### 5.2 Testing Authentication

#### 5.2.1 Login with a correct combination of username & password (Main Flow)

- Log into the system using one of the correct instructor credentials
- Click 'Subjects' on the left menu
- All subjects that the instructor teaches should be shown

#### 5.2.2 Login with wrong credentials(Wrong password)

- Log into the system using the wrong password
- Log in will be rejected, the user will be prompted to try again

## 5.3 Testing Authorization

### 5.3.1 Instructor is authorized to edit an exam

- Log into the system using an instructor credentials
- Click 'Subjects' on the left menu
- All subjects that the instructor teaches should be shown
- Click edit button of an exam and goes to the exam form
- Modify and update the existing exam information and questions
- Click submit button to update the exam
- The exam is updated successfully

### 5.3.2 Student is not authorized to edit an exam

- Log into the system using an student account
- Click 'Subjects' on the left menu
- All subjects that the students enrolled in will be displayed
- The student will not be able to see an "edit" button
- The student is only authorized to view or take the exam instead