Architecture Document

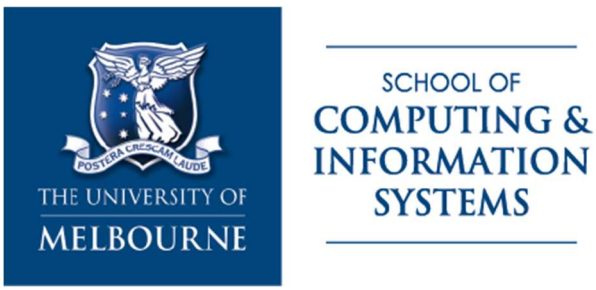
SWEN90007

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



Revision History

Date	Version	Description	Author
25/10/20	01.00-D01	Initial draft	Hao Ding
28/10/20	01.00-D02	Updated Concurrency	Hao Ding
29/10/20	01.00-D03	Added diagrams	Hao Ding
31/10/20	01.00-D04	Update class diagrams	Zihan Zhang
01/11/20	01.00-D05	Update rational	Zihan Zhang
01/11/20	01.00	Final version	Zihan Zhang

Table of Contents

1. Introduction 4
   1.1 Proposal 4
   1.2 Target Users 4
   1.3 Target Users 4
2. Architectural representation 4
   2.1 Class Diagram 6
3. Architectural Patterns 7
   3.2 < Authorization Enforcer> 8
   3.3 < Secure Pipe > 9
   3.4 < Pessimistic offline lock pattern > 10
4. Source Code Directories Structure 13
5. Testing Instructions 13
   5.1 Please note: 13
   5.2 Testing Authentication 14
      5.2.1 Login with a correct combination of username & password (Main Flow) 14
      5.2.2 Login with wrong credentials (Wrong password) 14
   5.3 Testing Authorization 15
      5.3.1 Instructor is authorized to edit an exam 15
      5.3.2 Student is not authorized to edit an exam 15

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

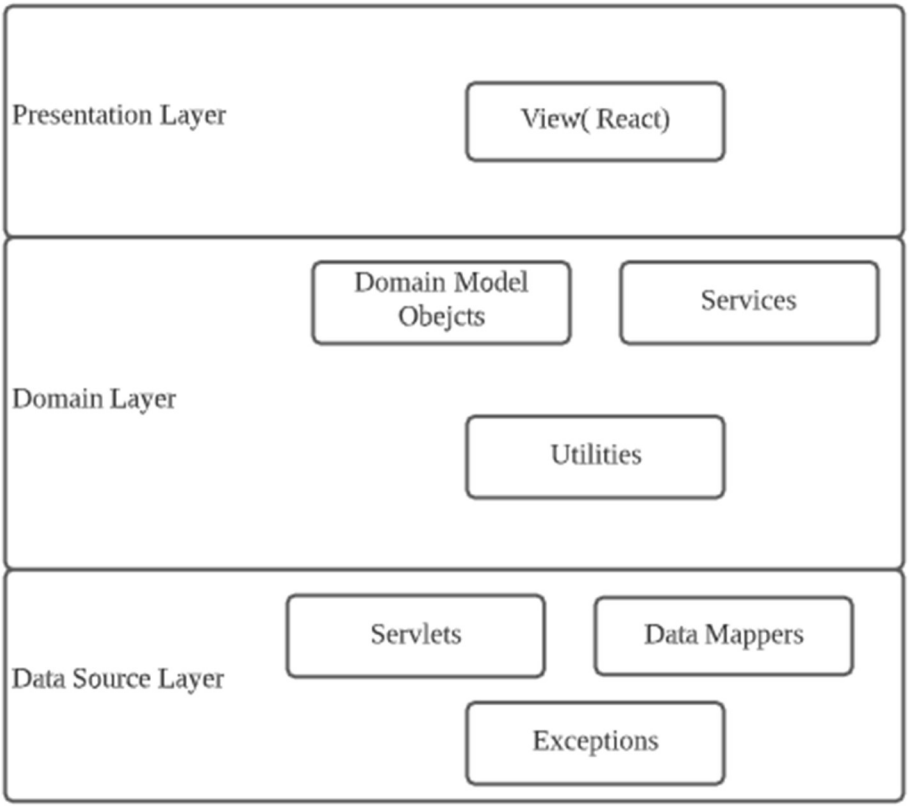
This section explains the concept of some important terms that will be used throughout this document. These terms are detailed alphabetically in the following table.

Term	Description
LMS	Learning Management System
SSL	Secure Sockets Layer

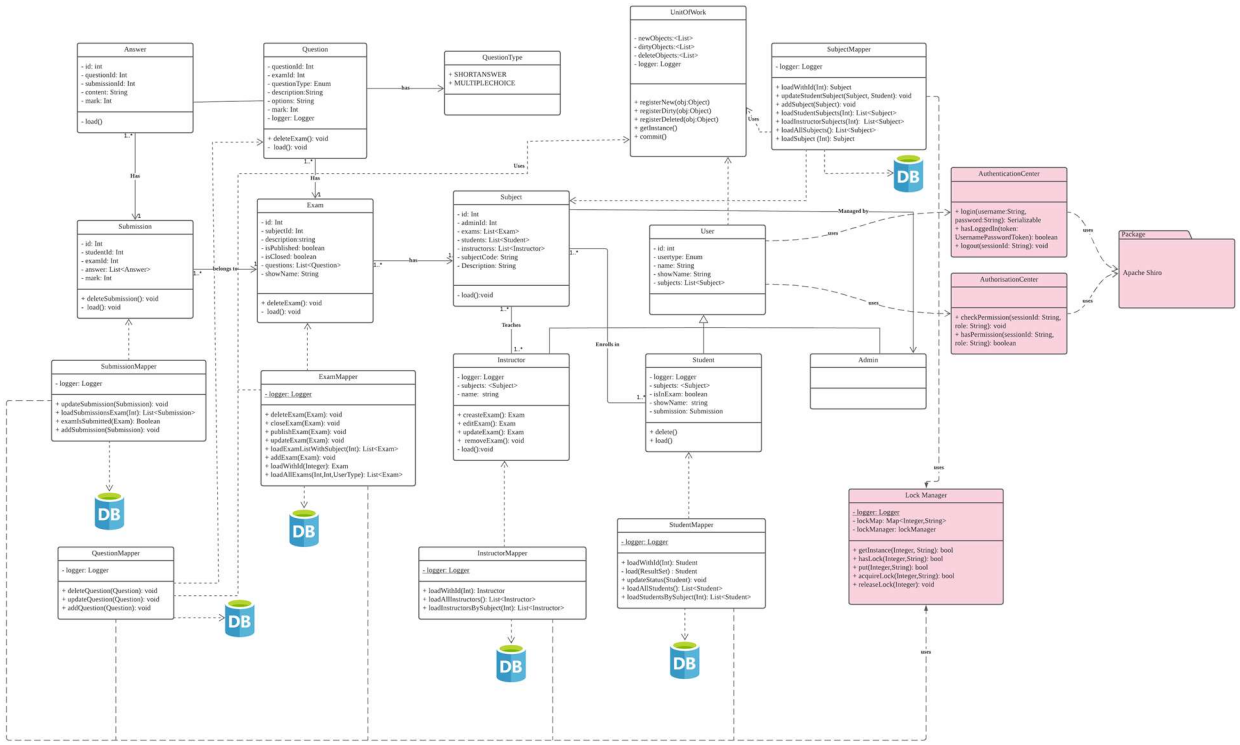
2. Architectural representation

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

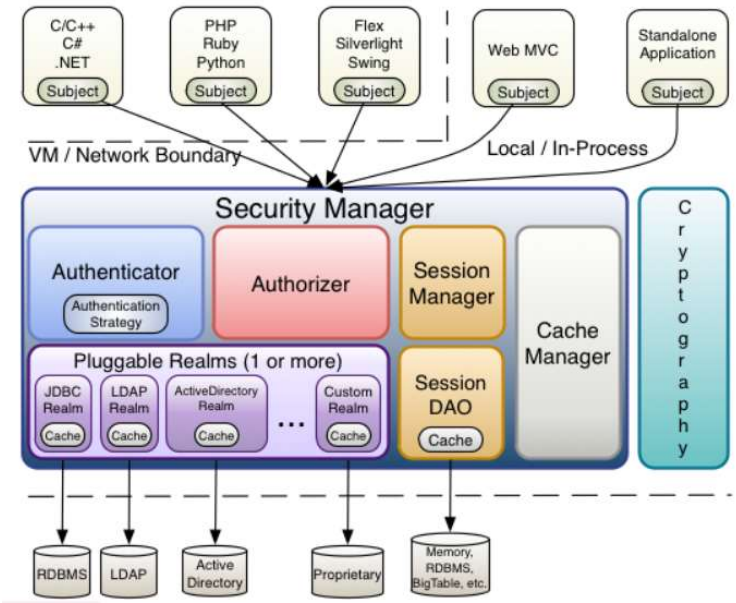
- The presentation layer displays the system to the users, it is the interface between user interactions and the system. We used React as the frontend framework. All the user inputs are sent from here to the lower layers.
- The domain layer contains all the objects as well as their business logic. Furthermore, 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.


Presentation Layer
View(React)
Domain Model Obejcts
Services
Domain Layer
Utilities
Servlets
Data Mappers
Data Source Layer
Exceptions

2.1 Class Diagram



We use Apache Shiro as the security framework to authenticate and authorize users. The underlying Shiro architecture is below:



Source: https://shiro.apache.org/architecture.html

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.

We use Apache Shiro as the security framework to authenticate and authorize users. It is a lightweight and easy-configured framework that provides all the functions that we need.

3.1 < Authentication Enforcer>

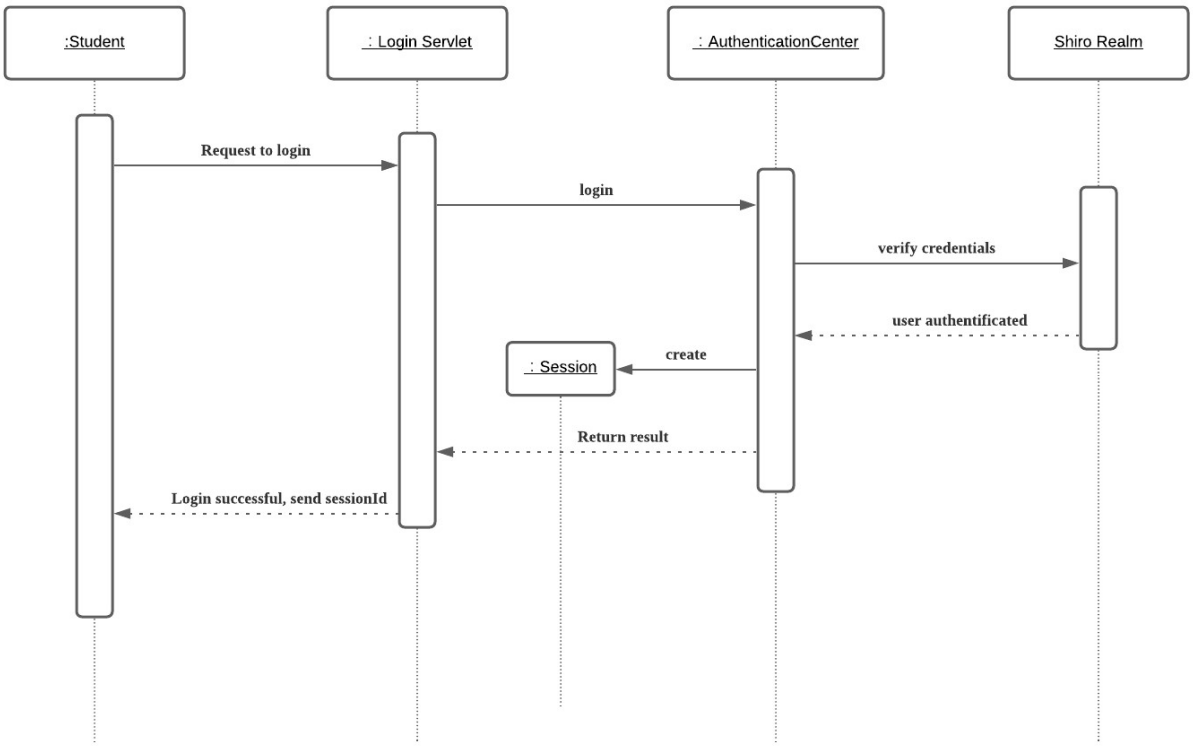
Pattern	Reason
Authentication Enforcer	The server can potentially receive HTTP requests sent from any clients. If the server accepts all the incoming requests, the server and the database information may leak or even be hacked. Therefore, we must ensure the end client is an authenticated entity, and 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 framework is utilized to implement the underlying authentication mechanisms. First of all, when the application launches, Shiro will create a realm that stores the credentials retrieved from the database. Then, when the login servlet passes the user's credentials to the authentication enforcer, and then a token is generated from the provided credentials. Shiro will help us to compare the token and the credentials that stored in the realm. If the authentication succeeds, a new session is created and stored in the session DAO, the associated session Id will also be sent to the user; if not succeeds, an error will raise and will refuse the user to log into the system.

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 a 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:

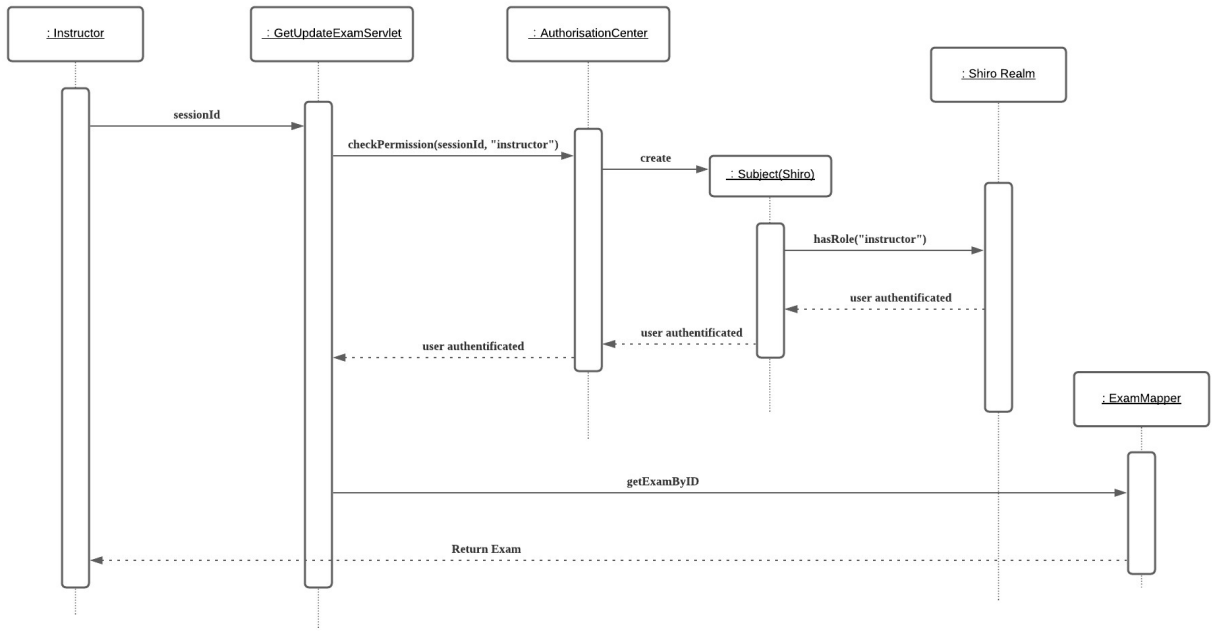
We implemented the authorization enforcer pattern whenever there is a communication between the user and the server. when the application launches, Shiro will create a realm that stores the permissions or privileges that stored in the database.

For example, after a student logging in, all the student's requests are sent to a centralized authorization enforcer, i.e. AuthorisationCenter. The enforcer verifies the student's identity through the sessionId and determines whether the request has the permission to do the action. 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 because the student does not have the permission.

It is also noteworthy that we are using React and React-Router as the frontend, so we have also used conditional rendering based on the user's identity. For example, the interface of a student will not have "Edit", "Mark", "Publish", "Close" buttons for an exam. This is the first authorization to make sure the user does not have the chance do unauthorized actions. However, if they manage to bypass this, we are still authorizing them by using Shiro.

Examples of Use:

After successfully logged in, the lecturer is assigned with a sessionId. When the lecturer requests to update an exam, the lecturer sends back the sessionId and AuthorisationCenter will use the sessionId to create a Subject (by Shiro) that is bound to that particular session. Shiro will check if the Subject has the permission by searching the realm. If this session has the permission, then it allows the subsequent business transactions, i.e. get the exam information through ExamMapper. If, however, the session does not have the permission, then an error will throw and refuse subsequent actions.



3.3 < Secure Pipe >

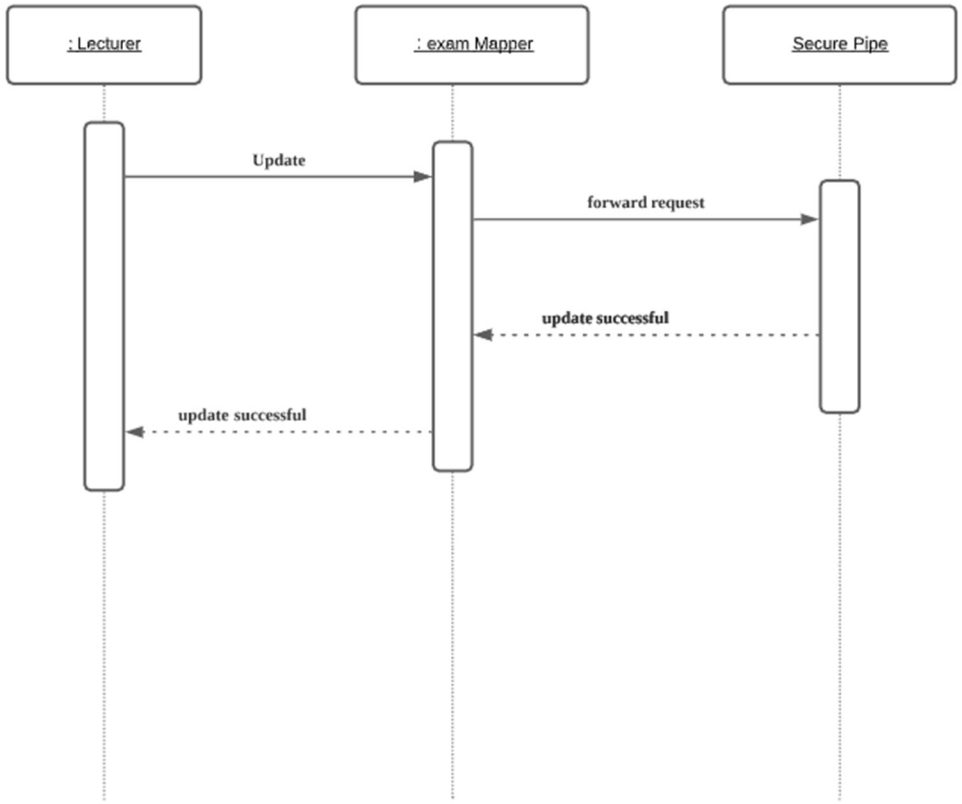
Pattern	Reason
Secure Pipe	Even with strong authentication and authorization mechanisms implemented, the online examination system is not safe yet due to a lack of secure communication channels. For instance, malicious attackers can intercept and tamper the data during the transmission between the end-users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

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 sent to the database via the secure pipe channel, and the database will return a formation via the secure channel.



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.

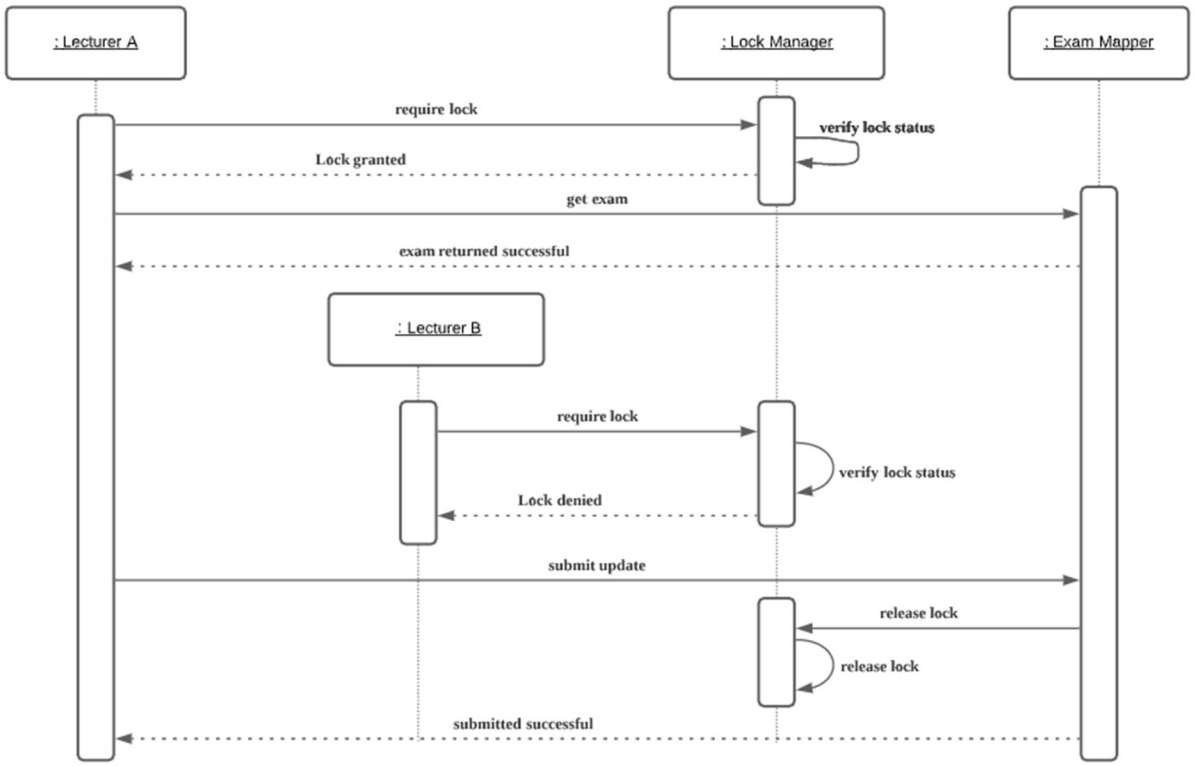
Implementation:

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 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.

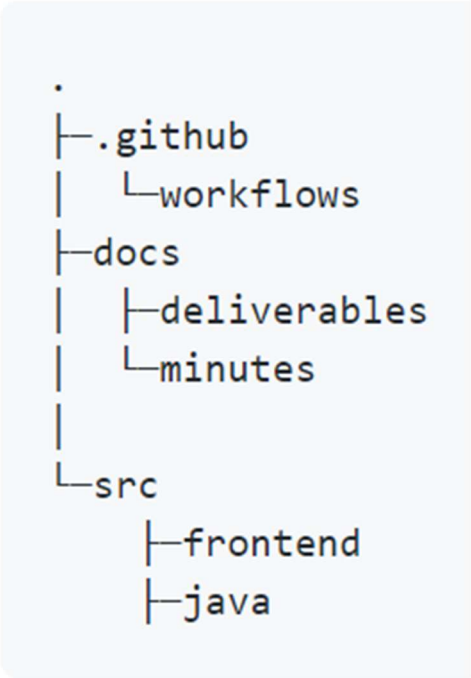
It is also noteworthy that we are still using singleton Unit of Work because we do not reply on UoW to manage the concurrency. If two actions happen simultaneously, we only allow one thread continue and block the other one. Once the first thread commits the changes to the database, we clear and empty the Unit of Work objects so that two sessions does not affect each other.

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.


: Lecturer A
: Lock Manager
: Exam Mapper
require lock
verify lock status
Lock granted
get exam
exam returned successful
: Lecturer B
require lock
verify lock status
Lock denied
submit update
release lock
release lock
submitted successful

4. Source Code Directories Structure



Heroku Link	https://online-examination-app.herokuapp.com/
Release Tag	SWEN90007_2020_Part3_A_Team
LucidChart Link to the diagrams	https://lucid.app/invitations/accept/46d1f857-611e-4b61-a7ce-1418bdf8dac0

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.

3. If you have the authority to do an action, for example, logged in as an instructor but it tells you that you do not have the authorization. Then it is probably because this session has expired so Shiro cannot authenticate the session subject anymore. Please logout and login again.

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	UserType
eduardo	123	admin
maria	123	instructor
toby	123	instructor
ben	123	instructor
zhazz	123	student
tony	123	student
ash	123	student
mark	123	student
steve	123	student
james	123	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 a 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