VIETNAMESE-GERMAN UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

Frankfurt University of Applied Science

Faculty 2: Computer Science and Engineering

Thesis Topic:

A Web-Based Tool for Supporting Student-Lecturer Classroom Interaction

Fullname: Le Khac Hoan Vu

Matriculation number:

First supervisor: Prof. Manuel Clavel

Second Supervisor: Prof. Michel Toulouse

BACHELOR THESIS

A thesis presented for the degree of Bachelor Engineering

Binh Duong, Viet Nam

Presented by

LE, KHAC HOAN VU

Matriculation number:

Bachelor Thesis

Web-Based Tool
for Supporting Student-Lecturer Classroom Interaction

30th November 2019 to 30th January 2020

Supervisor:

Prof. Manuel Clavel   Assessor

Prof. Michel Toulouse   Co-Assessor

Vietnamese - German University

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Declaration

I herewith claim that this thesis, which I present to Vietnamese German University for the Bachelor's Degree in Computer Science, is entirely arranged by me. I completed this thesis in accordance with the counseling and managament of Prof. Manuel Clavel and Prof. Michel Toulouse. I guarantee that this report has not been proposed anywhere, does not violate any laws concerning intellectual properties, and is only arranged for my scholarly specifications.

Date:

Signature:

Acknowledgement

Contents

References   **50**

Chapter 1

Introduction

The high tech environment of our present-day proposes a lot of alternatives for teachers and students. It can be mind-boggling to determine where to start since there are too many options. What could be the motivation for a beneficial advance in the classroom? What could benefit the students the most? The first thought that went through my mind was, classroom interaction between lecturers and students.

Besides, nowadays, there are online courses in which the students can take part at their homes, so to put on a competition, the traditional universities should be able to step up their game. **Interaction** is the key. They should make interaction in the classroom become as seamless as possible. In physical classrooms, teachers often have limited knowledge about their students, so in the most cases, they don't know that if there is a student who is having trouble understanding their lectures or not. The other disadvantages of the physical classrooms are that, sometimes classrooms are big, so the teachers cannot hear the calls for help from students. And, when a student is allowed to ask a question, they often have to speak very loud for the whole class to listen. The students sitting in front of the asking student will also have to turn back to look at the asking student.

For those reasons, I carried out my project, which is called the Web-Based Tool for Supporting Student-Lecturer Classroom Interaction. The app contains my solutions for the abovementioned problems, such as:

- A class map for the teacher to know where a particular student is sitting, and the basic information about that student. This class map can also be used to see the results of students on a particular quiz.

- Teachers can create quizzes, and students can do quizzes directly on the app. While the students are doing the quiz, the teacher can have a live update of their results through the class map. After a quiz is done, the teacher can have the average result of his class doing the quiz, and the student can also review their solution for that quiz.

- There is a function for a student to call for the teacher's attention just by clicking on one button. The teacher can close the call or accept it. If the teacher accept it, a video call window will be opened. The student's voice will then be heard through the teacher's speaker, and their image will be seen on the projector.

To make the thesis easy to follow, I divided the main content of it into two parts: one is the user manual, specifically written for end users so that anyone can use it without knowing about the tech behind; and one is the developer manual, which is designed for programming specialists to read, modify or develop the application.

Chapter 2

User manual

2.1 Login screen
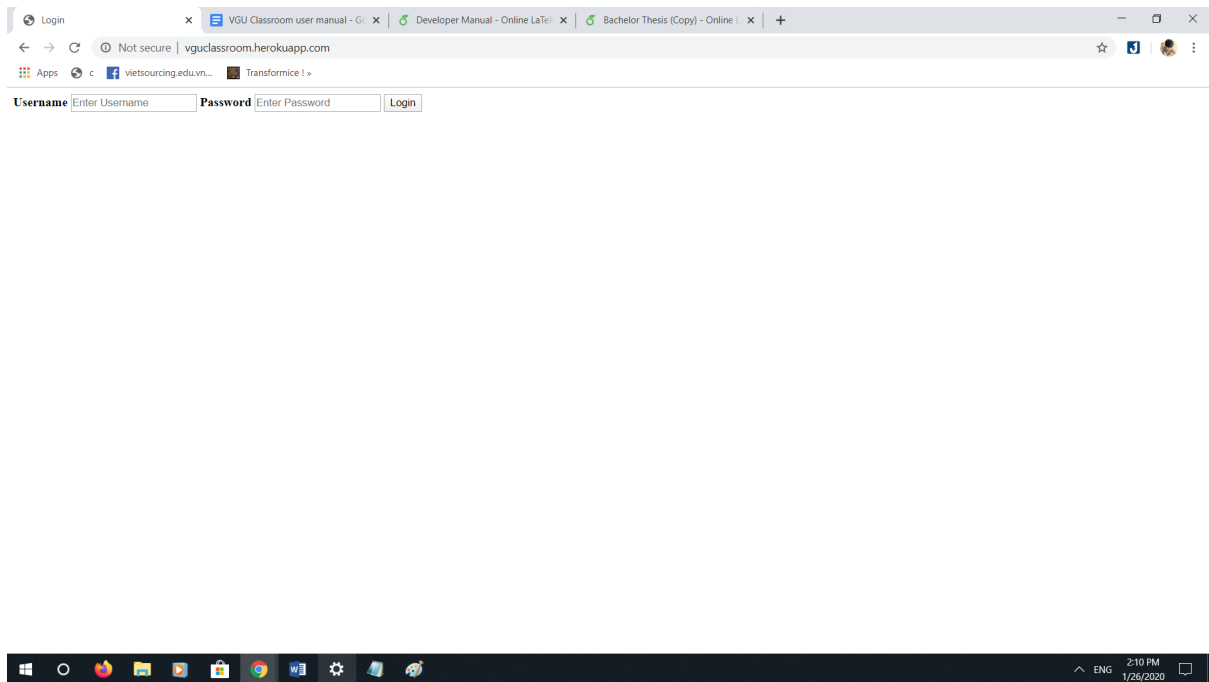
There are just two input fields to input username and password on the login screen. Enter your username and password to log into your account. After you logged in, the app will appear differently based on your account level. There are 3 levels: Administrator, Lecturer and Student.

2.2 Administrator
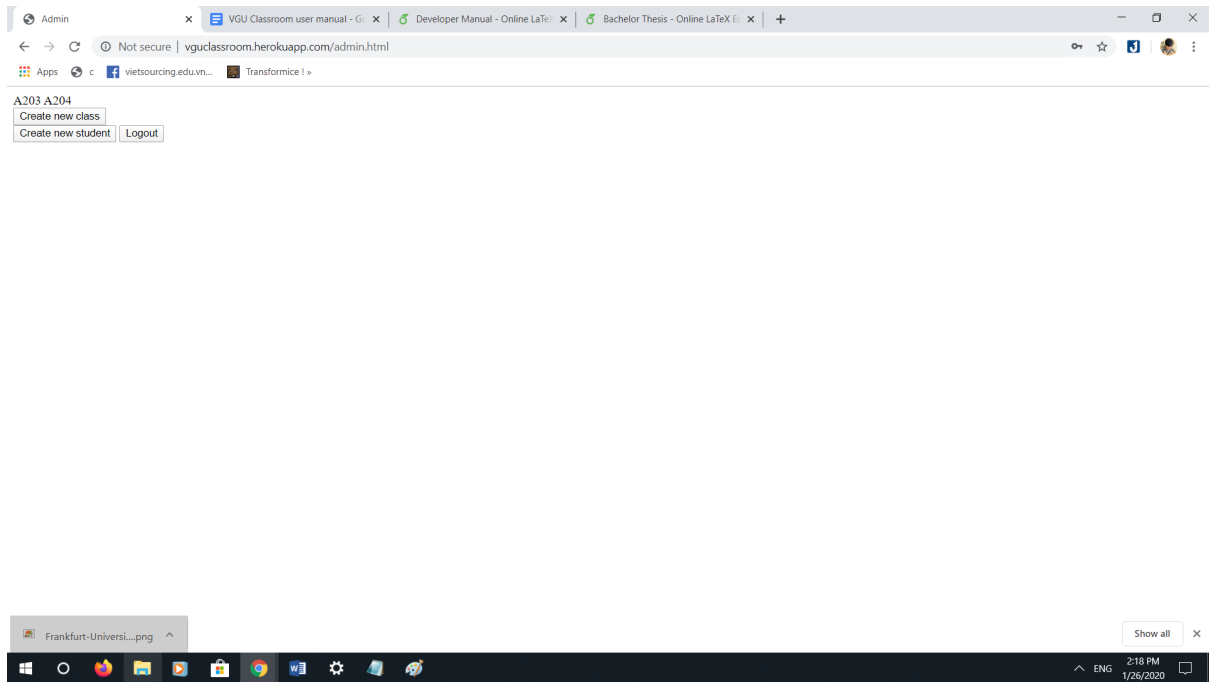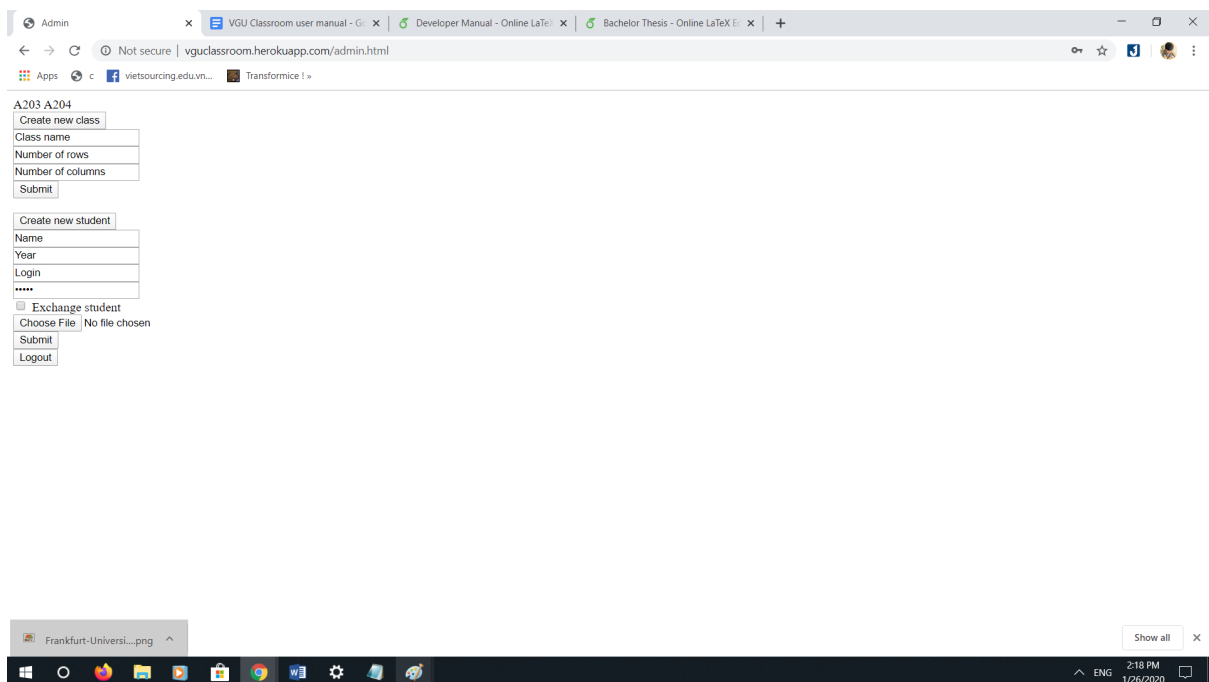
The screen for administrator is simple. The first row states the existing classes. There are 3 buttons, one for creating new class, one for creating new student, and one for logging out.

2.2.1 Creating new class
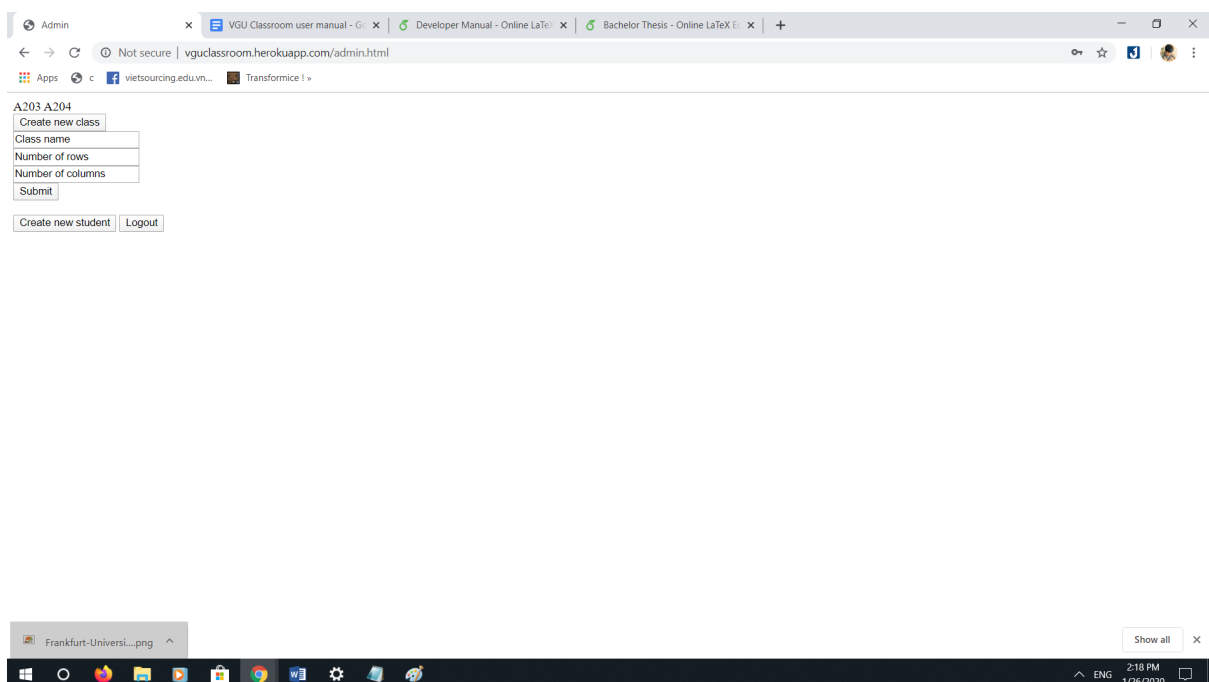
Clicking on the "Create new class" button will show input fields to input the class information. In this application, a seat map of a class is defined as a rectangle. Inputting the right number of rows and columns of seats will help forming the proper "rectangle" of the class map. After inputting all the information needed, click Submit to create a class.

2.2.2 Creating new student

Clicking on the "Create new student" button will show input fields to input the student information. Here, the admin can input the student's name, intake, username and password for logging into the application, and a checkbox to check if that student is an exchange student. The admin can also upload the photo of the student. After inputting all the information needed, click Submit to create a student.

2.3 Lecturer
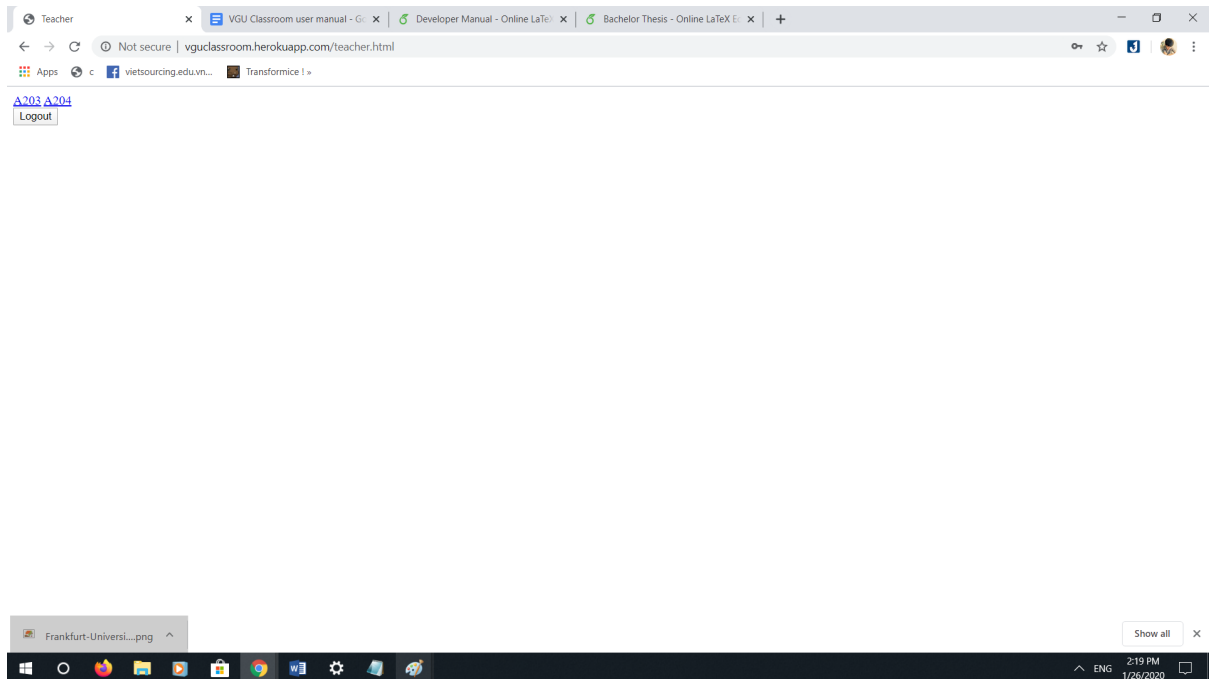
After logging in, the screen of a lecturer is simple, there are just names of the classes that are available. The lecturer can click on a class name to get into that class.

This is the display for a lecturer when he's in a class. The first thing to see is the class map. The cells that are colored in the bottom represent an occupied seat. If the student is in the same intake with the class, the bottom left frame will be green, otherwise it will be red. If a

student is an exchange student, the bottom right frame will be black, otherwise it will be white. If you hover the mouse on an occupied seat, it will show the image and the name of the student sitting in that seat.

Let's take a look at the information of the students who are taking the class:

	ABC name	ABC year	123 exchange
1	Vu	CS2015	0
2	Inu	CS2016	1
3	Thuy	CS2016	0
4	Foreign	CS2015	1

First, we will consider the frame for Vu:

As we can see from the information table, Vu's intake is CS2015, and he is not an exchange student. The above capture of the screen shows that, the intake that is taking the class is CS2016, therefore Vu is not in the same intake with the class, which colored the bottom left frame red. Since Vu is not an exchange student, the bottom right frame is white.

For the frame of Inu:

Inu's intake is CS2016, and he is an exchange student. He is in the same intake with the class, so the bottom left frame is green, and he is an exchange student, so the bottom right frame is black.

For the frame of Thuy:

Thuy's intake is CS2016, the same intake with the class, and he is not an exchange student. Therefore, the bottom left frame is green, and the bottom right frame is white.

For the frame of Foreign:

Foreign's intake is CS2015, not the same intake with the class, and he is an exchange student, so the bottom left frame is red, and the bottom right frame is black.

Below the class map, there are texts showing the state of the class, the intake taking the class, and buttons that the lecturer can click on to use his functions:

Class is: **ON**
Intake: CS2016

| Turn class ON | Turn class OFF |

| Create new multiple choice question | Show inactive questions |

Active quizzes:

| Show finished questions |

| Show result through class map |

| Show default class map | Logout |

2.3.1 Turning a class on and off

A class always has its default state as OFF. If the teacher wants the student to join the class, he will have to turn the class on first. Clicking the "Turn class ON" button will show a text field to input the Intake that is taking the class. After inputting this, click "Submit the intake and turn class ON" to turn on the class. After the lecture, the lecturer can turn the class off by clicking the "Turn class OFF" button. All seats taken by the student will be available again after turning the class off.

Class is: **OFF**

| Turn class ON |

| Input the intake |

| Submit the intake and turn class ON |

| Turn class OFF |

| Create new multiple choice question | Show inactive questions |

Active quizzes:

| Show finished questions |

| Show result through class map |

| Show default class map | Logout |

2.3.2 Quiz

In this application, a quiz only has one multiple choice question. You can create a new quiz by clicking the "Create new multiple choice question" button.

Create new multiple choice question

Name

Input question

Input Answer A

Input Answer B

Input Answer C

Input Answer D

Seconds:

Solution ▼

Submit

After clicking the button, the application will show fields that the lecturer needs to fill in to create the quiz. The required fields are the name of the quiz, the question of the quiz, the four choices, the time allotted for the quiz, and the right solution. After filling those fields, the lecturer can click Submit to create the quiz. A newly created quiz is an **inactive** quiz. The

lecturer can click on the "Show inactive questions" button to see all inactive quizzes.

Create new multiple choice question

Basic CS quiz

following is not a
computer language?

ALGOL

COBOL

PASCAL

DRAM

100

D

Submit

Show inactive questions

Basic CS quiz

Click on the link to come to the page displaying that quiz.
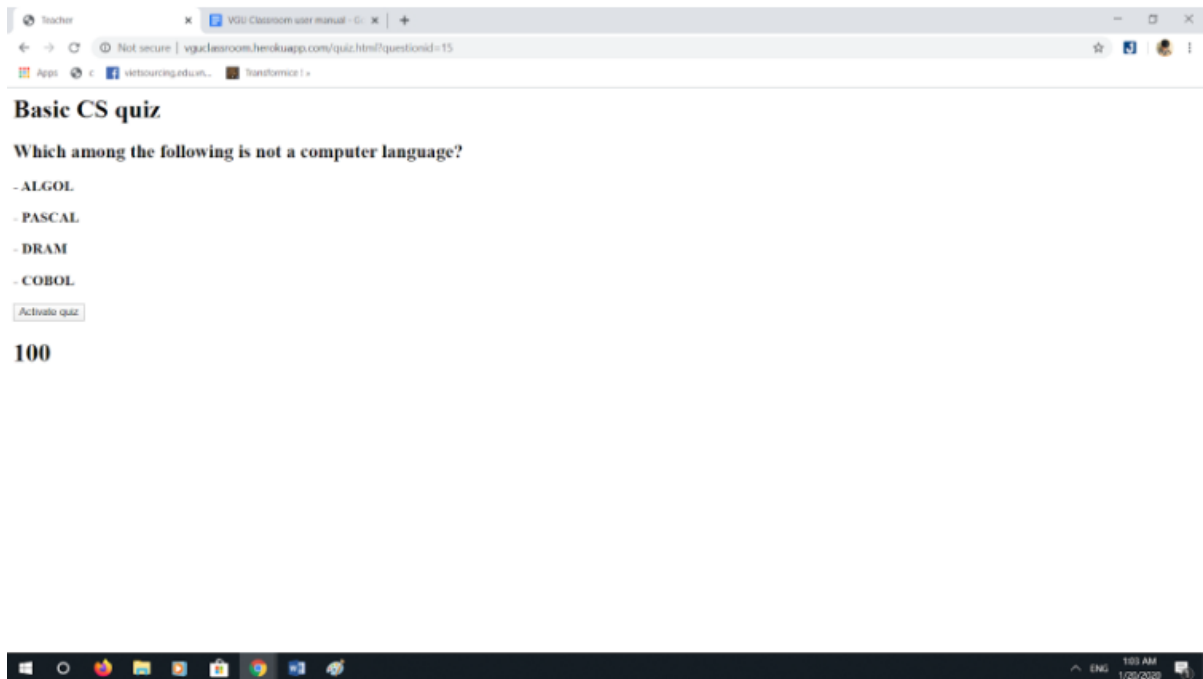
The page displaying a quiz will consist of its name, its question, its answers (the orders are randomized, so it will not be the same with the order when the lecturer first input the quiz), a button to activate the quiz, and a timer for a quiz. The lecturer can click the button to activate the quiz, so the students can start doing it. The timer will begin to work when the quiz is activated. After the timer runs out, the quiz is **finished**.

When a quiz is **active**, the teacher can see a live update of students doing the quiz by clicking the button holding the name of the quiz.
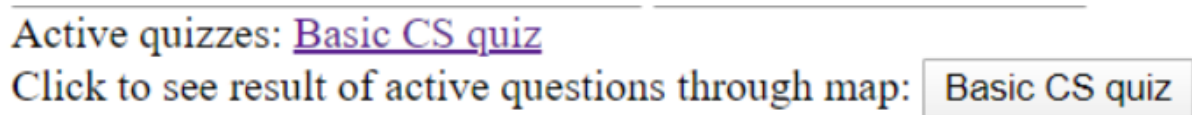
Apart from being capable of live updating, the result of this button is the same with the function for the "Show result through class map" button, so it will be discussed later.

Let's have an example. The four abovementioned students did the quiz, and this is the result. (N is for "no answer"). From this table, Vu and Thuy did the quiz right, Inu got it wrong, and Foreign had no answer.

	ABC name	ABC answer	ABC solution
1	Vu	D	D
2	Inu	C	D
3	Thuy	D	D
4	Foreign	N	D

The teacher can click on the "Show finished questions" to show the finished quizzes. Links of the finished quizzes will appear.

Show finished questions

Basic CS quiz
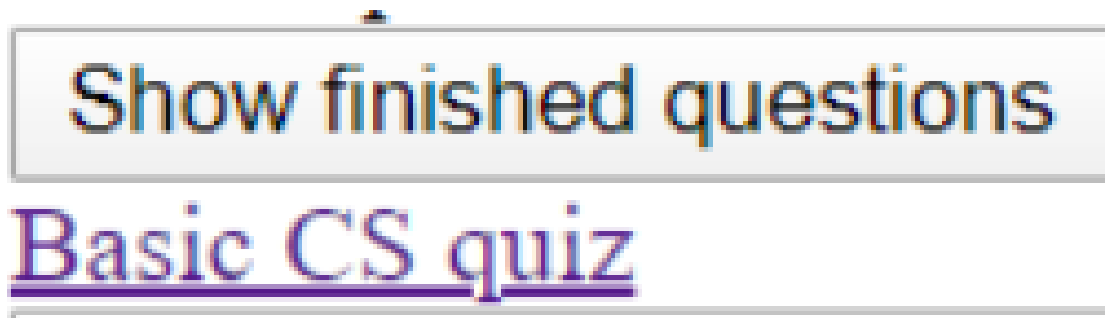
Click on the link to see the average result of the quiz.

Basic CS quiz information

Right answers: 2/4

Wrong answers: 1/4

No answers: 1/4
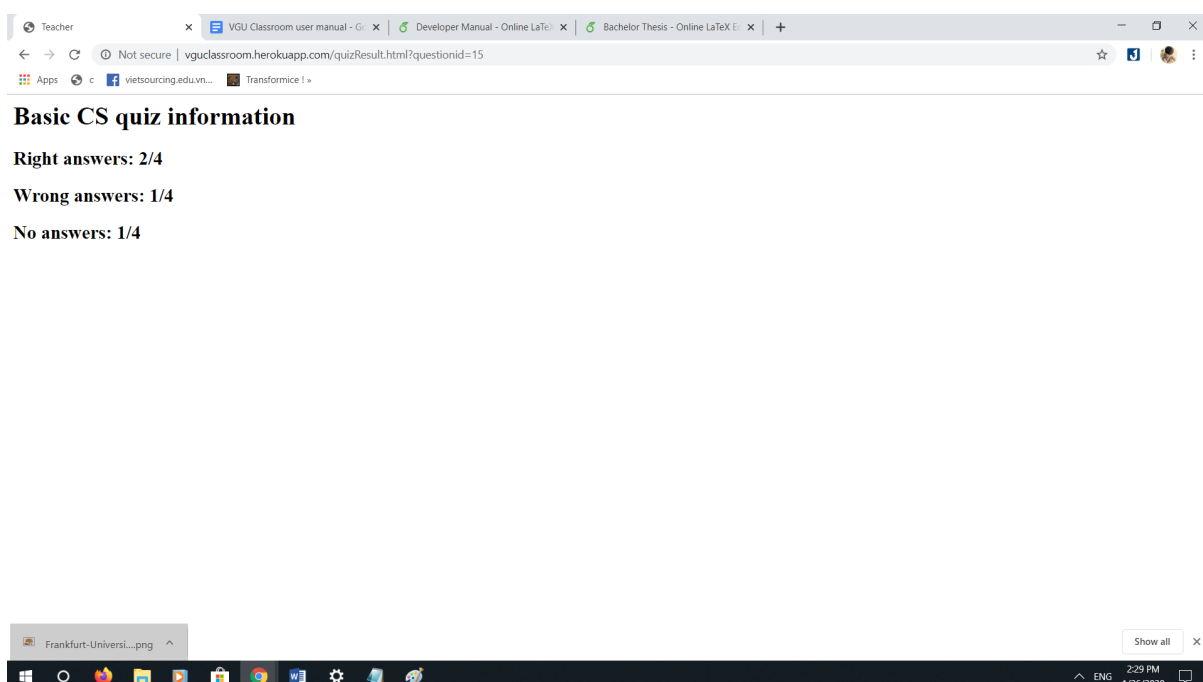
There are 4 students in the class at that moment, so the total number of students taking the quiz is 4. Vu and Thuy got it right, so there are 2 right answers out of 4. Inu was wrong, so there is 1 wrong answer. Foreign could not answer it in time, so there is 1 student who had no answer.

Besides this page which shows the average result of the quiz, the teacher can also see the result of the quiz through the class map by clicking "Show result through class map". It will show buttons representing the finished quizzes. Click on the button representing a finished quiz to see the result of that quiz through class map.

Vu and Thuy got the right answer, so their seat is colored green. Inu got the wrong answer, so their seat is colored red. Foreign did not have the answer, so his seat is black. The lecturer can also revert back to the normal class map by clicking "Show default class map."

2.4   Student

Upon logging in, the student will see the available classes, and the quizzes done by them. A student can click on a quiz to review that quiz, or click on a class to get into that class.

2.4.1 Quiz review
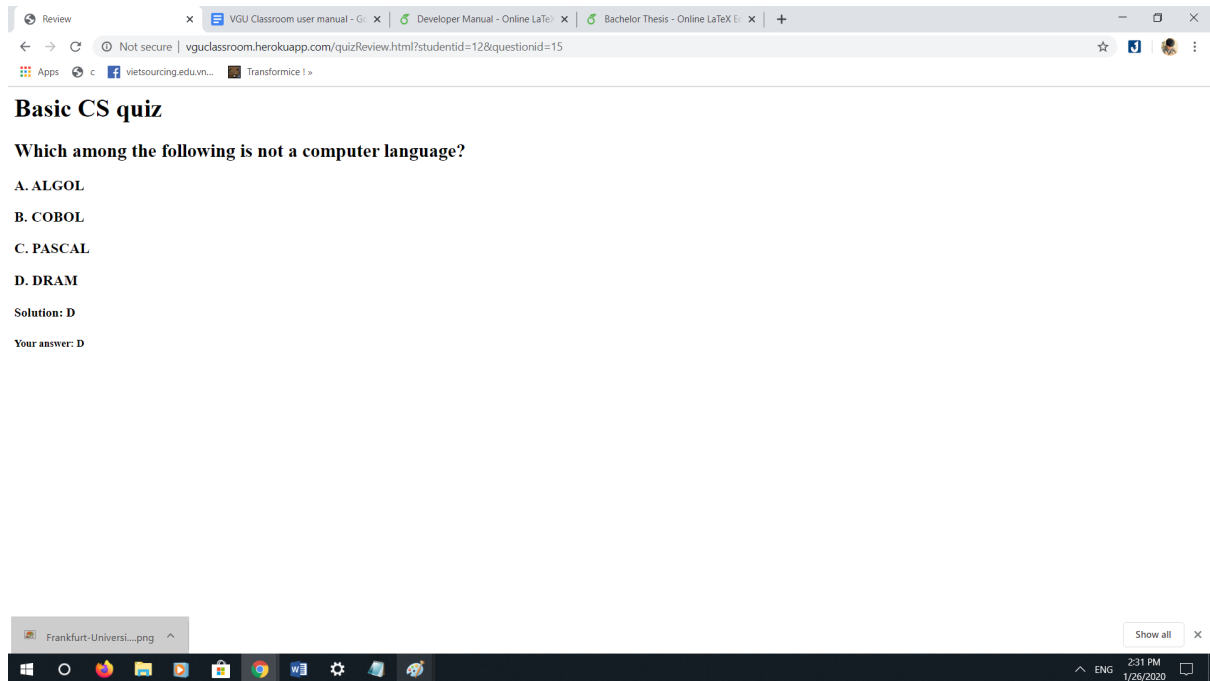
The quiz review screen shows the name of the quiz, the question, all the choices, the right solution, and the student's answer

2.4.2 Class
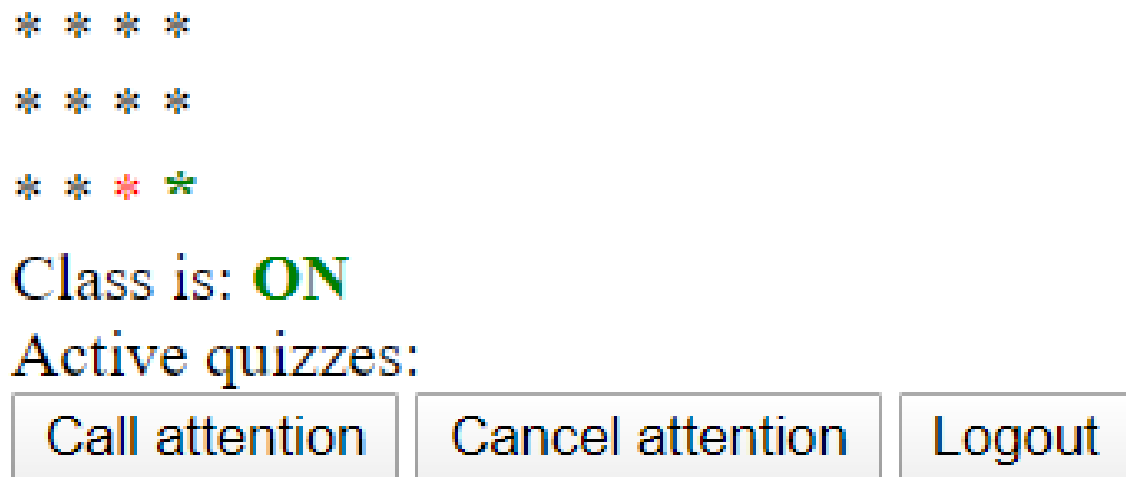
This is the display of a class for a student in the application. Since students do not need to see as much information of other students like the lecturers, the class map here is just represented by stars. Black stars are for the unoccupied seats, red stars are for occupied seat, and the larger

green star is for the seat that the user is sitting in. To occupy a seat, a user can click on the black star representing that seat. Occupied seat cannot be occupied by another student. If a student, who already occupied a seat, occupies another seat, his old seat will be left free, and will become a black star again.

If there is an active quiz during class, the application will update it immediately to the student by a link next to the text "Active quizzes". The student can click on the link to go to the quiz.
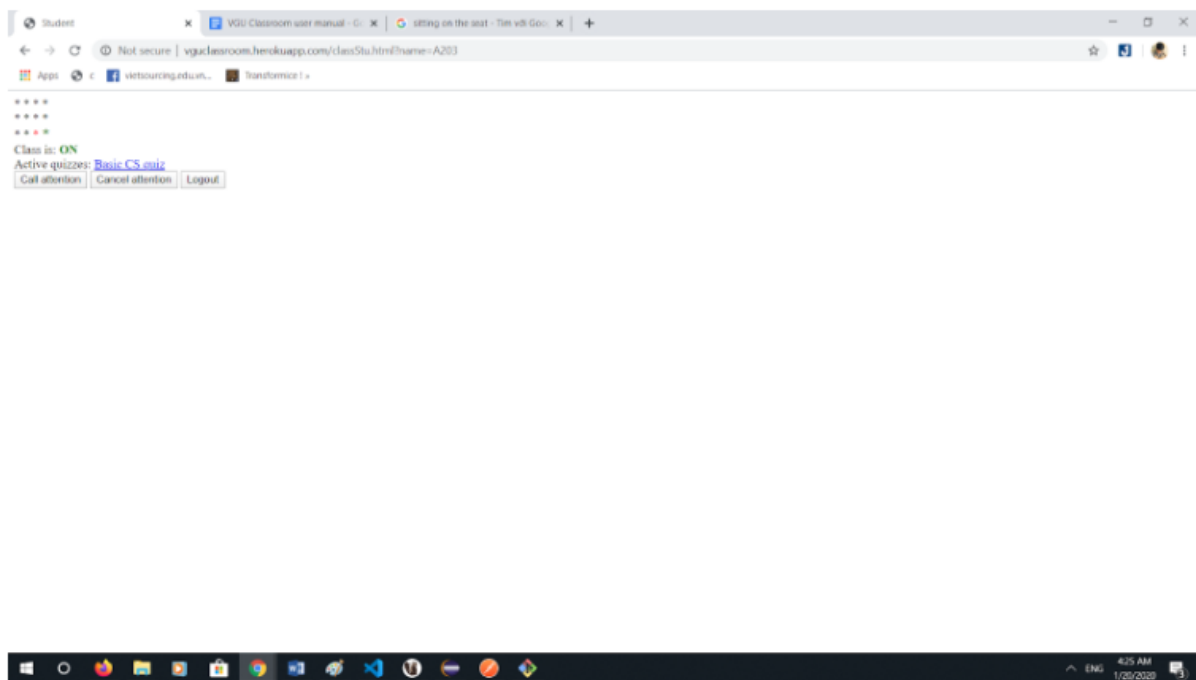
The page for the test will show the name of the test, the question, and the choices appearing in a random order. To complete the quiz, the student can check on the right answer, and click "Submit answer" before the timer runs out
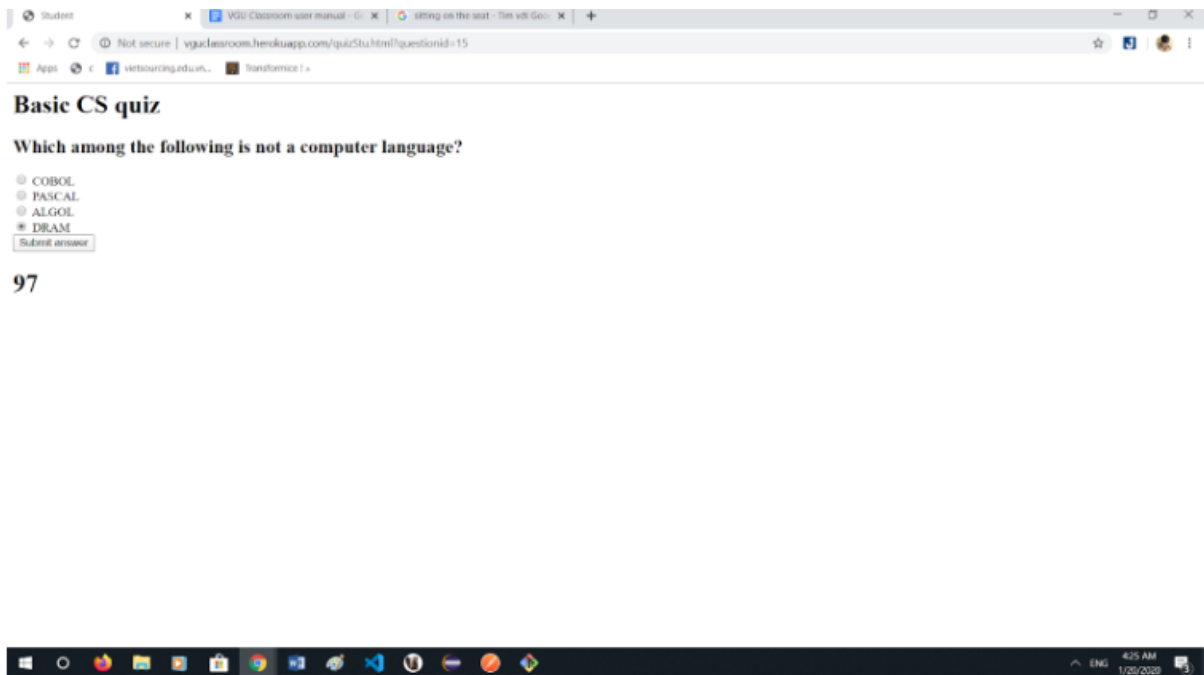
In the page of the Class, the student can click on the button "Call attention" to call for the attention of the lecturer, without having to wave their hand, or call for the lecturer by their voice. When a student is calling for attention, their seat on the lecturer's display will turn red, and the lecturer will hear beeping sound to alarm them that there is a student calling for them.

The student can also close the attention call with the button "Cancel attention"

2.5 Use case descriptions for some main functions

Use Case 1	**A student takes a seat**
Level:	User-goal
Actors	• Student • Class page
Preconditions:	• The Student is logged in. • The class is turned on by the Teacher • The Student is in the webpage of the Class
Postconditions:	• The Student occupied their desired seat • If the teacher reloads his Class page, he can see the student on the classmap.

Main Success Scenario:

1. The Student picks a seat.

2. The page tells the Student that he picked the seat successfully.

3. The page reloads and turns the star representing the picked seat into a large green star.

Alternate scenario:

2.a A seat has been picked by another student

1. Page shows failure message

2. Student returns to step 1

Use Case 2	Doing a quiz
Level:	User-goal

Actors	• Teacher
	• Student
	• Application

Preconditions:	• Both the Student and the Teacher is logged in
	• A quiz is created by the Teacher
	• Both the Teacher and the Student is in the web-page of the Class

Postconditions:	• The quiz is completed. (which means the remaining time for the quiz reaches zero)
	• The answer of the student is saved. If he cannot answer, it will be saved also.
	• The Teacher can check for the result of the Student through the class map.
	• The Teacher can check for the average result of the Students on the quiz.
	• The Student can review his answer for the quiz.

Main Success Scenario:

1. The teacher clicks on the "Show inactive questions"

2. The application shows all quizzes that the teacher has created, but has not activated.

3. The teacher clicks on the link of the quiz that he wants to activate.

4. The application shows the page of the quiz.

5. The teacher clicks "Activate quiz"

6. The application begins to count down.

7. The teacher returns to the Class page, and click on the button holding the name of the quiz to see the live update of the quiz through the class map.

8. The application automatically shows the link to the quiz on the screen of the Student.

9. The Student clicks on the link of the quiz.

10. The application shows the page of the quiz.

11. The student checks on the answer.

12. The student clicks "Submit answer"

13. The teacher sees the result of the student through the class map (the bottom frame will be colored green if the student answered right, and it will be red otherwise).

14. The quiz counts to zero, and end itself.

Alternate scenario:

12.a The student did not answer in time.

1. The application repleaces the button "Submit answer" with a "Time
   is over" line.

2. The teacher sees the result of the student through the class map,
   with the bottom frame colored black.

Use Case 3	**Attention call**
Level:	User-goal
Actors	- Teacher - Student - Application
Preconditions:	- Both the Student and the Teacher is logged in - Both the Teacher and the Student is in the web-page of the Class
Postconditions:	The student and the teacher has a video call.

Main Success Scenario:

1. The Student clicks the button "Call attention"

2. The application tells the Student that the call is successful.

3. The application colors the seat of that student red on the Teacher's screen, and beeps.

4. The teacher clicks "Accept attention call"

5. The application redirects both the teacher and the student to a video call page.

Alternate scenario:

2.a There is another call at the moment

   1. The application tells the Student that there is another call right now.

   2. Student returns to Step 1.

4.a The teacher does not accept the call

   1. The teacher clicks "Close attention call"

   2. The application stops beeping, and returns the class map to normal.

4.b The student wants to cancel the calll

   1. The student clicks "Cancel attention"

   2. The application stops beeping, and returns the class map to normal on the Teacher's screen.

Chapter 3

Developer manual

3.1 Web app

A web app is an application that applies the client-server model, where the client is the browser and the server is the webserver. The Hypertext Transfer Protocol (HTTP) is applied for transaction of data over systems of connections. With this concept, the users do not have to count on any particular Operating System (OS) or hardware configuration. Therefore, web apps are able to support services that are cross-platform.

In a web app, all data and logic exist in the basis domain server. The app can be connected to with any browser, and without installation. Redistributing, administrating and preserving the app on multiple OS, which are the time-consuming processes of the development of a desktop app, are not necessary in web apps. The process to develop a web app bears 2 parts: Client-side and Server-side, which are often called Front-end and Back-end, respectively.

3.1.1 Front-end

A front-end developer is the one who is in charge of developing the graphical user interface arrangement of a site. The process of front-end developing is crucial on account of the fact that the user interface is the sole place that the users can communicate with the app. In this process, I used HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript (JS).

- HTML: The approved markup language for webpages, whose elements are the components of HTML pages.[1]

- CSS: This defines how HTML elements are to be presented. [2]

- JavaScript: The programming language for the Web. HTML and CSS can be interacted with by JavaScript. It can be used to compute, handle and authenticate data. [3]

3.1.2 Back-end

Back-end development generally aims attention at the logic and services of the web app. As the gist of the web app, it basically stores and executes all the functionalities of the application. It operates the data from the front-end, and sends back the outcome to the front end in a comprehensible manner.

A web server develops the back-end of the app. The mission of the web server is that, it delivers data for the clients, which, for the most part, are the browsers. There are various elements that make up a web server, and the center of it is an HTTP server. This HTTP server is a computer program that realizes the concept of URL and the protocol of HTTP. One can understand the fundamental concept of this process by looking at the following figure:
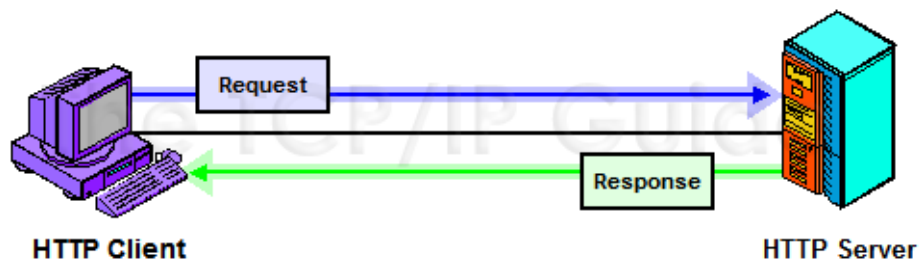
Figure 3.1: HTTP Client/Server information exchanging

From the figure, we can see the steps of the process as follows:

1. A connection with the Server is created by the Client using the protocol of HTTP.

2. Client transmits a request for data from Server.

3. A response is returned to the Client, transferring back the data.

This process comes with a fixed array of principles:

- The server shall respond to the client's request exclusively, and it cannot deliver requests to the client.

26

- It is necessary for the server to respond to all arriving requests. Alternatively, a timeout message will be sent to the client, signifying that the server has not replied within the particularized time.

- Each request needs to include a URL address that displays a specific service

3.1.3 Multiple-Page Application

Since the functions of the application can be accessed differently by three roles, and there are functions that should be viewed in different pages, putting the pattern of Multiple-Page Application into use is convenient for the application development. Multiple-Page Applications are the classic web apps that refresh the whole page and show a new one upon getting communicated with by the users. This pattern is good with the search engine, and it supplies an observable outline of the application to the user.[4]

3.1.4 Web workers

A web worker is a JavaScript that operates behind-the-scenes. It runs separately from other scripts without influencing the efficiency of the page. Rendering a new web worker additionally creates new threads to carry out that JavaScript code. As a result, a multi-thread architecture is efficiently developed, in which an application may accomplish multitasking.

In my application, 4 web workers are used for the following functions:

- 1 worker for the live update of students doing the quiz through class map for the lecturer.

- 2 workers for the function of Attention Call. (one worker is to update for the teacher that if there is any student calling him, and another is to update for the student that if their call is accepted by the teacher)

- 1 worker for the live update of active quizzes for the students.

3.1.5 JSON Web Token

JSON Web Token (JWT) is an open standard that gives description to a solid and independent method for a secured communication between partakers as a JSON Object. The information

transmitted in the communication can be confirmed and trusted because it is digitally signed. In my application, the JWTs are signed applying a secret (with the HMAC algorithm).

JWTs are made up of three components split by dots (.), which are the Header, the Payload, and the Signature. For that reason, a JWT usually appears like this: xxxxx.yyyyy.zzzzz

The header commonly contains two parts: the type of the token, which is JWT, and the signing algorithm. In my application, I used HMAC SHA256.

```
{
  "alg" : "HS256",
  "typ" : "JWT"
}
```

This JSON is Base64URL encoded to make the first part of the JWT

The payload makes the second part of the JWT, which holds the declarations about the user. In my application, a payload will look like this:

```
{
  "id" : 1,
  "role" : 1
}
```

Like the header, this JSON is also Base64URL encoded to make the second part of the JWT.

To construct the signature, you must sign the encoded header and payload with a secret and the algorithm designated in the header. The signing function will be: Algorithm(base64UrlEncode(header) + "." + base64UrlEncode(payload), secret). This signature is to validate: (1) the message was not adjusted while being sent, and (2) the sender of the JWT. In my application, when the user inputs the right username and password to login, they will earn a JSON Web Token. At whatever time the user desires to use a function, the user agent should send the JWT in the Authorization header employing the Bearer schema. The header should resemble this:

Authorization: Bearer (token)

The server will inspect for a genuine JWT in the Authorization header, and if it exists, the user will be permitted use the functions based on their role, which is stated in the payload of the JWT.[5]

3.1.6 REpresentational State Transfer (REST)

It is a structural fashion for determining principles between computer systems on the Internet, facilitating the data transmission between systems. The typical features of systems that follow REST, frequently declared as RESTful systems, are that they are stateless, and they set apart the concerns of client and server.

Division of client and server

In REST, the client and the server can independently put their tasks into action without each being aware of the other. The developer can alter the code on the client side on any occasion without influencing the working of the server, and vice versa.

Considering each side realizes what form of message to transmit to each other, the modularity and detachment can be maintained. The matters of the UI are detached from the matters of the data storage, which means that the adaptability of the interface traversing the platforms and the scalability will be better, because the elements of the server are reduced. Furthermore, this permits both the client and the server to be developed separately.

Statelessness

This means that, it is not required for the server to recognize the state the client is in, and the other way around also applies. Thus, the server and the client are both able to comprehend any message that they are in receipt of, even without perceiving prior messages. This is accomplished by way of using *resources*, instead of *commands*. Resources define literally anything that the user wants to keep or transmit to other services. REST systems do not have to depend on the employment of interfaces, on account of their communication method; it is through standard actions on resources.

These constraints assist RESTful apps in reaching reliability, fast performance, and scalability, as elements that can be controlled, brought up to date, and reiterated without disturbing the system in the course of its operation.

Information exchanging between client and server

In REST, it is necessary for a client to make a request to the server to get back or alter data on the server. A request is mainly made up of:

- an HTTP verb, which decides what type of action to carry out. In REST, to connect to the resources, we apply 4 elementary HTTP verbs:

  1. GET - get a particular resource, or an array of resources

  2. POST - construct a new resource

  3. PUT - amend a particular resource

  4. DELETE - delete a particular resource

- a *header*, which permits the client to transfer extra information together with the request. In my application, I use it to pass the token for authorization.

- a path to a resource. In a request, there shall be a path to the resource that the action should be carried out on.

- a noncompulsory request body including data[6]

3.2   Library and Framework

3.2.1   Overview

JavaScript and Java, two of the most well-known programming languages, are used during the whole of the application development, with JavaScript being used to make the front-end, and Java is used for the back-end. JavaScript is an all-powerful language that can do many things for a website, like powering the site's common interactivity. It allows the site to have extra services that is not alternatively doable with just HTML and CSS. It favors the webpages to "answer" the actions of the users and vigorously update themselves. Although my application is a multipage web app, there also are functions that can work on one page, and JavaScript makes it possible for a function to complete without asking for a page refresh.[7]

For the back-end, I used Spring Boot. Before discussing Spring Boot, first, we have to look at the definition of the Spring Framework. It is a Java platform that brings thorough infras-

tructure backing for developing Java apps. It manages the infrastructure so the developer can concentrate on the app itself.[8]

Spring Boot is a Java-based framework used to develop a stand-alone and production-grade Spring application that you can run with least configurations, compared to the complicated configuration setup of the Spring framework. Spring Boot has many advantages, and the followings are the ones that made me pick Spring Boot as my first choice in the process of developing the back-end of the application:

- It stays away from the complicated XML configuration of the Spring Framework.

- It is effective in controlling REST endpoints.

- It presents annotation-based Spring application.

- It helps dependency control.[9]

```java
// create a multiple choice question
@CrossOrigin
@PostMapping("/createmultiplechoicequestion")
public @ResponseBody HashMap<String, String> createMultipleChoiceQuestion(
        @RequestHeader("Authorization") String token, @RequestBody String json) throws Exception {
    JwtUtil jwtUtil = new JwtUtil();
    HashMap<String, String> map = new HashMap<String, String>();
    if (jwtUtil.validateRole(token, 2)) {
        sql s = new sql();
        ObjectMapper mapper = new ObjectMapper();
        try {
            Map<String, String> json_map = mapper.readValue(json, new TypeReference<Map<String, String>>() {
            });
            String name = json_map.get("name");
            String className = json_map.get("className");
            String question = json_map.get("question");
            String A = json_map.get("A");
            String B = json_map.get("B");
            String C = json_map.get("C");
            String D = json_map.get("D");
            int time = Integer.parseInt(json_map.get("time"));
            String solution = json_map.get("solution");
            if (s.createMultipleChoiceQuestion(name, className, question, A, B, C, D, time, solution)) {
                map.put("Success", "Created question successfully");
            } else {
                map.put("Failed", "Can't create question");
            }
        } catch (Exception e) {
            e.printStackTrace();
            map.put("Failed", "Can't create question");
        }
        s.closeConnection();
    } else {
        map.put("Failed", "Unauthorized");
    }
    return map;
```

Figure 3.2: An endpoint coded with Spring Boot

31

3.2.2 jwt-decode

jwt-decode is a tiny browser library that assists in the decoding of JWTs tokens which are Base64Url encoded.[10] I used it in the front-end to decode the JWT received from the server after logging in.

3.2.3 WebRTC

WebRTC is a project that can be used to add the abilities of real-time communication to the application.[11] For my project, I used it for the students to send voice and video to the teacher.

3.2.4 Apache Commons Codec

This package consists of simple encoders and decoders for numerous algorithms.[12] I used this library to generate the signature for the JWT.

3.2.5 MySQL Connector/J

This library was used to connect the back-end with the MySQL database.

3.3 Data model

Below is the Entity-relationship diagram of my application's database. There are 6 entities in the database:

- Class

- Role

- User

- State

- MultipleChoiceQuestion

- MultipleChoiceAnswer
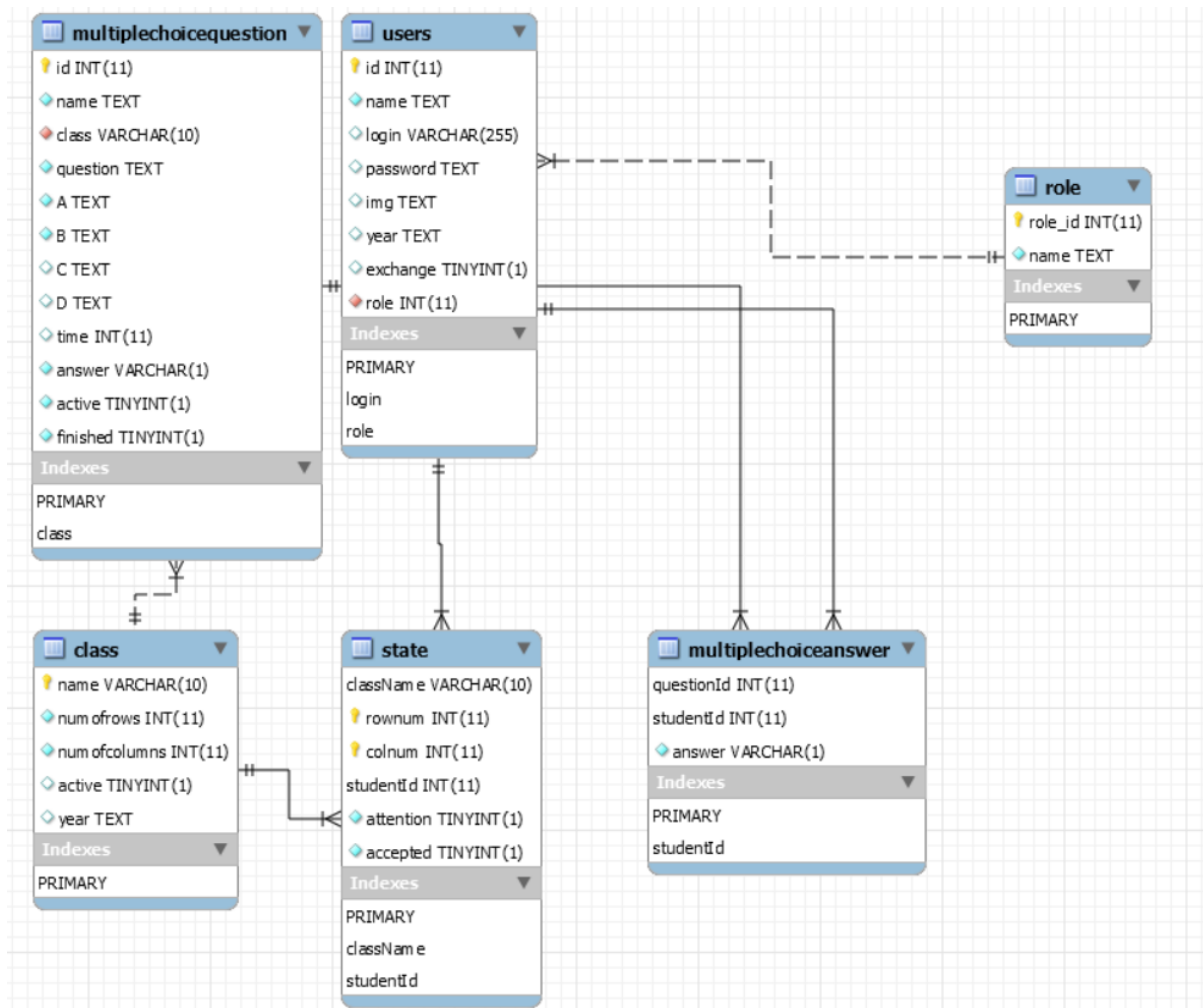
3.3.1 Class

This entity is to represent a real life classroom. It will have a **name** as its primary key. As stated before, a class seat map is represented by a rectangle **numofrows** and **numofcolumns** are the two attributes to form that rectangle. The boolean attribute **active** is to see that if the class is active or not. The attribute **year** is to show that which intake is taking the class.

3.3.2 Role

This entity is to represent the roles that a user can have. It has an integer named **role-id** acting as the primary key, and an attributed named **name** to hold the name of the role. In my application, there are only 3 role ids for 3 roles: 1 for Administrator, 2 for Lecturer, and 3 for Student.

3.3.3 Users

This entity is to represent a user. An integer named **id** is the primary key. The attributes of this entity are **name** for the name of the user, **login** and **password** for the credentials, **img** for the path of the image of that user stored in the server, **year** is for the intake of that user (if the user is a student), **exchange** is a boolean to check that if the student is an exchange student or not, **role** is a foreign key referencing **role-id** of the table **Role**

3.3.4 State

"State" is a concept that I created to define the connection between a student and a seat in a classroom. The primary keys are the **className**, the **rownum** and the **colnum**, which represents a seat in a classroom. **className** is also a foreign key, referencing attribute **name** of the table **Class**. The attribute **studentId**, which is a foreign key referencing the attribute **id** of table **Users**, is to represent the student sitting on the seat. The boolean attribute **attention** is to check that if that seat is calling for attention or not, and the boolean attribute **accepted** is to check that if the attention call from that seat is accepted by the lecturer or not.

3.3.5 MultipleChoiceQuestion

As stated above, in this application, a quiz is defined to consist of a single multiple choice question. **id** is its primary key, and the attribute **class**, which shows the class that that quiz belongs to, is a foreign key referencing attribute **name** of the table **Class**. The other attributes of the quiz are **name**, **question**, **A**, **B**, **C**, **D**, **time** for the time that the students have to answer the quiz, **answer** is for the solution of the quiz, **active** is a boolean to check that if that quiz is on or not, **finished** is also a boolean, and it is used to check that if that quiz is finished or not.

3.3.6 MultipleChoiceAnswer

This entity is to represent the answer of a student to the above quiz. The primary keys are **questionId** and **studentId**, which are also foreign keys that respectively referencing **id** of **MultipleChoiceQuestion**, and **id** of **Users**. The remaining attribute is **answer**, which shows the answer that a student made for a question. It could be A, B, C, D, if the student could answer the question, or N, if the student had no answer.

3.4 API documentation for the back-end

3.4.1 Request

Except for logging in, a request should always contain an Authorization header. If it is a POST or PUT request and the request body is in JSON, then there should be a Content-Type header.

Authorization	Bearer (token)
Content-Type	application/json

3.4.2 Authorization

POST /api/login

Request body:

`@login`	string	Your username
`@password`	string	Your password

Response:

`@token`	string	Attach this to your Authorization header for the subsequent requests

3.4.3 Endpoints

GET /api/showclass

This API is to show all classrooms that are available. All roles can access this endpoint.

Response: The response will be a JSON Array containing objects of classes. A class object, in this endpoint, will contain:

`@name`	string	Name of the classroom.

Actually, the response will also show other attributes of a classroom, but they are all set to 0 or null, since for this endpoint, the only things that matters are the name of the classroom. Responses of other endpoints will also work this way; it will return all attributes of an object, but the unnecessary attributes will be set to 0 or null.

POST /api/uploadFile

This API is for the administrators to upload the image of a student to the server. Only the administrator can access this endpoint.

Request parameter:

`@file`	file	The image of a student

Response:

`@fileName`	string	Name of the file
`@fileDownloadUri`	string	The path of the file in the server
`@fileType`	string	Type of the file
`@size`	int	Size of the file

POST /api/createclass

This API is for the administrators to create a class. Only the administrators can access this endpoint.

Request body:

`@name`	string	Name of the class
`@row`	int	Number of rows (used to form the rectangle representing the class map)
`@col`	int	Number of rows (used to form the rectangle representing the class map)

POST /api/createstudent

This API is for the administrators to create a student. Only the administrators can access this endpoint.

Request body:

`@name`	string	Name of the student
`@img`	string	The path of the file in the server
`@year`	string	The intake of the student
`@login`	string	The username of the student
`@password`	string	The password of the student
`@exchange`	boolean	Is this an exchange student?

GET /api/class

This API is to get information of a single class. The lecturers and students can access this endpoint

Request parameter:

`@name`	string	Name of the class

Response:

`@name`	string	Name of the class
`@rows`	int	Number of rows (used to form the rectangle representing the class map)
`@cols`	int	Number of columns (used to form the rectangle representing the class map)
`@active`	boolean	Is this class active?
`@year`	string	The intake taking the class

GET /api/state

This API is to get the states of a class. The lecturers and students can access this endpoint

Request parameter:

@name	string	Name of the class

Response: The response will be a JSON Array of state objects. A state object will look like this:

@row	int	The row-coordinate of the seat
@col	int	The column-coordinate of the seat
@student	int	The ID of the student who is taking the seat

GET /api/student

This API is to get the information of a student. Only the lecturers can access this endpoint

Request parameter:

@id	int	ID of a student

Response:

@name	string	Name of the student
@img	string	The path of the image of that student in the server
@year	string	The intake of the student
@exchange	boolean	Is this an exchange student?

PUT /api/activeclass

This API is for the lecturers to activate a class. Only the lecturers can access this endpoint

Request parameter:

@name	string	Name of the classroom
@year	string	The intake that will take that class

PUT /api/deactivateclass

This API is for the lecturers to deactivate a class. Only the lecturers can access this endpoint

Request parameter:

@name	string	Name of the classroom

POST /api/createmultiplechoicequestion

This API is for the lecturers to create a quiz. Only the lecturers can access this endpoint

Request body:

@name	string	Name of the quiz
@className	string	Name of the class that the quiz belongs to
@question	string	The question of the quiz
@A	string	Answer A
@B	string	Answer B
@C	string	Answer C
@D	string	Answer D
@time	int	Time allotted for the quiz
@solution	string	The character representing the right answer of the quiz

GET /api/average

This API is for the lecturers to get the average result of a class on a quiz. Only the lecturers can access this endpoint.

Request parameter:

@questionid	int	ID of the quiz

Response:

@right	string	right answers/total number of students
@wrong	string	wrong answers/total number of students
@no	string	students who had no answer/total number of students

GET /api/finishedquestion

This API is for the lecturers to see the finished questions that belong to a certain class. Only the lecturers can access this endpoint.

Request parameter:

@className	string	Name of the class

Response: The response will be a JSON Array of quiz objects. A quiz object will look like this:

@id	int	ID of the quiz
@name	string	Name of the quiz

GET /api/activequestion

This API is to show the active questions that belong to a certain class. The lecturers and students can access this endpoint.

Request parameter:

@className	string	Name of the class

Response: The response will be a JSON Array of quiz objects. A quiz object will look like this:

@id	int	ID of the quiz
@name	string	Name of the quiz

GET /api/inactivequiz

This API is to show the inactive questions that belong to a certain class. The lecturers and students can access this endpoint.

Request parameter:

@className	string	Name of the class

Response: The response will be a JSON Array of quiz objects. A quiz object will look like this:

@id	int	ID of the quiz
@name	string	Name of the quiz

GET /api/questionname

This API is for the lecturers to get a name of a single quiz. Only the lecturers can access this endpoint.

Request parameter:

@questionid	int	ID of the quiz

Response

@name	string	Name of the quiz

POST /api/activequiz

This API is for the lecturers to activate a quiz. Only the lecturers can access this endpoint.

Request parameter:

`@questionid`	int	ID of the quiz

GET /api/question

This API is to show the information of a quiz. The lecturers and students can access this endpoint.

Request parameter:

`@questionid`	int	ID of the quiz

Response:

`@name`	string	Name of the quiz
`@className`	string	Name of the class that the quiz belongs to
`@question`	string	The question of the quiz
`@A`	string	Answer A
`@B`	string	Answer B
`@C`	string	Answer C
`@D`	string	Answer D
`@time`	int	Time allotted for the quiz
`@solution`	string	The character representing the right answer of the quiz

GET /api/quiztime

This API is to show the remaining time of a quiz. The lecturers and students can access this endpoint.

Request parameter:

`@questionid`	int	ID of the quiz

Response:

`@time`	int	The remaining time of the quiz

GET /api/quizreview

This API is to the information of a quiz, and the result of a particular student on that quiz. The lecturers and students can access this endpoint.

Request parameter:

`@questionid`	int	ID of the quiz
`@studentid`	int	ID of the student

Response:

`@name`	string	Name of the quiz
`@question`	string	The question of the quiz
`@A`	string	Answer A
`@B`	string	Answer B
`@C`	string	Answer C
`@D`	string	Answer D
`@solution`	string	The character representing the right answer of the quiz

GET /api/answerbyseat

This API is for the lecturers to see the answers of the student by seats. Only the lecturers can access this endpoint.

Request parameter:

`@questionid`	int	ID of the quiz

Response: The response will be a JSON Array containing objects of AnswerBySeat. An AnswerBySeat object, in this endpoint, will contain:

`@name`	string	Name of the student
`@img`	string	The path for the image of that student in the server
`@row`	int	The row-coordinate of the seat that that student is sitting in
`@col`	int	The column-coordinate of the seat that that student is sitting in
`@answer`	string	The answer of that student
`@solution`	string	The solution of the quiz

PUT /api/callattention

This API is for the students to call for attention. Only the students can access this endpoint.

Request parameter:

`@studentid`	int	ID of the student

PUT /api/closeattention

This API is to close the call for attention. The lecturers and students can access this endpoint.

GET /api/checkattention

This API is for the lecturers to check if any student is calling for attention or not. Only the lecturers can access this endpoint.

Request parameter:

@classname	int	Name of the class

Response:

@row	int	Row-coordinate of the student who is calling for attention
@col	int	Column-coordinate of the student who is calling for attention
@studentId	int	ID of the student who is calling for attention

PUT /api/setaccepted

This API is to modify the state of acceptance for a call for attention. The lecturers and students can access this endpoint.

Request parameter:

@turn	string	Only two values are allowed: "on" or "off". "On" means that the call is accepted, "off" means that the call is done and not accepted anymore

POST /api/createstate

This API is for the students to create states. Only the students can access this endpoint.

Request body:

@class	string	The name of the class
@row	int	Row-coordinate of the seat that that student wants to sit in
@col	int	Column-coordinate of the seat that that student wants to sit in
@student	int	ID of the student

Response:

@Result	string	Result of the state creation process.

POST /api/createmultiplechoiceanswer

This API is for the students to create an answer for a quiz. Only the students can access this endpoint.

Request body:

@questionId	int	The id of the quiz
@studentId	int	The id of the student
@answer	string	The character representing the answer of the student

GET /api/quizdone

This API is to show the quizzes done by a student. Only the students can access this endpoint.

Request parameter:

@studentid	int	ID of the student

Response: The response will be a JSON Array of quiz objects. A quiz object will look like this:

@id	int	ID of the quiz
@name	string	Name of the quiz

GET /api/checkaccepted

This API is for a student to check that if their call for attention is accepted or not. Only the students can access this endpoint.

Request parameter:

@studentid	int	ID of the student

Response:

@Result	boolean	Is the call for attention accepted?

Chapter 4

Conclusion

4.1 Summary

The thesis introduces the progress of my project in an effort to develop a tool that can make interaction between lecturers and students better. To recap, the main features of the application are the class map, the quiz and the "attention call".

In addition to that, this paper also summarizes my first experience in developing a whole web application from back-end to front-end. My supervisor, Mr. Clavel told me to limit the use of frameworks, since they force us to code in ways that are different to our coding style, and personally, I agree with him. For that reason, except for Spring Boot, which I am very familiar with, I did not use any other frameworks, although there are many famous frameworks for front-end development like React or Vue. This may made the coding process slower, but it truly helped me to control my application better, because I wrote every line of code myself, so I completely know how my app works. For me, I have attained good outcomes with the purpose of putting a tool that enhances classroom interaction. I pushed the source code on GitHub, and it is also deployed on Heroku.

4.2 Future works

There are various functions that I firstly intended to add to my application, but due to the 2-month deadline, I did not have time to finish them. Those are:

- Students can send to teachers anonymous assessments.

48

- The students can follow the teacher's slide on their own screens. This is ideal for large classrooms, since looking at the projector is a bit hard in those classes.

- An online whiteboard that can be zoomed in and out.

References

[1] *What is HTML?* URL: `https://www.w3schools.com/whatis/whatis_html.asp`.

[2] *What is CSS?* URL: `https://www.w3schools.com/whatis/whatis_css.asp`.

[3] *What is JavaScript?* URL: `https://www.w3schools.com/whatis/whatis_js.asp`.

[4] Goldy Benedict Macquin. *Single Page Applications vs Multiple Page Applications — Do You Really Need an SPA?* URL: `https://medium.com/@goldybenedict/single-page-applications-vs-multiple-page-applications-do-you-really-need-an-spa-cf60825232a3`.

[5] *Introduction to JSON Web Tokens*. URL: `https://jwt.io/introduction/`.

[6] *What is REST?* URL: `https://www.codecademy.com/articles/what-is-rest`.

[7] Kevin Schroeder and Ken Sugiura. *What Is JavaScript Used for in Front-end Web Development?* URL: `https://www.hilemangroup.com/Thought-Leadership/Hilelights-Blog/JavaScript-and-Front-end-Development`.

[8] *1. Introduction to Spring Framework*. URL: `https://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/html/overview.html`.

[9] *Spring Boot - Introduction*. URL: `https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm`.

[10] auth0. URL: `https://github.com/auth0/jwt-decode`.

[11] URL: `https://webrtc.org/`.

[12] URL: `https://github.com/apache/commons-codec`.