Software Workshop (06 - 26955 / 06 - 26956)
Small Team Project
Campus Attendance & Announcement System
Group 33

Group details:

Group 33
Members: Rebekah Lapham, 1411526
Oliver Hayes, 1521601
Zibo Wang, 1919180
Junlin Li, 1889232
Shuqin Feng, 1954406

Table of contents

1. Introduction

Inspired by the university Canvas system, our group project is to design and create a piece of software for; recording student's attendance at lectures, creating and viewing announcements, commenting on received announcements and viewing other user's comments. The aim of our system is to provide convenient communication between lecturers and students by using this system to save some redundant lecture processes and reduce the time costs.

The user's login details correspond to being either a student or lecturer and each type of user should allow for different functionality. An error message will occur if the user's login details cannot be matched with those stored in the database and the user will be given unlimited attempts at logging in. In contrast, the lecturer client will have greater control on the system than the student client, students only need to complete the given tasks from lecturers.

As a lecturer, once logged in the list of courses they are teaching will be visible. They can select a course and will then be presented with a list of possible actions. These involve:

- Creating and releasing an attendance code
- Deleting an unreleased code
- Creating and releasing announcements

As a student, once logged the list of courses they are taking will be visible. They can select a course and will be presented with a list of possible actions. These involve:

- Inputting an attendance code if possible
- View announcements
- GIve comments on received announcements
- View other user's comments

For the project, a database has been created to store user login details, date information, attendance codes, announcement contents and the user comments contents. This database will be accessed via the server. When the user tries to execute an operation, by clicking the button on the GUI, a data request will be sent to the client, and client will transfer it to the server. When the server get the requested data from the database, it will send the requested data back to various clients. The relationship between the client, server, database and the GUI is simply shown in the picture [1] below.

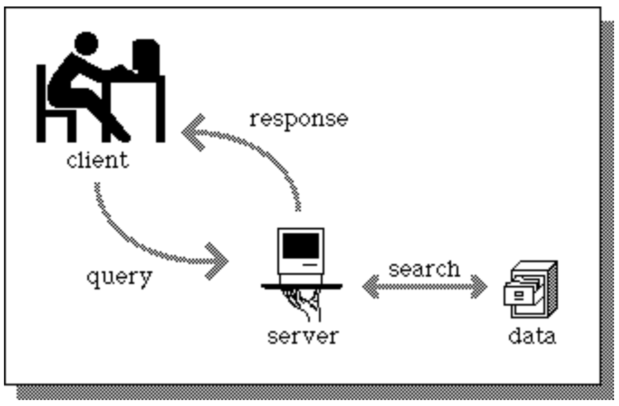


Figure 1

In this project, we will keep track of our work and allow for easy adjustment of code using an SVN server. The team member contribution table and the weekly agenda sheets are included into the appendix. In the following report, implemented methods for various functions will be presented in detail.

2. System preconditions

- All the students and lecturers have been pre-allocated an account, and the account information is recorded in the database. All lecturers' usernames start with an "L" so the system knows they are a lecturer when they login. The students' usernames start with an "S". The usernames are automatically generated and will be unique, meet the required length and will not begin with a digit. The passwords are also automatically generated and are unique are secure.
- Courses lists have been pre-allocated to each student and lecturer. This information is stored in the database, and when the user successfully logs into their account they will be able to view it on profile page.
- The course lists content cannot be extended or deleted by the user.
- Information about each user has been stored into the database, such as attendance date, course name, personal username, user type. These information can be viewed from the profile page.
- Users cannot reset their usernames or passwords.
- One account can only be logged in on one device at same time.
- The user has infinite attempts to enter their login details correctly.
- The lecturer will give the attendance code to students in class to fill out attendance.

3. Detailed functional requirements

System functional requirements are discussed as below. These are the detailed operations which can be truly implemented by users in the future.There are two types of client, Student and Lecturer. So, for some specific operations, like attendance and announcement operations will be separately explained for both user types.

3.1 Login operation (common)

Common user

1. Users open the software, a login window is shown with username and password input fields, and two option buttons - "clear" and "login".
2. If the login failed, a warning window will pop up with a message stating that the login attempt was unsuccessful.
3. If the login process is successful, the user will enter into the user personal profile page, which will show a list of the courses they are currently enrolled on.

3.2 User profile (common)

Common user

1. "USERTYPE:" will be the main title of the page.
2. User is able to see a list of courses they are enrolled in on the profile page.
3. Users can select a course and then choose from the two options 'announcement' and 'attendance'. They will then be shown the appropriate pages for their selected function.
4. Users click the logout button, to exit the application.

3.3 Attendance Operation (student/lecturer)

a. Student user

1. Student clicks the attendance button on profile page. If there is an available attendance at the time of clicking a pop up window will appear. Students can see their usernames in the upper right corner of the page
2. "Attendance" will be the main title of the page, and the course will be shown in the top right corner
3. The student will be presented with a box to enter the attendance code.
4. If the attendance code has been input correctly, the student will return to the previous page displaying their courses.

b. Lecturer user

1. When the lecturer clicks the attendance button, they will be taken to the attendance page.
2. "Attendance" will be the main title of the page.
3. The information of each course, the date, time duration, attendance status and the specified attendance code are stored in one bar, and they are connected one by one as a list.
    a. *Date: the released date of the attendance*
    b. *Time: the available time duration for the attendance*
    c. *Attendance code: code inputter by lecturer*
4. The attendances will appear in the order they are created.
5. Lecturers can click the "create new attendance" button to create a new attendance case.
6. Clicking the button, a new page for creating a new attendance code will appear, with four information input boxes for date, time duration, and the attendance code, one button for submission, one button for resetting the fields and a cross button for closing the page are displayed.
7. Clicking the "submission" button, the page closes and a new attendance case has been added to the list assuming details are correct.
8. If details have been input incorrectly the fields will be reset to blank.
9. Lecturers cannot modify attendances once created. If a mistake is made they must delete the attendance and create a new one.
10. The lecturer can only delete assuming a student has not attended their attendance.
11. Clicking the back button will take the user back to the attendance main page.

3.4 Announcement Operation (student/lecturer)

a. Student user:

1. Clicking the announcement button for the selected course will generate a new page for further announcement action.
2. The main page title is "Announcement".
3. Students can see the course name and course ID at the upper right corner of the page.
4. The received announcements are divided into two lists with respect to unread and read, students can press "read" button to read the these announcements in unread list. They can also press the "review" button to read the announcement

from read list. The more recent announcements are displayed at the button of the list.

5. Clicking on the back button, takes the student back to the announcement main page.

b. Lecturer user

1. Clicking the “announcement” button for the selected course will generate a new page for further announcement action.
2. The main page title is “Announcement”
3. Lecturers can see the course name, course id and the reading people number at the upper right corner of the page.
4. Lecturers can see the a list of sent announcements on the page. The most recent announcements will be at the button of the list.
5. Clicking the “add” button will generate a new window with one empty text input box and two buttons. One button is for releasing the announcement, one button is for closing the page.
6. The lecturer cannot save a draft for the announcement, if they close the window, they have to start a new case again.
7. Clicking the “submit” button, will send the announcement to the students enrolled on the selected course, then window close.
8. Clicking on the back button, lecturer back to the profile page.

3.5 Comment Operation (student/lecturer)

a. Student user:

1. Students select a announcement from list and press button on announcement content review page, enable them see comments list below the announcement content frame.
2. Students can see the comment-sent users’ username and comment time on the comment case in the list.
3. Student can press the “open” button to review the content of the selected comment in a new open window.
4. In the new open window, student can see the course name cid at the upper right corner, and the “Comment” is the main title of the window.
5. “Student name + comment time” is the title of the reading frame, student can read the comment from frame.
6. Students click the “back” button on comment-review window to close it.
7. Students click the “comment” button to submit comment for the announcement in existing window, then a new window will be opened.
8. In create comment window, student can edit content in a empty reading frame, then clicking “submit” button to submit or clicking the “clear” button to empty content in reading frame.
9. By pressing “submit” button, comment window closes.

b. Lecturer user

1. Lecturers select a released announcement case from list and press button on announcement-content-review page, enable them see comments list below the announcement content frame.

2. Lecturers can see the comment-sent users' username and comment time on the comment case in the list.
3. Lecturers can press the "open" button to review the content of the selected comment in a new open window.
4. In the new open window, lecturers can see the course name cid at the upper right corner, and the "Comment" is the main title of the window.
5. "Student name + comment time" is the title of the reading frame, lecturers can read the comment from frame.
6. Lecturers click the "back" button on comment-review window to close it.

4. User case diagram

The user case diagram for the Campus Announcement & Attendance system is shown in the appendix 1. The actors that have been identified are shown outside of the system boundary, which is denoted by the large rectangle (solid black line). The actor on left side is the user including client and GUI, other side are the server and database as actors. The main use cases are shown with their relationships to the main actors and each other.

5. Activity diagram

The "student login" and "student announcement-comment" two user cases are selected for further analysis using activity diagrams, which are all presented in the appendix 2. The activity diagram 1 is based on the scenario 3 and use case 3.

1. The user has already logged in the system by a student's username.
2. The user has selects one course in the home pages.
3. The user has already entered the announcement list interface of certain course.

The user opens his/her application, enters his/her username and password, and taps the login button. The client will send a login request to the server, which will acquire the user's authentication data from database and then test whether those two set of data match. If the authentication has failed then client will display login failure message and the user reads the failure message and retypes his/her username or password. If the authentication has succeed, the server begins to load the user's account information generated from database.

While student clicks the "attendance" button to do the attendance function for a selected course, the client sends the request, personal information and course information to server, which checks if the attendance action is available at this moment. If the response is "No", the attendance warning GUI is activated to show the caution, user clicks the "ok" button to close the existed warning window and send request for showing the user profile page again, server gets the request, then selecting course list from database and sending it back to client, client will initialize the user profile page and show it to user. If the response is "Yes", the client initializes the student do attendance page. User inputs the attendance code then clicking "submit" button, if the attendance code is correct, database records the successful attendance of the student and send it back, client get message and initialize the user profile page. If user inputs the wrong attendance code, GUI gives a warning notification in code input text field, and get into the next attendance loop.

6. Documented use cases

6.1 Scenario 1

Michael teaches Cryptography, Software Workshop and Spanish in the university. He has an attendance set for Spanish for next week but he has moved the lecture time to tomorrow. He need to delete the attendance in next week and create a new attendance for tomorrow. He logs in Attenda by his staff username and password. He chooses Spanish course and he clicks “attendance”. Opening the page of adding a new attendance, he fills in the time and code and then submit it. It resets the text boxes and doesn’t submit the attendance.The code is required to be exact six characters. He rewrites the attendance using a correct code length. Then the interface turns back to the previous page automatically and he selects the attendance for the course in next week and deletes it. Finally, Michael clicks “log out”. The detailed table is shown in the appendix 3.

6.2 Scenario 2

After the teacher shows the attendance code in the class, student Linda logs in Attenda by her student username and password. She selects Cryptography course and clicks “open”. And then Linda clicks “attendance”. It shows a notification that there is no attendance at the moment. She waits for nearly three minutes till the beginning time of the course. This time, it shows the interface to fill in the code. Linda inputs the code given by Michael and submits it. The page turns back to the course page. Linda logs out and start to concentrate on the course. The detailed table is shown in the appendix 3.

6.3 Scenario 3

Tom is a student in the university. His classmates talk about a new announcement with him. It’s about an illustration for the latest assignment of Physics. He decides to see the details of the announcement. Tom logs in Attenda by his student ID number and password. He selects Physics course and clicks “open”. And then Tom clicks “announcement”. He saw one announcement in the unread announcements list. He opens it and reads it. Tom still has some question about the announcement. He then turns back to the previous page and opens one comment which is opened by lots of people. He finds his question is not mentioned in the comment. He then turns back to the previous page and opens other people’s comment but still doesn’t see the similar question as his. Tom decides to post a comment himself. He clicks “comment” and write his question in the new corresponding place. Then he clicks “submit” and clicks “log out” after the interface turns to the announcement page. The detailed table is shown in the appendix 3.

6.4 Scenario 4

Tom’s Physics teacher logs in Attenda and chooses Physics course to see the announcement. He opens the newest announcement and notices lots of students gave comments to it. He selects one comment and opens it. After reading it, he clicks “back”and open some other students’ comments one by one. He thinks it’s necessary to post a new announcement to answer students’ questions. So he goes back to the announcement list interface and clicks “add”. After writing and submit the announcement successfully, he clicks “log out”. The detailed table is shown in the appendix 3.

6.5 Scenario 5

Two users can log-in at the same moment. The detailed table is shown in the appendix 3.

6.6 Scenario 6

One user try to login the system if his/her account is still logged in elsewhere. The detailed table is shown in the appendix 3.

7. Class diagram

As shown in the appendix 4, the class diagram is used to describe the structure of the system by showing all classes from the system. Owning to the classes quantity, the frame is not be able to include all of them. So, each class's attributes and operations are hidden, only show the relationship lines among objects.

8. System design

8.1 Client design

The client for this project contains all interfaces, corresponding controller classes and a Client class. Once a student or lecturer begins using this software, the Client class, which contains a Socket, is instantiated and used in every controller classes for message delivery between client and server. Once an operation(e.g click button) is made by users on the interface, the Client object will send the request and corresponding information to the server and wait for a reply. After the Client object gets reply from the server, it will pass this reply to a controller class and a new interface will be constructed with information in this reply by the controller.

- Client: mainly contains the socket for communicating with the server
- Protocol:contains labels for identifying a request

8.2 Server design

The server for this project contains two threads: a Server class and a CheckClient class. In the Server class, a ServerSocket object keeps waiting for connection from client Socket by an accept() method in a while loop. Once a client is connected with the server, a new thread CheckClilent will start. This thread will keep checking the username and password until they are correct or the connection between server and client ends. After logging in successfully, a user will be put into the Map<username,Socket> as online user and will be removed when he logs out.
For each online user, the corresponding CheckClient thread keeps receiving requests from the client and handling them by sending data back to client from database or just modifying the database.

- CheckClient: checks the username and passwords for logging in and deals with request
- Protocol:contains labels for identifying a request
- Server:keep accepting connections from clients and start new CheckClient thread
- UserConnectionMap:one for students and another one for lecturers,contains account who is online now to avoid that the same account keeps online at the same time.

8.3 Database Design

The below image shows the ER diagram for our database. We created our database tables by identifying the entities of our proposed system and considering how we would like them to interact. For example, given students and lecturers in our system have different functionalities we decided to have them in separate tables. The rest of the ER diagram then followed as we designed each functionality and assigned the primary and foreign keys necessary for each entity.

We designed database functions in DatabaseConnection class. The class contains a constructor for creating the connection to the database and which then has access to a number of defined functions used to access data from our database. This was used in the server part of our application. Upon starting the server it constructs a Database Connection instance which then can be used to respond to requests by the client.

8.3.1 Table entities:

- announcement (annid, time, cid, pid, context)
- attendance (attendid, cid, begintime, endtime, code)
- comments (annid, sid, commenttime, content)
- courses_list (cid, name)
- staff (pid, firstname, lastname, username, password)
- staff_courses (pid, cid)
- student (sid, firstname, lastname, username, password)
- student_attend (attendid, sid, cid)
- student_courses (sid, cid)
- student_read_announcement (annid, sid)

8.3.2 Relationship among server, client and database:

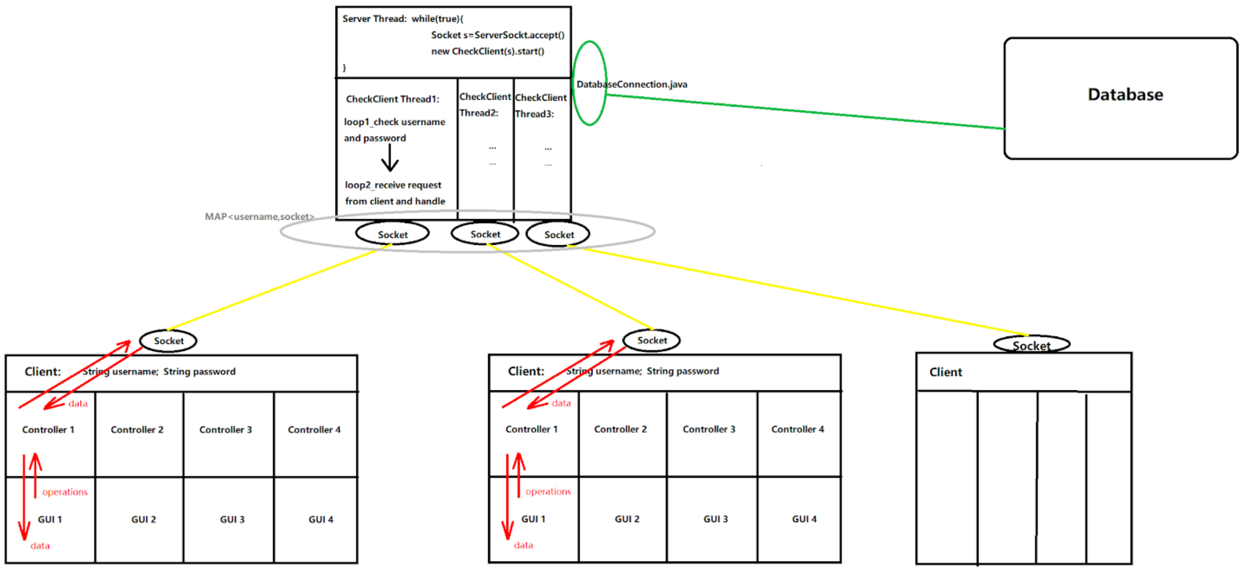


Figure 2 Relationship among server, client and database

8.3.3 ER diagram:

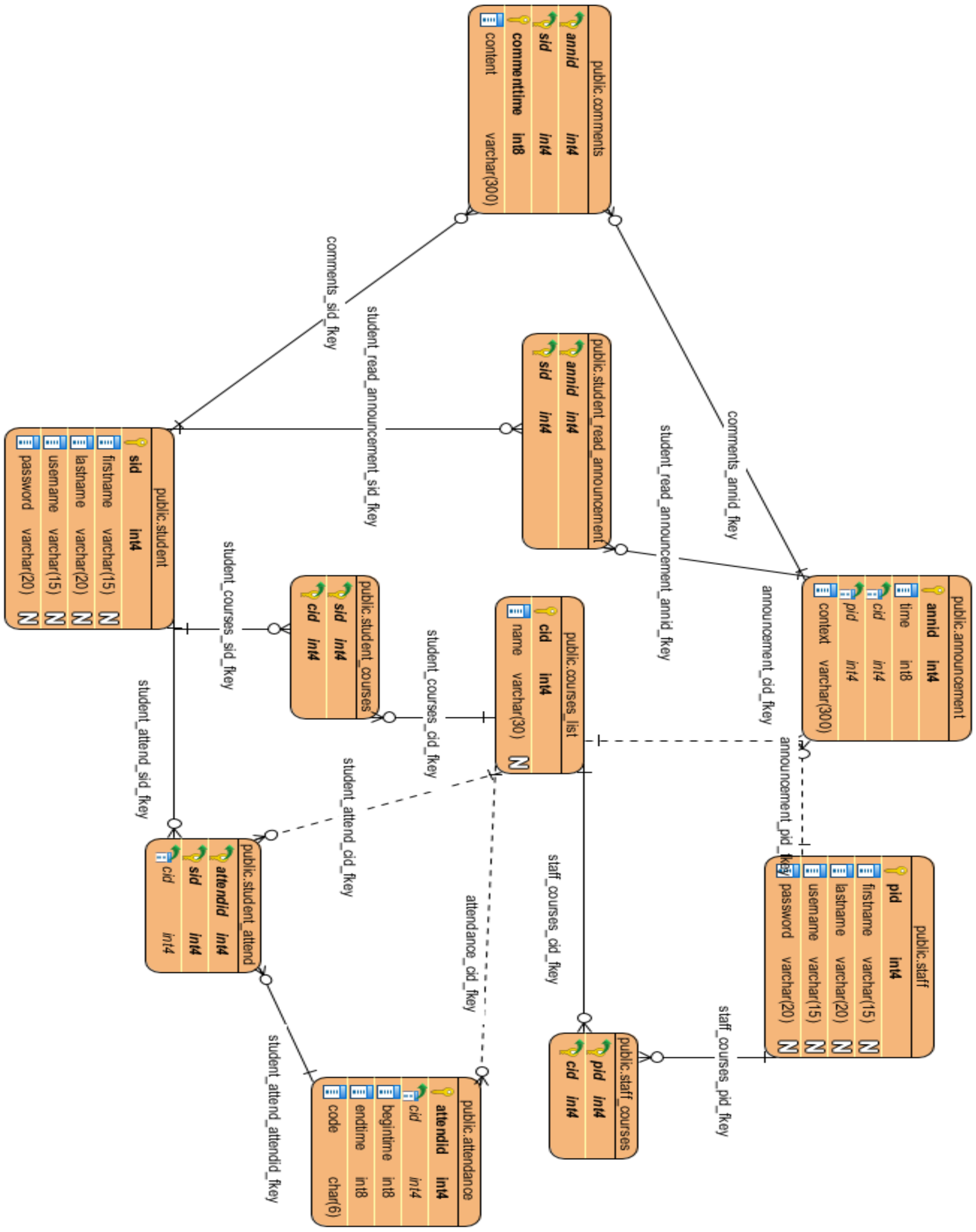


Figure 3 ER diagram for database

8.4 Login function implementation

Below are required functions and detailed implementation methods for these main functions while the GUI is connected with the client, user by clicking the button to send a request to server. For instance, when user is pressing the button to log into the system, server will get these courses list from database then send it back to the client. Thus, client got the response, it would upload them into the view list on the user profile page GUI.

In order to make the overall code structure be more clear, each GUI will correspond to a specific controller class with various of methods, not all be put into the same controller class. This will be easier to make changes and debugs in the later step. If there is a requirement to add more functions, it will also simplify the complexity of the operation. The controller mainly include these methods to be completed after the JavaFx button is activated by user. Each method is regarded as a mini-function action package. Thus, it will be bounded with the correspondent button, aiming to connect the GUI with the client. Basically, most of the methods are based on the interaction between the client, the server and the database. Below are required functions and detailed implementation methods while the GUI is connected with the databases in server:

Required database:

- courses_list (cid, name)
- staff (pid, firstname, lastname, username, password)
- staff_courses (pid, cid)
- student (sid, firstname, lastname, username, password)
- student_attend (attendid, sid, cid)
- student_courses (sid, cid)

Preconditions:

- all the students/teachers have registered an account, and the account information were already recorded into the database
- the users cannot reset their usernames or passwords on the website
- the account can only be logged in only one device
- the account can be unsuccessfully logged in for infinite times
- the students' usernames start with letter "S" and the teachers' usernames start with letter "L"

Common function:

1. **loginButtonSubmit()**: Pressing the "log-in" button, the username and password that a student or a teacher fill in will be submitted to the server and check it with the username and password stored in the database.
2. **LoginPageErrorController()**: If the username or password is incorrect, a warning interface will pop-up.
3. **deleteInput()**: Pressing the "clear" button, the content in the username and password will be deleted.

8.5 Attendance function implementation

Required database:

- attendance (attendid, cid, begintime, endtime, code)
- courses_list (cid, name)
- staff (pid, firstname, lastname, username, password)
- staff_courses (pid, cid)
- student (sid, firstname, lastname, username, password)
- student_attend (attendid, sid, cid)
- student_courses (sid, cid)

Preconditions:

- Assuming attendances will only be shown to students during the available time of every attendance
- Lecturer will give the attendance code to students in class
- Lecturer can’t edit the attendance which has been submitted
- Lecturer cannot delete an attendance once a student attended it

Below are required functions and detailed implementation methods while the GUI is connected with the databases in server:

Teacher client:

1. **attendanceAddButton()**: Pressing the “add” button on an attendance page, a page (lecturer_createAttendance.fxml) which can be used to create a new attendance will be shown. Teachers need to fill in the begin time and end time and code of the new attendance.
2. the “**date + available time + code**” form the name of the attendances.
3. **attendanceDeleteButton()**: Pressing the “delete” button, the selected attendance will be deleted from the available view list (the content in database is also deleted).
4. **submitCreatedAttendanceButton():** Pressing the “submit” button, the information will be stored in the database and the new attendance information will be added into the attendance list. If the format of the time or code that teachers submitted is not correct, notifications will be shown to the teachers.
5. **cancelCreatedAttendanceButton():** Pressing the “cancel” button, the page(lecturer_createAttendance.fxml) for creating a new attendance will be closed.

Student client:

1. **StudentDoAttendanceController():** Pressing the “attendance” button after students choose the lectures, the page(student_doAttendance.fxml) which for students to fill in the attendance code will be shown.
2. **AttendanceWarning():** If there is no attendance at the moment, the page(attendanceWarning.fxml) which tell students that there is no attendance for the lectures at the moment will pop-up. The student then is asked to press “ok” button which will make the interface turn to the previous home page (userProfilePage.fxml).
3. **submitButtonStudentAttendance():**Pressing the submit button, the code will be sent to the server and then check it with the corresponding code. If the code is correct, it will be submitted successfully and the window will be closed. If the code is not correct, the code students fill in will be deleted on the page and students need to fill in it again.
4. **cancelButtonStudentAttendance():** Pressing the “cancel” button, it will turns back to the previous course list page(userProfilePage.fxml).

Common function:

1. backButton(): Pressing the “back” button, it will turns back to the previous page.

8.6 Announcement function implementation

Required database:

- announcement (annid, time, cid, pid, context)
- comments (annid, sid, commenttime, content)
- courses_list (cid, name)
- staff (pid, firstname, lastname, username, password)
- staff_courses (pid, cid)
- student (sid, firstname, lastname, username, password)
- student_courses (sid, cid)

- student_read_announcement (annid, sid)

Preconditions:

- Lecturer can only add new announcement case into the list, but cannot delete any released announcement case from the list
- Lecturer can view the comments from students on a selected announcement, but cannot reply these comment

Lecturer client:

1. **Lecturer_AnnouncementPageController():** the method is is a class constructor of the Lecturer_AnnouncementController class, it is a special method that is used to initialize the objects. It is called when objects are created.
2. **initialize():** it is used to initialize these objects (LectureAnnouncementList, courseDetail) from the correspondent interface page (Lecturer_AnnouncementPage.fxml)
3. **lecturerAddAnnouncement():** the method is used to when the lecturer press the "Add" button, a new interface (lecturer_createAnnouncement.fxml) will be opened for adding a new announcement case
4. **lectureOpenAnnouncement():** the method is used to when the lecturer select a announcement case then press the "open" button, a new interface (lecturer_reviewAnnouncement.fxml) will be opened for reviewing the selected announcement content.
5. **lectureBackFromAnnouncePage():** this method is used to when user press the "back" button on the announcement-review page (lecturer_reviewAnnouncement), the window will close and back to the announcement content page (student_announcementPage.fxml).
6. **getChoosenAnnouncement():** this method is used to when the user select an announcement case, then press another button will enable to do further action based on the selected announcement.
7. **lectureCreateAnnounceController():** the method is is a class constructor of the LecturerCreateAnnouncementController class, it is a special method that is used to initialize the objects. It will be called when objects are created.
8. **initialize():** it is used to initialize these objects (courseDetail) from the correspondent interface page (Lecturer_createAnnouncement.fxml)
9. **lecturerSubmitNewAnnounce():** the method is used to when user press the submit button, then transfer edited announcement content to the database, then close the new announcement submit window(lecturer_createAnnouncement.fxml).
10. **lecturerClearAnnounceContent:** the method is used to clear the input contents in the text area (newAnnouncementContent), which is a initialized input farme
11. **lectureBackFromCreateAnnounce():** this method is used to when user press the "back" button on the new announcement create page (lecturer_createAnnouncement.fxml), the window will close and back to the announcement content page (lecturer_announcementPage.fxml).
12. **LecturerReviewAnnouncementController():** the method is is a class constructor of the LecturerReviewAnnouncementController class, it is a special method that is used to initialize the objects. It will be called when objects are created.
13. **initialize ():** it is used to initialize these objects (courseNameCid, readerCounter, announcementTime, content, comment) from the correspondent interface page (Lecturer_reviewAnnouncement.fxml)
14. **openComments:** this method is used to when user press the "open" button, the selected comment will be opened for review.
15. **backToLecturerAnnouncementList():** this method is used to when user press the "back" button on the announcement review page (lecturer_reviewAnnouncement.fxml), the

window will close and back to the announcement content page (lecturer_announcementPage.fxml).
16. **getChosenComments():** this method is used to when the user select an comment case, then press another button will enable them to do further action based on the selected comment case.

Student client:

1. **initialize():** it is used to initialize these objects (courseNameCid, unreadAnnouncementList, readAnnouncementList) from the correspondent interface page (Student_AnnouncementPage.fxml)
2. **student_AnnouncementPageController():** the method is is a class constructor of the Student_AnnouncementPageController class, it is a special method that is used to initialize the objects. It will be called when objects are created.
3. **readNewAnnouncementButton():** this method is used to when the user press the "read" button on student_announcementPage, user will be able to read the announcement case in a new window (student_reviewAnnouncement.fxml). Then the selected announcement case will be moved into the read list.
4. **openAnnouncementForStudent():** this method is used to when the student select a announcement case then press the "open" button, a new interface (student_reviewAnnouncement.fxml) will be opened for reviewing the selected announcement content.
5. **backToUserProfilePage():** this method is used to when students press the "back" button on the announcement page (student_announcement.fxml), the window will jump back to the user profile page (userProfilePage.fxml).
6. **getChosenReadAnnouncement():** this method is used to when the user select an announcement case from read list, then press another button will enable them to do further action based on the selected case.
7. **getChoseUnreadAnnouncement():** this method is used to when the user select an announcement case from unread list, then press another button will enable them to do further action based on the selected case.

Common function:

1. **ReviewAnnouncementController():** the method is is a class constructor of the ReviewAnnouncementController class, it is a special method that is used to initialize the objects. It will be called when objects are created.
2. **initialize():** it is used to initialize these objects (courseDetail, announcementContent) from the correspondent interface page (ReviewAnnouncement.fxml)
3. **backAnnouncementList():** this method is used to when users press the "back" button on the review announcement page (ReviewAnnouncement.fxml), the window will jump back to the lecturer announcement list page.

8.7 Comment function implementation

Required database:

- comments (annid, sid, commenttime, content)
- courses_list (cid, name)
- staff (pid, firstname, lastname, username, password)
- student (sid, firstname, lastname, username, password)
- student_read_announcement (annid, sid)

Preconditions:

- Lecturer can view the comments from students on a selected announcement, but cannot reply these comment
- Student cannot give personally reply to a received announcement
- Student can view other user’s comments, but cannot reply on these comments
- Student can submit comment on a selected announcement
- Student can submit comments for infinite times

Lecturer client

- **initialize():** it is used to initialize these objects from the correspondent interface page (lecturer_openComment.fxml)
- **LecturerOpenCommentController():** this method is a class constructor of the LecturerOpenCommentController class, it is a special method that is used to initialize the objects. It is called when objects are created.
- **backToAnnouncementContentPage():** this method is used to when user press the back button on the comment-review page, the window will close and back to the announcement content page (student_reviewAnnouncement.fxml).

Student client

- **StudentCreateCommentController():** this method is a class constructor of the StudentCreateCommentController class, it is a special method that is used to initialize the objects. It will be called when objects are created.
- **initialize():** it is used to initialize these objects from the correspondent interface page student_reviewAnnouncement.fxml.
- **backToStudentReviewAnnouncement():** this method is used to when user press the back button on the comment page, the window will close and back to the review-announcement page (it is used to initialize these objects from the correspondent interface page student_openComment.fxml).
- **clearCommentText():** it is used to clear the input contents in the text area (commentText), which is a initialized input farme
- **submitComment():** the method is used to when user press the submit button, then transfer edited comment content to the database, then close the comment-submit window.
- **StudentOpenCommentController():** this is a class constructor of the StudentOpenCommentcontroller class, it is a special method that is used to initialize the object. It is called when objects are created.
- **initialize():** it is used to initialize these objects from the correspondent interface page student_openComment.fxml.
- **backToAnnouncementContentPage():** this method is used to when user press the back button on the comment-review page, the window will close and back to the announcement content page (student_reviewAnnouncement.fxml).

9. GUI design

As shown in the following Figure, the graphical user interface mainly include: the login, the failed login warning, the attendance and the announcement. The initial GUI conceptual design is presented in the appendix. Based on the prototype of this draft design, more features have been improved to make the interaction between the GUI page and the user be more simple and humane, and these pages also looks more aesthetic. The final GUI design is shown in the appendix 8. It's detailed functional requirements on each button were already indicated in the previous content.

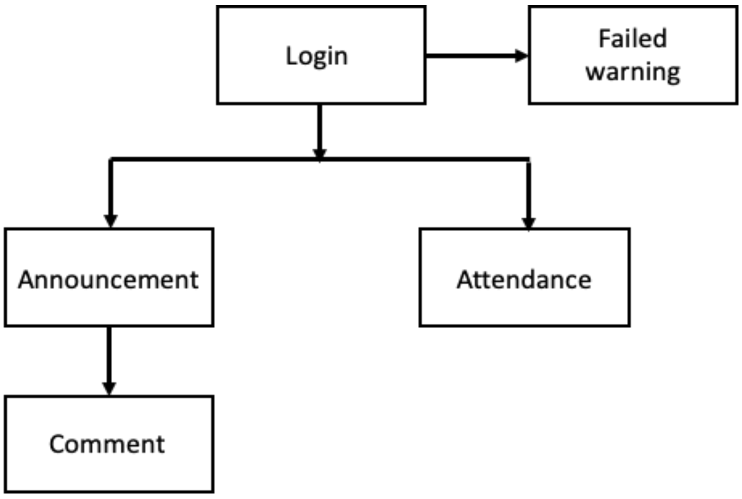


Figure 4 GUI block diagram

10. System Testing

Our plan of testing the system was to produce scenario use cases that we could attempt to implement using the system to see if the outcome was as expected. As described earlier in the report we produced six separate scenarios that would test various parts of the system and you can see in the appendix their corresponding tables with preconditions, flow of events and post conditions. The purpose of these scenarios was to come up with potential tests for our system based on what users may attempt to do. For example we designed the 'date' in the 'add attendance' section to be in the form yyyy/mm/dd others may not do this and as such our system should recognise this. Based on this we produced the following table outlining various tests:

Table 1 Function testing table

Test Number	Summary	Steps/Process	Expected result	Actual result
1	Attempt to login without any input and then with incorrect input.	Open application and click 'login'. Retry with incorrect details	Error message shown twice	As expected
2	Attempt to login with correct details.	Open application and input details and click 'login'	Logged in successfully, opens homepage	As expected

3	As a lecturer, Attempt to create new attendance for a course but input incorrect details. Assume already in homepage.	Select a course and click 'attendance'. Select 'add'. Input details that would be invalid for creating the attendance.	Error message shown, saying invalid input.	As expected
4	As a lecturer, Attempt to create new attendance for a course that is the during the same time as another attendance. Assume already in homepage and another attendance exists.	Select a course and click 'attendance'. Select 'add'. Input details that would have the time crossover with the other attendance	Error message shown, saying invalid input	As expected
5	As a lecturer, Attempt to create a new attendance successfully	Select a course and click 'attendance'. Select 'add'. Input valid details and click 'submit'	Return to previous page with new attendance added	As expected
6	As a lecturer,Attempt to delete an existing attendance. Assume already in attendance and assume selected attendance has no student having attended	Select an attendance and click 'delete'	Returns attendance page with attendance deleted.	As expected
7	As a lecturer,Attempt to delete an existing attendance. Assume already in attendance and assume selected attendance has a student having attended	Select an attendance and click 'delete'	Error message shown saying 'Cannot delete an attendance where a student has attended'	

8	As a student, attempt to attend an attendance for a course before the valid time. Assume in homepage	Select a course and click 'attendance'	Error message showing 'no attendance available'	As expected
9	As a student, attempt to attend an attendance for a course in the valid time, first with an invalid code, then with a valid code and then attempt it again	Select a course and click 'attendance'. Enter invalid code. Enter valid code. Select the same course and click 'attendance'.	Attendance page opens with option to fill in code. On entering invalid code error message occurs. On entering valid code page closes and returns to homepage. On attempting to reopen attendance for that course error message shows 'no available attendance'.	As expected
10	As lecturer, attempt to make an announcement for a course. Assume on homepage.	Select a course and click 'announcement'. Click 'add' and Enter content and click submit	Returns to announcement page with added announcement	As expected
11	As lecturer, attempt to view an announcement. Assume in announcement page	Select announcement and click 'open'	Opens page with announcement available to read	As expected
12	As a student, Attempt to view and comment on an announcement. Assume in announcement list page.	Select announcement and click 'open'. Turn to announcement page. Then click 'comment', enter comment and	Return back to announcement page, and the new comment will show in the comment list.	As expected

		click ‘submit’		
13	As a student, Attempt to read an announcement and its associated comments from other students. Assume in announcement page	Select announcement and click ‘open’. Read announcement and all comments from other students.	Page with announcement and comments shown	As expected

11. System evaluation

Upon reflection of our work, it would have been better to have decided fully on the main functionality that we were aiming to create, before beginning coding. Due to this, we ended up doing a lot of unnecessary work on the main GUI, such as, trying to implement a quiz function which did not meet requirements and would have taken too long to get fully operational. There are also some weaknesses of the system that have been figured out after we fully did the testing on the whole system. They all should be be improved in the future coding work. Some are based on the coding, and some on the GUI design. These weakness are listed as:

- Owning to preconditions, user's password cannot be encrypted, these real passwords data could be viewed from the database, that will increase the risk of user information disclosure
- The comments for announcement will be easier to read, if we place them directly under the corresponding announcement in the order of time rather than placing a list of their titles there, which will show comment content after being opened.
- For comment list, it will be better if students can reply to other students directly. And it will be more user friendly if teachers can reply to students directly or post comments and show it at the top of the comment list.
- Teachers can’t see the details of which students have submitted. If we can add a list of it it’s better.
- It will be more clear for users if we give a notification when students do attendance successfully rather than making the interface turn back to user profile page directly.
- For announcement list, it is easy for us to use time and professorID as title in list. But it will be more clear for user to understand if we use their real title made by lecturers who wrote them.
- Student don’t receive notifications of announcements and attendance, which may make them miss some important information.

It is found that in the time frame we had we were not going to be able to implement a quiz function that we had originally intended on producing. However this is an area we recognise that the application could benefit from if it were to be continued to be developed. The following is showing some conceptual ideas of the implementation for quiz function.

Conceptual quiz function implementation

User type	Functional requirements
Student user	1. Clicking the quiz button will take the user to the quiz page. 2. The main quiz page title is “Quiz for: xxx (student name)” 3. Students can see their usernames in the upper right corner of the page. 4. The main page is divided into two sections: one for available quizzes, and one for unavailable quizzes (expired or graded), both shown in lists type. Each section will be assigned a small tag to show their status (available or unavailable). 5. Available quizzes that have passed the deadline for submission will be moved into the unavailable quizzes list. 6. Available quizzes section: each quiz will be shown as a bar button and connected as a list. The information displayed on the bar are the quiz assigned number, deadline and the maximum points available. a. *Clicking on one of the quizzes loads a new question page.* b. *Each quiz consists of one or more multiple choice questions.* c. *The chosen options will be recorded in the database.* d. *Students can only submit each answer once.* e. *The student can click the submit button, and then will be taken back to the quiz main page.*
Lecturer user	1. Clicking the quiz button will load the quiz page. 2. The main quiz page title is “Quiz designer: xxx (lecturer name)” 3. Lecturers can see their usernames in the upper right corner of the page 4. The main page is divided into two sections, one for available quizzes, the other one for history (finished quizzes), both shown in lists. Each section will be assigned a small tag to show their status (available or history). 5. Lecturer can click the “create a new quiz” button to create a new quiz. 6. Clicking the button takes the user to a page for creating a new quiz. Lecturers can create the quiz content and assign answers for different options, and click “submit” to submit the new quiz. It will then be added into the available quiz list. 7. Lecturers can click the “release quiz” button, to display it for students in lectures. 8. After the quiz is done, lecturers can click the “finish” button, and the quiz will be moved into the history list. 9. Unavailable quizzes section: same as the available quizzes, the displayed information on the bar will be the quiz number, deadline, the given mark (e.g. 3/4) and the marking status (graded or pending) a. *Students can click on each bar button, but there won’t be any response* b. *If the marking status is pending, the given mark will show as “?/4”* c. *If the marking status is graded, the given mark will show as “3/4”* d. *If the lecturer changes the mark, the mark will be recalculated* 10. Clicking the back button will take the user back to the quiz main page.

Conceptual implemented methods for Quiz	
Database utilisation	1. Quiz (cid, date, quizzed, available, title) 2. Student (sid, Firstname, Lastname, username, password) 3. Staff (pid, Firstname, Lastname, username, password) 4. Courses (cid, name)

Preconditions	1. Assuming each course will only have one assigned lecturer, so when the lecturer creates the quiz and submit it into the database, student will know the quiz is from which course 2. Released quiz cannot be edited, deleted 3. Five text area for maximum five questions are pre-settled into the edited window, no add function used for the text area addition.
Lecturer client function	**1. quizAdd()**: pressing the add button on page, lecturer creates a quiz (question, four options, correct answer and the quiz tile) in the popped up window, and press the update button (**quizSubmit()**), the content will be updated into the database (**quizUpdate()**). 2. the "**course name + quiz title + date**" form the name of the quiz and be added into the available view list. **3. quizRelease():** press the release button, the selected quiz is removed from the available view list and be added into the released view list **4. quizDelete()**: press the delete button, the selected quiz is deleted from the available view list (the content in database is also deleted) **5. quizEdit()**: press the edit button, a quiz edit window is popped for the selected quiz (same as the window in 1st function), finish editing the content and repress the update button. (if the title keeps in same, the content in database is recovered by the new)
Student client function	**1. student_quizTake():** pressing the "take the quiz" button, student can only do the selected quiz from the available view list in a popped up window (**student_quizPage.fxml**). (getting the quiz content from the databases, and load them into the pre-settled text area) **2. student_quizSubmit():** pressing the submit button, to update the chosen option into the database. Thus, that quiz is moved from available view list and be added into the released view list.
Common function	**1. quizReview():** press the review button, user can only view the content of the selected quiz (question, four options, correct answer and the quiz tile) in popped up window(**quizAnswerReview.fxml**), they cannot be modified. **2. reviewClose():** press the close button to close the review window **3. backMain():** press the back button, to back to the main profile page

12. Conclusion

We have produced a working system that provides lecturers and students with the ability to communicate via announcements and for students to register attendance for lectures. Although this was the outcome of our project it wasn't how we originally intended it at the outset. As mentioned we had looked at the possibility of implementing a quiz function. However we adapted during the process and changed our plan due to time constraints and the project requirements. The main purpose of our application was to allow for attendance and announcements and as such these features were essential and we needed to ensure that these were working properly before attempting to add additional features.

References

[1] Differencebetween.info. (n.d.). *Difference between Client and Server | Client vs Server*. [online] Available at: http://www.differencebetween.info/difference-between-client-and-server [Accessed 12 Mar. 2019].

Appendix

1. User case Diagram

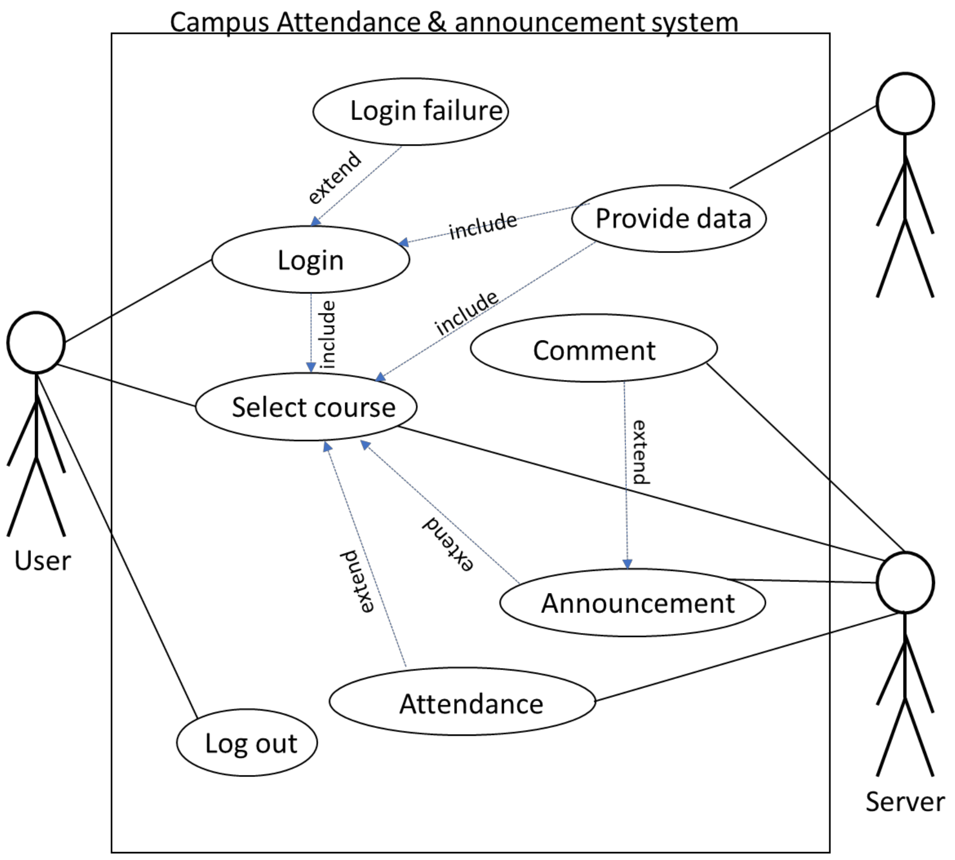


Figure 5 User case digram

2. Activity diagram

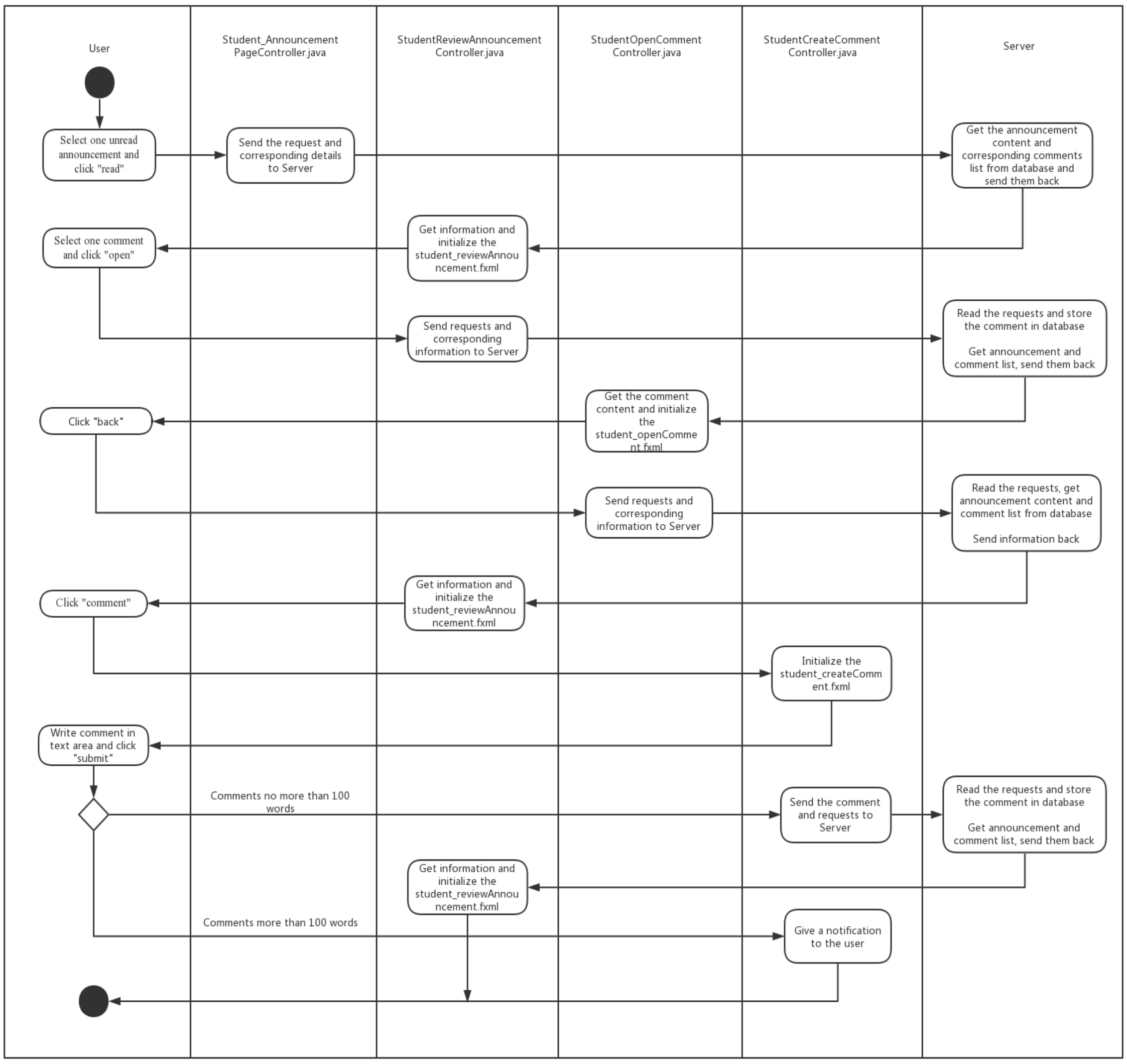

User
Student_Announcement PageController.java
StudentReviewAnnouncement Controller.java
StudentOpenComment Controller.java
StudentCreateComment Controller.java
Server
Select one unread announcement and click "read"
Send the request and corresponding details to Server
Get the announcement content and corresponding comments list from database and send them back
Get information and initialize the student_reviewAnnouncement.fxml
Select one comment and click "open"
Send requests and corresponding information to Server
Read the requests and store the comment in database
Get announcement and comment list, send them back
Get the comment content and initialize the student_openComment.fxml
Click "back"
Send requests and corresponding information to Server
Read the requests, get announcement content and comment list from database
Send information back
Get information and initialize the student_reviewAnnouncement.fxml
Click "comment"
Initialize the student_createComment.fxml
Write comment in text area and click "submit"
Comments no more than 100 words
Send the comment and requests to Server
Read the requests and store the comment in database
Get announcement and comment list, send them back
Get information and initialize the student_reviewAnnouncement.fxml
Comments more than 100 words
Give a notification to the user

Figure 6 Activity Diagram for student announcement

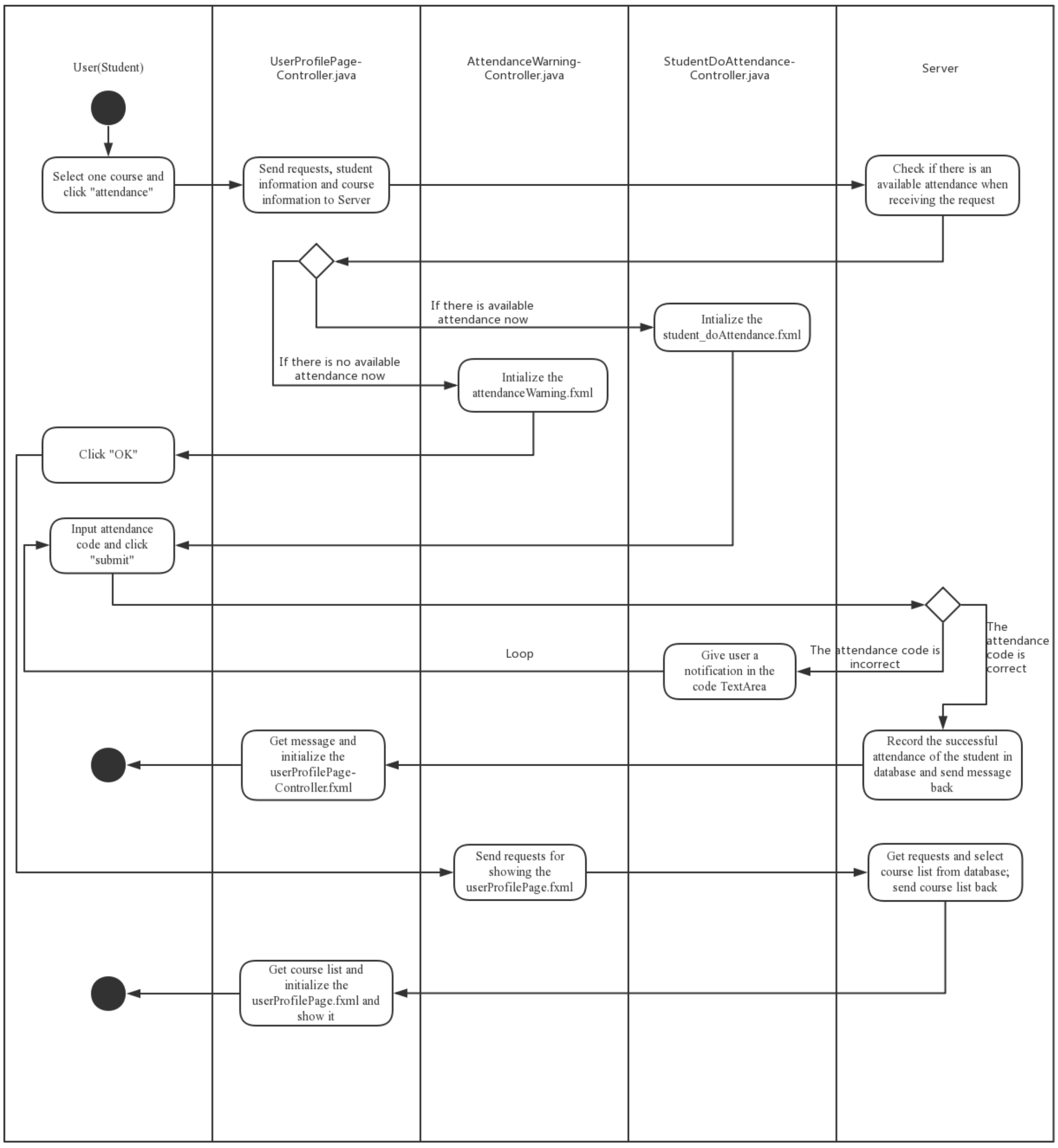

User(Student)
UserProfilePage-Controller.java
AttendanceWarning-Controller.java
StudentDoAttendance-Controller.java
Server
Select one course and click "attendance"
Send requests, student information and course information to Server
Check if there is an available attendance when receiving the request
If there is available attendance now
Intialize the student_doAttendance.fxml
If there is no available attendance now
Intialize the attendanceWarning.fxml
Click "OK"
Input attendance code and click "submit"
Loop
Give user a notification in the code TextArea
The attendance code is incorrect
The attendance code is correct
Get message and initialize the userProfilePage-Controller.fxml
Record the successful attendance of the student in database and send message back
Send requests for showing the userProfilePage.fxml
Get requests and select course list from database; send course list back
Get course list and initialize the userProfilePage.fxml and show it

Figure 7 Activity Diagram for student attendance

3. Documented use cases

3. 1 Documented use case 1

One use case has been selected according to scenario 1. This is the “attendance function” for teacher accounts. The detailed information is listed in the following table, showing the relevant actors, preconditions, flow of events and post-conditions.

Table 2 Use case 1

Use case 1: Attendance Function_teacher	
Actors	User, Database

Preconditions	1.The user has already logged in the system by a teacher's username. 2.The user has selects one course in the home pages. 3.The user has already selected attendance option in the course pages.
Flow of events	1.The user clicks "add". 2.The system requests the user to fill in the date, time and code of the new attendance. 3.The user clicks "submit". *a. If the format of date, time or code is not correct, the system will show error notification on the page.* *b. If the format of date, time or code is correct, the system will update it* in the attendance lists. 4.The system shows the previous attendance list page. 5.The user selects one attendance and clicks "delete" button. 6.The system deletes the selected attendance.
Post-conditions	1.The user has been logged out successfully. 2.All the changes must be stored and updated in the system.

3.2 Documented use case 2

One use case has been selected according to scenario 2. This is the "attendance function" for student accounts. The detailed information is listed in the following table, showing the relevant actors, preconditions, flow of events and post-conditions.

Table 3 Use case 2

Use case 2: Attendance Function_student	
Actors	User, Database
Preconditions	1.The user has already logged in the system by a student's username.
Flow of events	1.The user selects one course in home page. 2.The user selects attendance option in the course page. 3.If the current time is not in any periods of time of the attendance, the system will give a notification to state that there is no attendance at the moment. 4.The system requests the user to fill in the code of the attendance. 5.If the response is "fail authorization", the system will clear the input. 6.The system goes back to the previous page.
Post-conditions	1.The user has been logged out successfully. 2.The information must be stored and in the system database.

3.3 Documented use case 3

One use case has been selected according to scenario 3. This is the "announcement function" for student accounts. The detailed information is listed in the following table, showing the relevant actors, preconditions, flow of events and post-conditions.

Table 4 Use case 3

Use case 3: Announcement Function_student	
Actors	User, Database
Preconditions	1.The user has already logged in the system by a student's username. 2.The user has selects one course in the home pages. 3.The user has already entered the announcement list interface of certain course.
Flow of events	1.The user selects one unread announcement and clicks "read" button. 2.The user selects one comment and clicks "open" button. 3. The user clicks 'back" button. 4.The system goes back to the previous announcement page. 5.The student clicks "comment" button. 6.The system requests the user to fill in the content of the new comment. 7.The user clicks "submit". a. If the words submitted is more than the words of limitation, the system will show a notification. b. If the words submitted is no more than the words of limitation, the system will update it in comment lists. 8.The system goes back to the previous announcement page.
Post-conditions	1.The user has been logged out successfully. 2.All the changes must be stored and updated in the system.

3.4 Documented use case 4

One use case has been selected according to scenario 4. This is the "announcement function" for teacher accounts. The detailed information is listed in the following table, showing the relevant actors, preconditions, flow of events and post-conditions.

Table 5 Use case 4

Use case 4: Announcement Function_teacher	
Actors	User, Database

Preconditions	1.The user has already logged in the system by a teacher's username. 2.The user has selects one course in the home pages.
Flow of events	1. The user has already selected announcement option of certain course. 2. The user selects one announcement and clicks "open". 3.The user selects one student's comment and clicks "open". 4.The user selects the option of going back to the previous announcement page. 5.The user selects the option of going back to the previous announcement list page. 6.The user clicks "add". 7.The system requests the user to fill in the content of the new announcement. 8.The user clicks "submit". a. If the words the user submitted is more than the words of limitation, the system will show a notification. b. If the words the user submitted is no more than the words of limitation, the system will update it in announcement lists. 9.The system goes back to the previous announcement page.
Post-conditions	1.The user has been logged out successfully. 2.All the changes must be stored and updated in the system.

3.5 Documented use case 5

Table 6 Use case 5

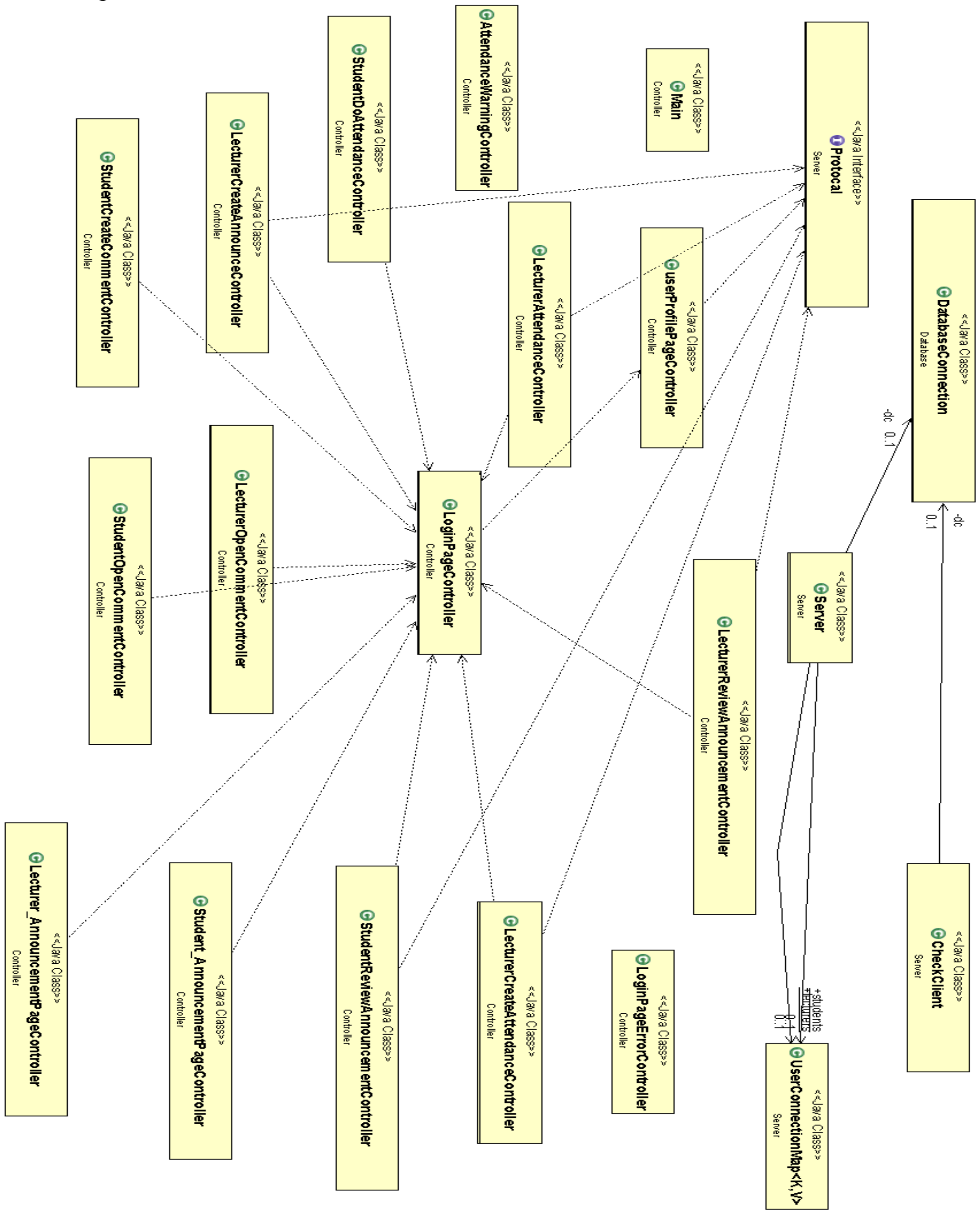
Use case 5: Multiple users login	
Actors	User1,User2 Database
Preconditions	1.Both users have an account with details already stored in the database 2. User1 will login first and then User2 will try to login whilst User1 is still logged in.
Flow of events	1.User1 open application and enters details and clicks "login". 2.User2 opens application and enters details and clicks "login".
Post-conditions	1.Both users are logged in successfully and are viewing their home page

3.6 Documented use case 6

Table 7 Use case 6

Use case 6: Announcement Function_lecturer	
Actors	User, Database
Preconditions	1.The student user has login details stored in the database
Flow of events	2. The User opens the application and enters his details and clicks "login". 3. The user opens the application again, perhaps on a different device. 4. The user enters their details and clicks login. 5. The application fails to login the user and resets the login page.
Post-conditions	1.The User remains logged in but cannot login anywhere else.

4. Class diagram

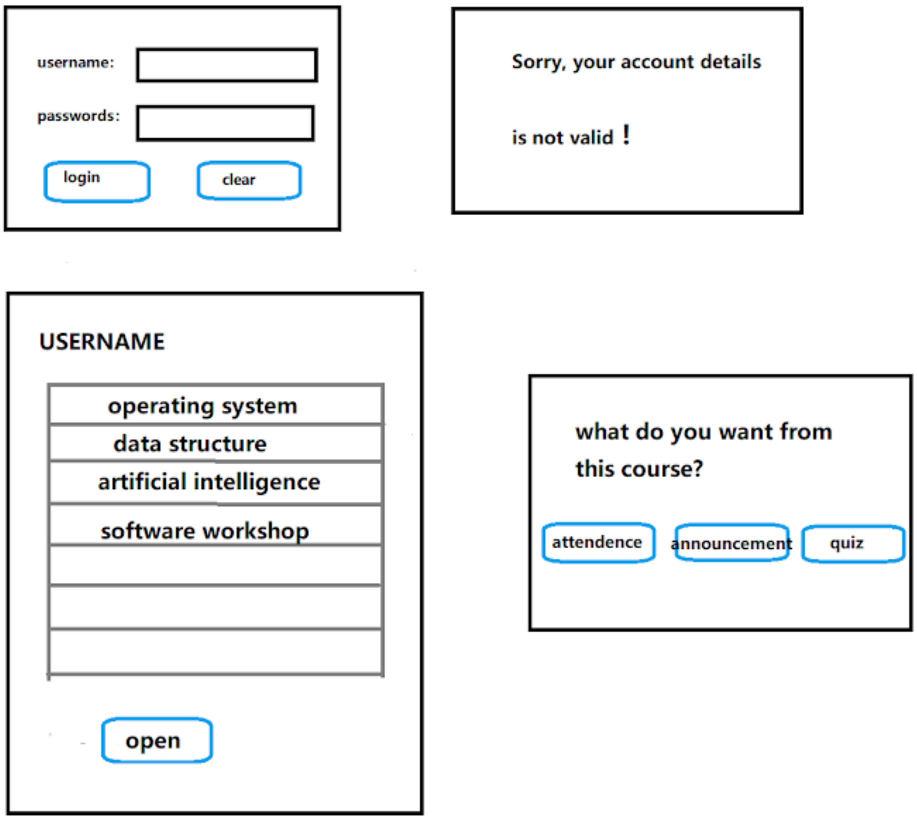


<<Java Class>> StudentCreateCommentController Controller
<<Java Class>> LecturerCreateAnnounceController Controller
<<Java Class>> StudentDoAttendanceController Controller
<<Java Class>> AttendanceWarningController Controller
<<Java Class>> Main Controller
<<Java Interface>> Protocal Server
<<Java Class>> DatabaseConnection Database
<<Java Class>> LecturerAttendanceController Controller
<<Java Class>> userProfilePageController Controller
<<Java Class>> LoginPageController Controller
<<Java Class>> StudentOpenCommentController Controller
<<Java Class>> LecturerOpenCommentController Controller
<<Java Class>> LecturerReviewAnnouncementController Controller
<<Java Class>> Server Server
<<Java Class>> CheckClient Server
<<Java Class>> Lecturer_AnnouncementPageController Controller
<<Java Class>> Student_AnnouncementPageController Controller
<<Java Class>> StudentReviewAnnouncementController Controller
<<Java Class>> LecturerCreateAttendanceController Controller
<<Java Class>> LoginPageErrorController Controller
<<Java Class>> UserConnectionMap<K,V> Server
-dc 0..1
-dc 0..1
+students
#lecturers
0..1

Figure 8 Class diagram

5. Project GUI

5.1 Conceptual GUI design



Sorry, your account details is not valid !

what do you want from this course?

Figure 9 Common used GUI

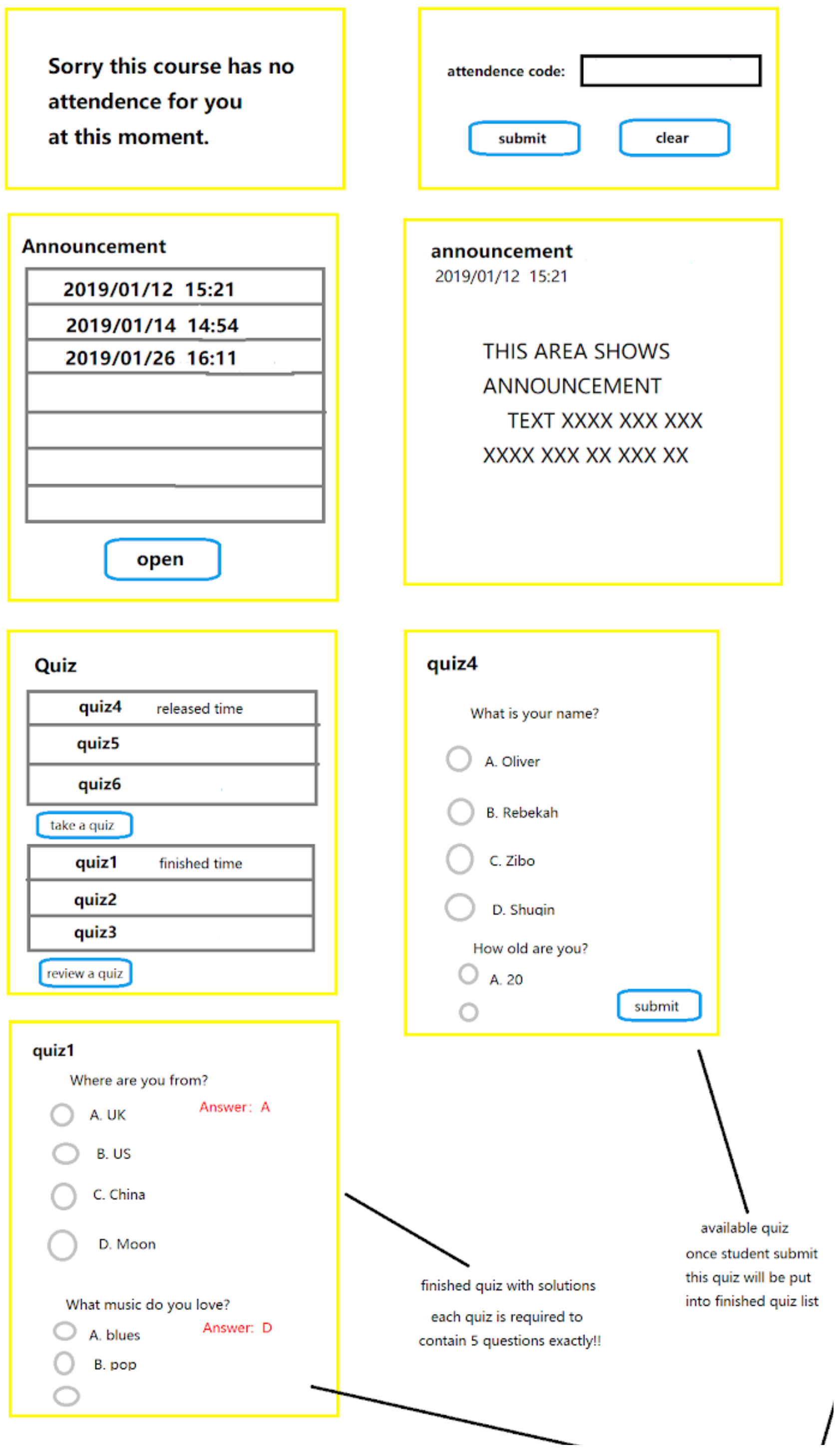


Figure 10 Student used GUI

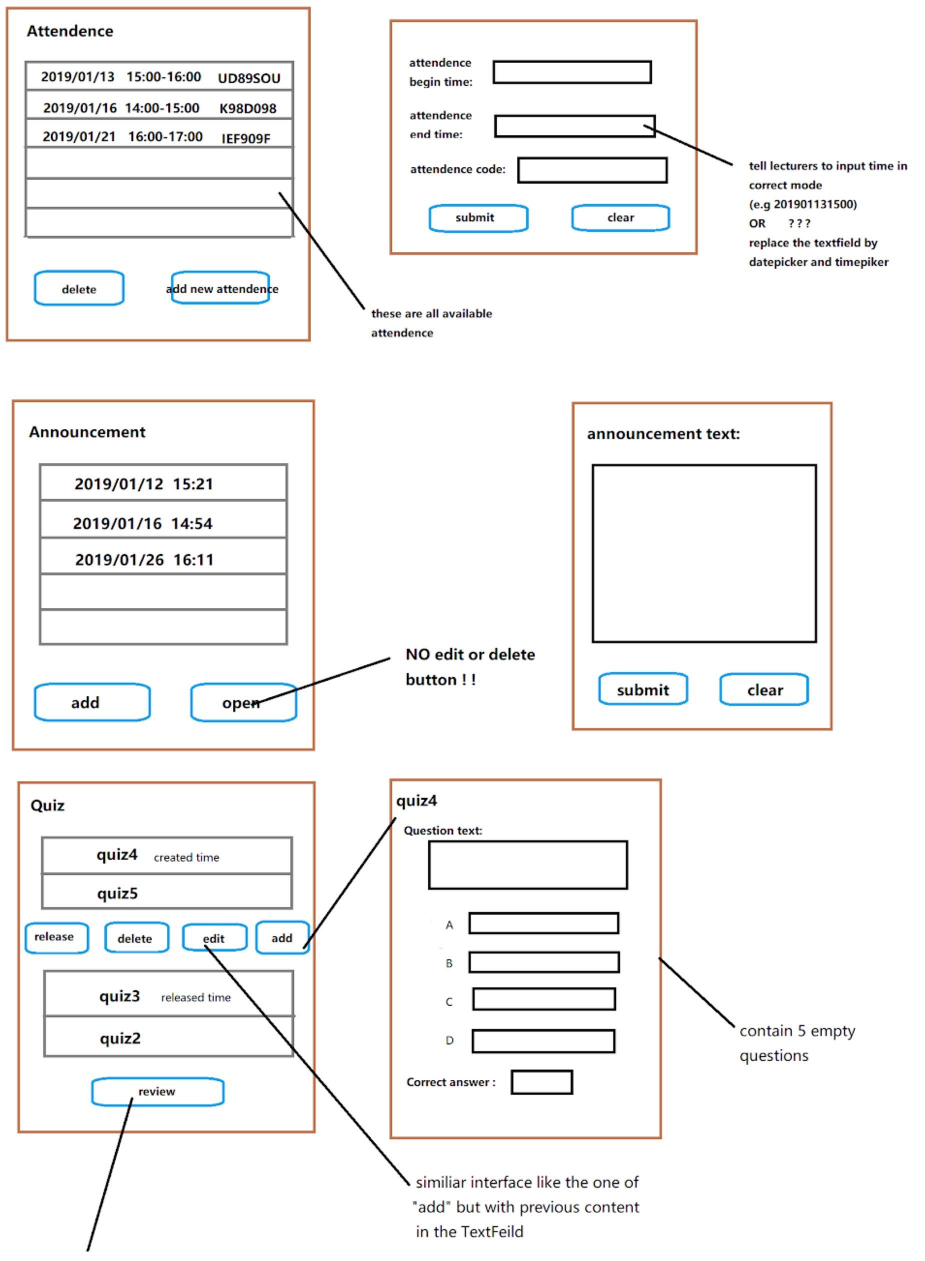
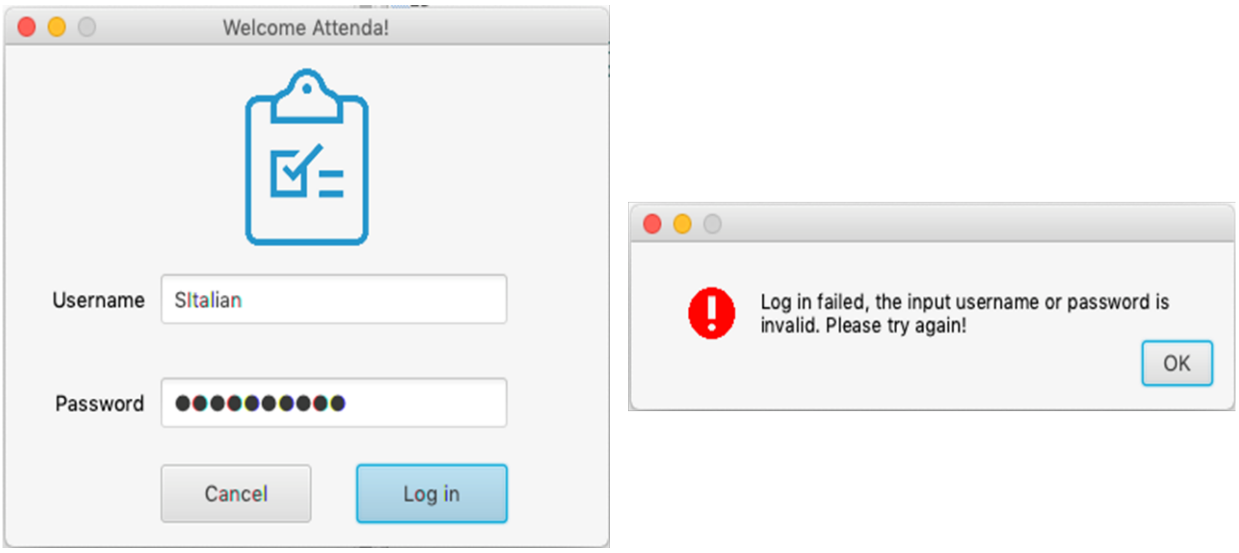

Attendence
2019/01/13 15:00-16:00 UD89SOU
2019/01/16 14:00-15:00 K98D098
2019/01/21 16:00-17:00 IEF909F
delete
add new attendence
these are all available attendence
attendence begin time:
attendence end time:
attendence code:
submit
clear
tell lecturers to input time in correct mode (e.g 201901131500) OR ? ? ? replace the textfield by datepicker and timepiker
Announcement
2019/01/12 15:21
2019/01/16 14:54
2019/01/26 16:11
add
open
NO edit or delete button ! !
announcement text:
submit
clear
Quiz
quiz4 created time
quiz5
release
delete
edit
add
quiz3 released time
quiz2
review
quiz4
Question text:
A
B
C
D
Correct answer :
contain 5 empty questions
similiar interface like the one of "add" but with previous content in the TextFeild

Figure 11 Lecturer used GUI

5.2 Final GUI design



Log in failed, the input username or password is invalid. Please try again!

Figure 12 Login GUI

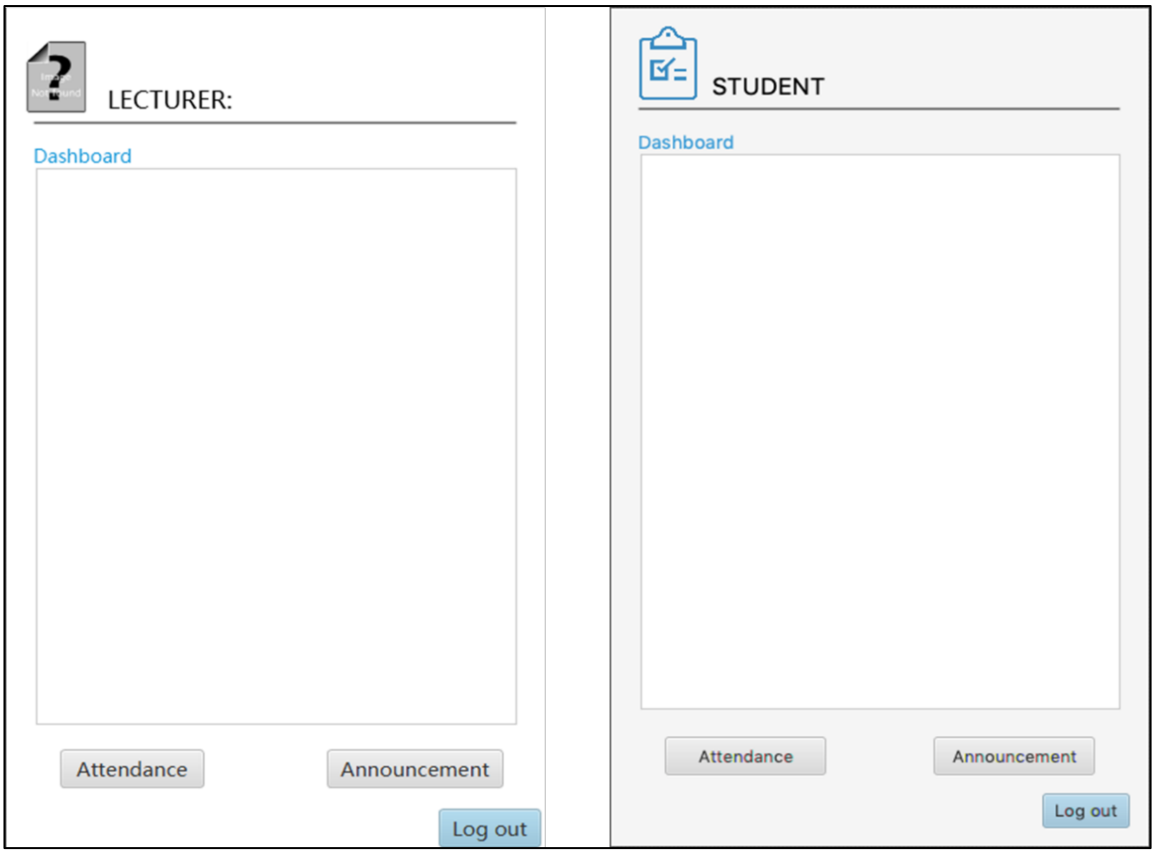


Figure 13 Lecturer_StudentUserProfile GUI

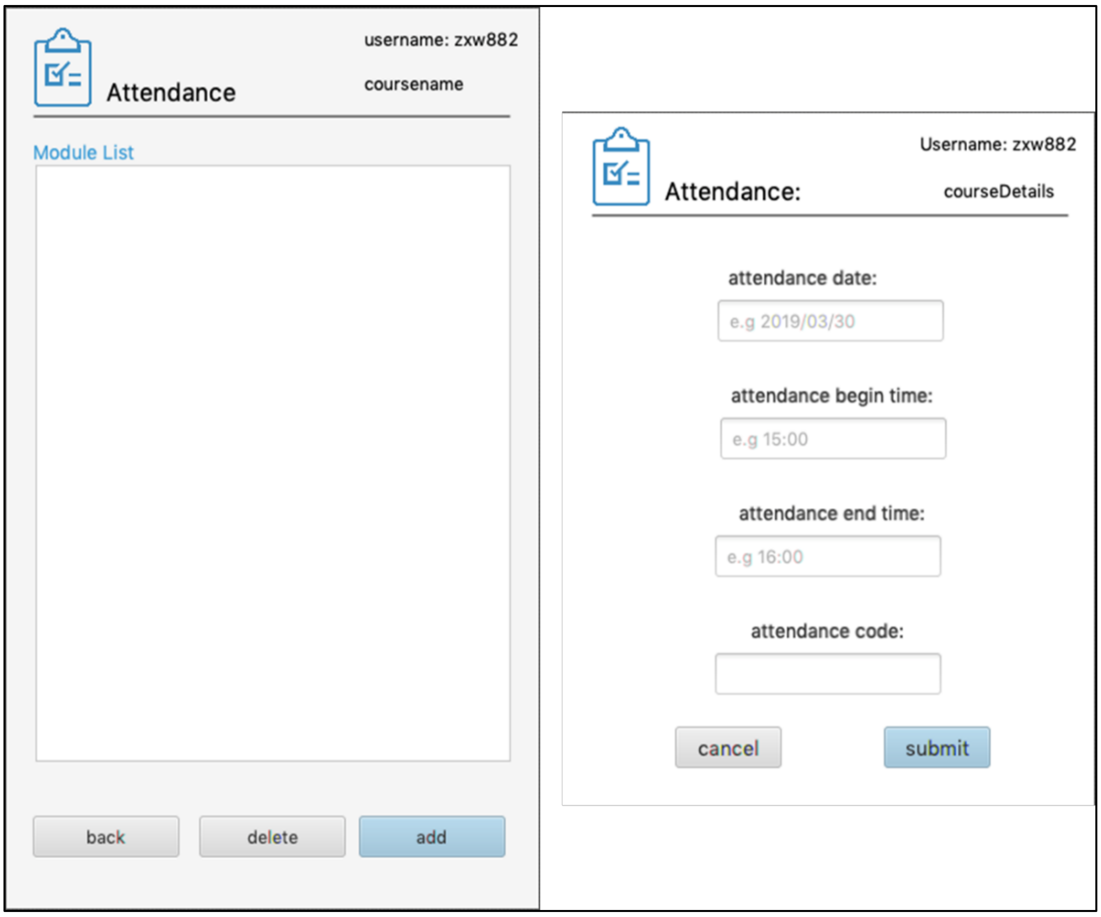
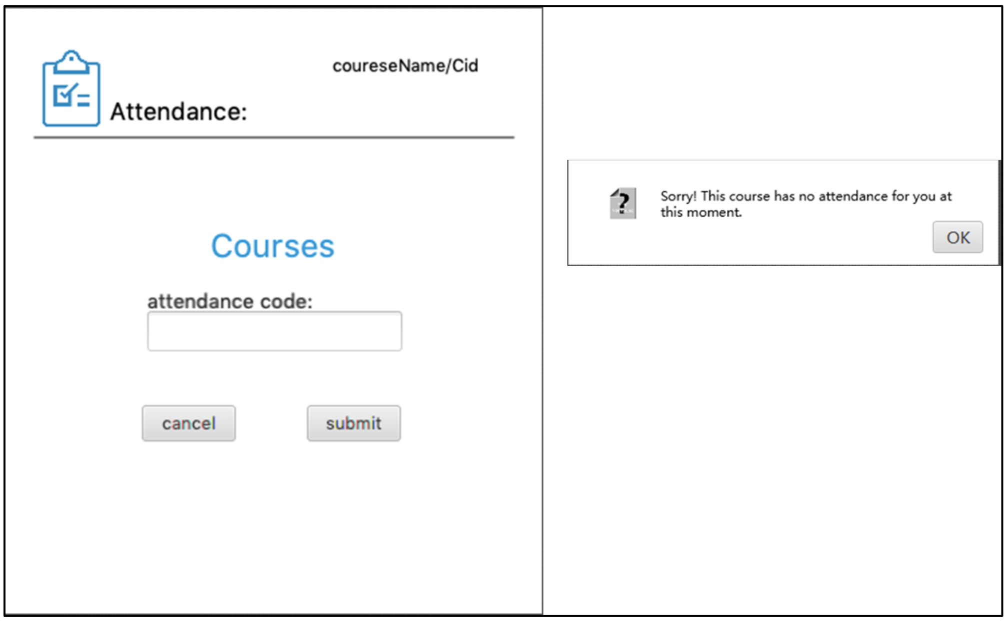


Figure 14 LecturerCreate_DeleteAttendanceGUI



Sorry! This course has no attendance for you at this moment.

Figure 15 StudentDoAttendanceGUI

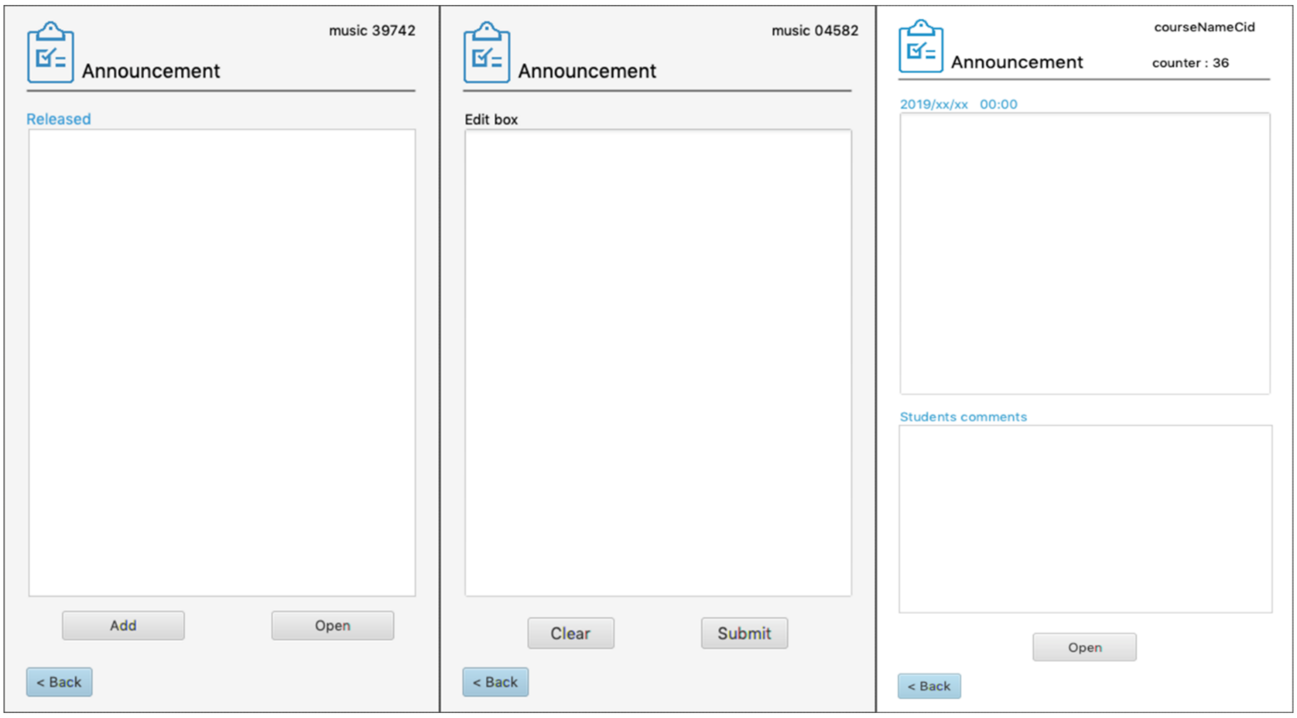


Figure 16 LecturerSelect_Add_ReadAnnouncementGUI

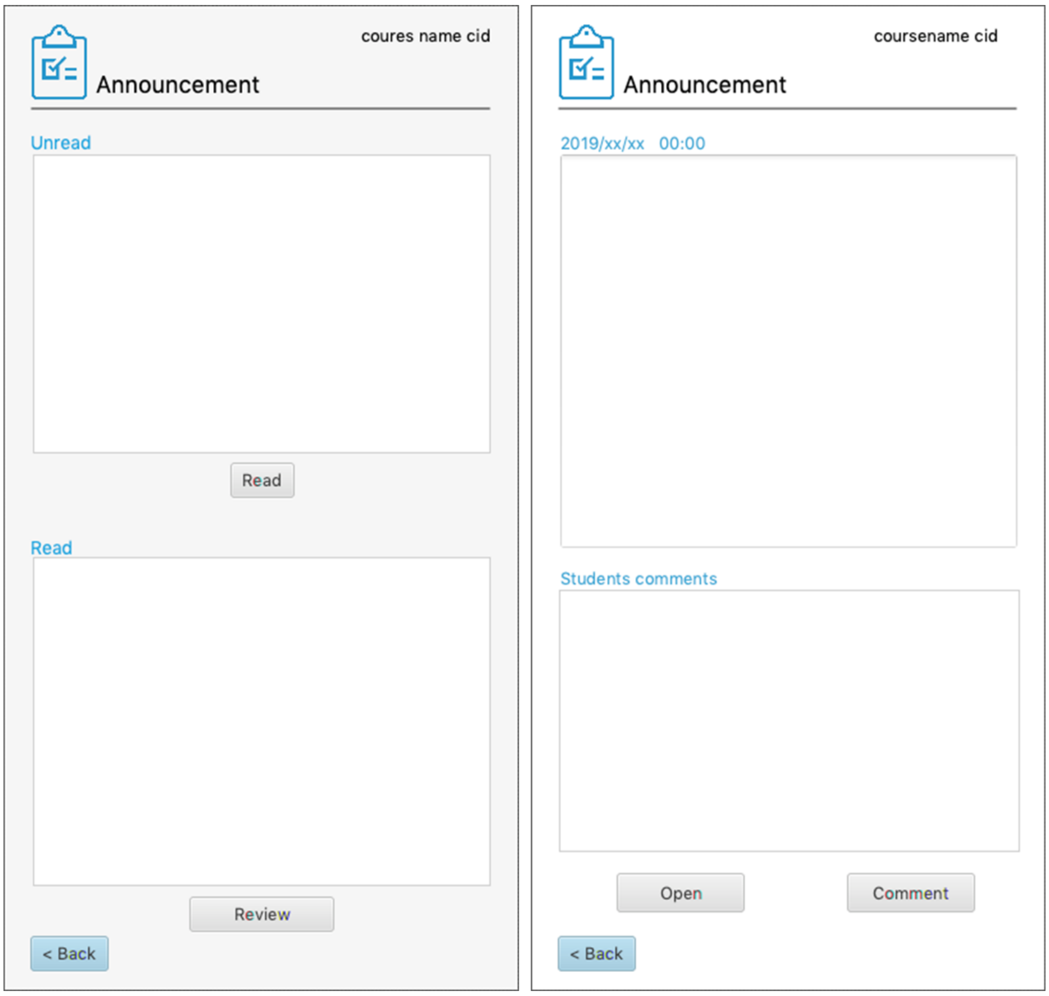
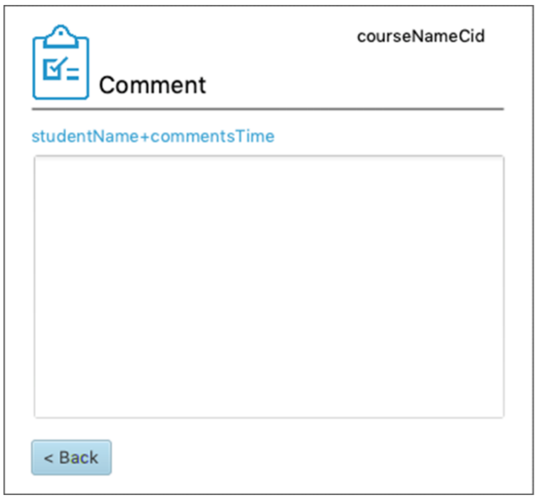


Figure 17 StudentSelect_ReadAnnouncementGUI



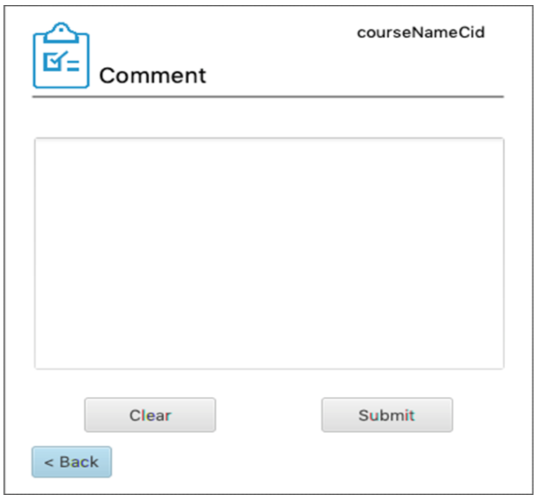


Figure 18 StudentRead_CreateCommentGUI

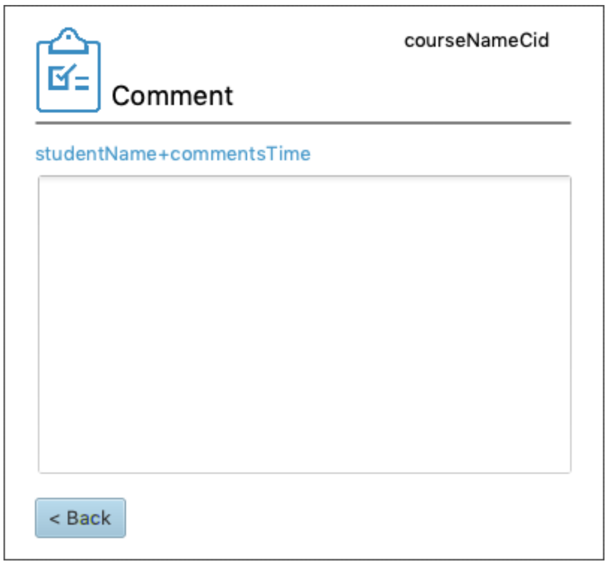


Figure 19 LecturerReadCommentGUI

6. Weekly agenda sheets

Table 8 Weekly agenda table

Agenda week	Task allocation
Week 7	**Oliver:** - Design the tables for the database and write the SQL statements to create them - Start writing the java program for connecting to the database - E-R modelling **Junlin, Zibo:** - Update the specification where necessary - Design the interfaces for all proposed functionalities of the system - Work on client server code. Assume access to a database

		Shuqin, Rebekah: - Create the login GUI
Week 8		**Oliver:** - Continuing on the database creation **Zibo:** - User profile page design - Announcement page and quiz page design **Rebekah:** - Login window and sign up window (Rebekah) **Betty:** - Attendance page design **Junlin:** - The GUI interface sketch design - Continuing on the client and server design
Week 9	3.13 - 3.15	**Oliver:** - Do E-R analysis and produce a diagram; - Finish methods in Server to select courses' names' for course list interface of student and lecturers, **Junlin:** - Finish methods in Controller to build course name list interface of student and lecturers **Rebekah:** - Finish report draft; - Finish methods in Controller to build announcements list interface of student and lecturers **Zibo:** - Finish report draft - Finish methods in Server to select attendance list for attendance list interface of lecturers' and to decide whether a student can has attendance **Shuqin:** - Finish button methods in Controller for attendance of lecturers and students
	3.16 - 3.17	**Oliver:** - Finish methods in Server to select quizzes details for quizzes list interface of students and lecturers **Junlin:** - Finish methods in Controller to build attendance list interface of lecturers and to decide whether a student can has attendance **Rebekah:** - Finish methods in Controller to build quiz list interface of students and lecturers **Zibo:** - Finish button methods in Controller for quiz function of lecturers **Shuqin:** - Finish button methods in Controller for announcement of lecturers and students

	3.18	**Oliver:** - Finish methods in Server to select announcements titles and time for announcements list interface of student and lecturers **Zibo:** - Finish button methods in Controller for quiz function of students
Week 10		**Oliver, Rebekah:** - Get the attendance GUI fully working, include a back button and check button functionality **Junlin, Zibo, Shuqin:** - Get the attendance GUI fully working, include a back button and check button functionality

7. Meeting minutes

19/03/19(3:00-4:00)

- Oliver, Junlin, Zibo, Shuqin, Bekah
- Discussed about how the structure and content of the report will be.
- Following the suggestion of the tutor, we decide to design more interaction in the project. For example, students can reply to the announcement, and can see how many students read it and their comments.
- Discussed how to write the report and complete the project if we can’t finish the quiz part.
- Discussed details about how to change the announcement part,which is about how to add the feedback part.
- Discussed one change in the attendance part about ”cancel” button.

21/02/19 (2:30 – 3:30)

- Oliver, Junlin, Zibo, Shuqin, Bekah
- Discussed ideas of project and landed on system that allows students to submit attendance, answer quizzes, and see announcements along with some other functionalities
- Created word document with project description/specification

26/02/2019(3:00 - 4:00)

- Oliver, Junlin, Zibo, Shuqin
- Discussed GUI interfaces for parts of the system
- Discussed how the system would be implemented, e.g would a person be able to be logged in on two machines at the same time
- Discussed tasks to be done for this week and assigned these tasks to members of the group

01/03/2019(3:00 - 4:00)

- Oliver, Junlin, Zibo, Shuqin, Bekah
- Discussed progress with our tasks and got group opinions on decisions needing to be made regarding database tables and interface designs
- Discussed aim for over the weekend

- Room 117 for tutor

05/03/2019(2:00 - 4:00)

- Oliver, Junlin, Zibo, Shuqin, Bekah
- Discussed progress with our tasks and got some advice from the tutor
- Discussed aim for over the weekend
- Room 117 for tutor

12/03/2019 (3:00 - 4:00)

- We need to refine our tables in database such as highlighting the Foreign key and Key in tables.
- We need to do E-R analysis for database to make sure the correctness.
- We should keep progressing our project when there is some problem. Just skip those problem and do hard code for the rest part.
- Try our best to have these coding finished before next tutorial and do a report draft on this Saturday.

19/03/19(3:00-4:00)

- Discussed how the structure and content of the report will be.
- Following the suggestion of tutor Rajiv, we decided to design more interaction in the project. For example, students can reply to the announcement and can see how many students read it and their comments.
- Discussed how to write the report and complete the project if we can't finish the quiz part.
- Discussed details about how to change the announcement part, which is about how to add the comments part.
- Discussed one change in the attendance part about "cancel" button.

21/03/2019 (10:00 - 12:00)

- All present
- Meeting to finish off the project report and work on code

8. Group Contributions

8.1 Overall percentage contribution of each member:

Table 9 Member contribution table

Member	Oliver	Junlin	Zibo	Rebekah	Shuqin
Contribution	20%	20%	20%	20%	20%

8.2 Summary of Primary Contributions:

Oliver - Designed and filled Database, produced ER diagram and database connection functions for the server to use in the application. Worked to complete the attendance functionality with the client-server-database system. Contributed to the report on areas specific to database and contributed to decisions and writing of functional requirements, testing and other areas of the report.

Junlin - Produced the bulk of the server-client system for our application and joined it with our GUI and database code. Did activity diagram. Contributed to the report on areas specific to the server-client system and contributed to decisions and writing of functional requirements, testing and other areas of the report.

Zibo - Initial design of GUI and produced the GUI and corresponding codes for the client to use in the login and user profile part of the application. Contributed to final GUI changes needed towards the end of the project. Did Class Diagram. Contributed to the report on the GUI section and contributed to decisions and writing of functional requirement,testing and other areas of the report.

Rebekah - Produced GUI for announcement and the corresponding code for changing pages. Contributed to final GUI changes needed towards the end of the project. Contributed to the report on the GUI section and contributed to decisions and writing of functional requirements, testing and other areas of the report.

Shuqin - Produced GUI for attendance and the corresponding code for changing pages. Contributed to final GUI changes needed towards the end of the project. Did Use Case Diagram. Contributed to the report on the GUI section and contributed to decisions and writing of functional requirements, testing and other areas of the report.

Everyone - Towards the end of the project all members contributed in completing the code and the report and contributed to a variety of areas.