School of Computer Science and Informatics

Coursework Submission Cover Sheet

Please use Adobe Reader to complete this form. Other applications may cause incompatibility issues.

Student Number	1865452
Module Code	CMT303
Submission Date	01/07/2019
Hours spent on this exercise	50
Special Provision	

(Please place an x in the box above if you have provided appropriate evidence of need to the Disability & Dyslexia Service and have requested this adjustment).

Group Submission

For group submissions, *each member of the group must submit a copy of the coversheet.* Please include the student number of the group member tasked with submitting the assignment.

Student number of submitting group member	1865452 (Team 4)

By submitting this cover sheet you are confirming that the submission has been checked, and that the submitted files are final and complete.

Declaration

By submitting this cover sheet you are accepting the terms of the following declaration.

I hereby declare that the attached submission (or my contribution to it in the case of group submissions) is all my own work, that it has not previously been submitted for assessment and that I have not knowingly allowed it to be copied by another student. I understand that deceiving or attempting to deceive examiners by passing off the work of another writer, as one's own is plagiarism. I also understand that plagiarising another's work or knowingly allowing another student to plagiarise from my work is against the University regulations and that doing so will result in loss of marks and possible disciplinary proceedings.

Educational System

Team 4

CMT303

Group Members:

Armando Castany

Hannah Rosser

Qifa Cai

Spencer Du

Wang Chaoran

Zak Kyriakou

Report Word Count: 5731

Table of Contents

Summary of Members

Individual Member 1: < Armando Castany >

Individual Member 2: < Hannah Rosser>

Individual Member 3: < Qifa Cai>

Individual Member 4: < Spencer Du >

Individual Member 5: <Chaoran Wang>

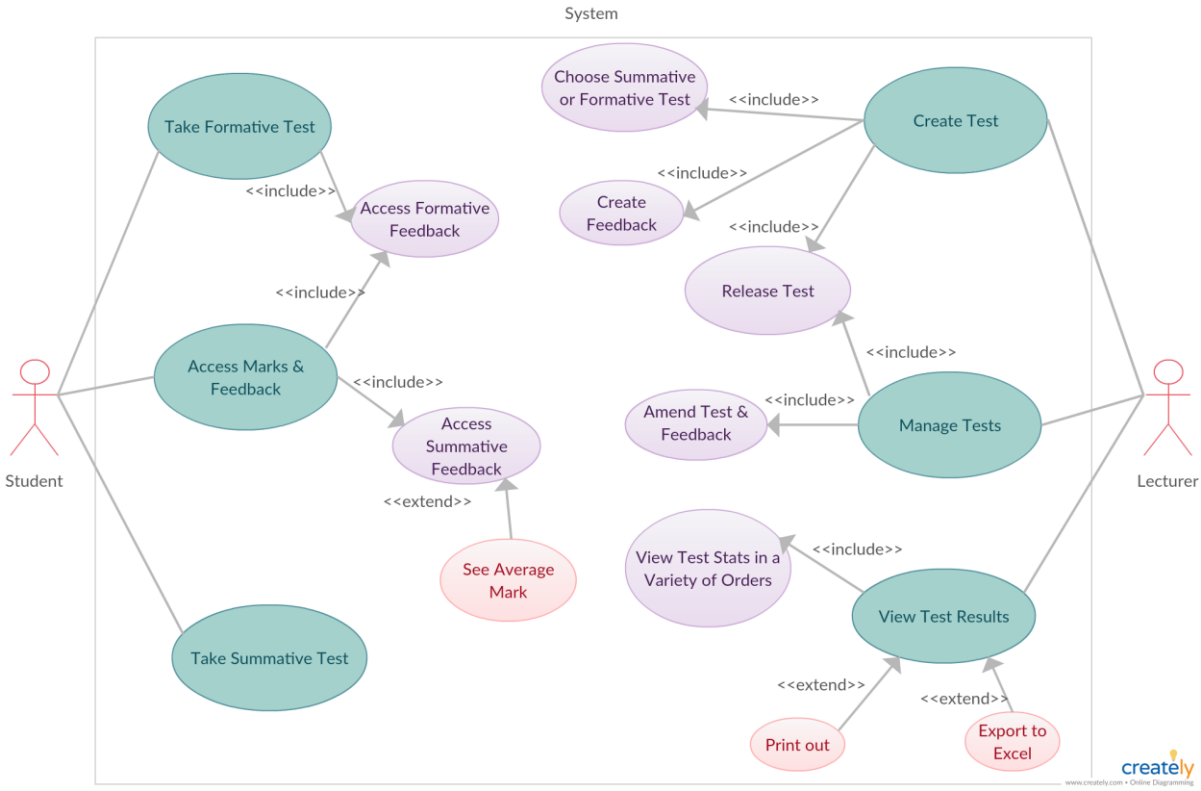
Individual Member 6: < Zak Kyriakou >

Introduction

With cohort sizes growing, the marking workload for lecturers is increasing. Although automated making of assessment is not desirable for all assessments, the assessment of knowledge and understanding of a topic often can be automated. As a school, staff are being encouraged to consider automated marking were appropriate to provide students with more timely feedback. Staffs are also being encouraged to provide formative assessment opportunities which allow students to gain a better idea of their understanding of a module or topic area. Learning central does provide facilities for online tests but many staff does not like the learning central interface and so are turned off trying this. We need a system that provides an assessment test that can be Summative or Formative depending on the needs of the module, lecturers and Students.

Requirement document

➢ Use Case Diagram:



➢ Use Case Description:

Use Case Description of a Functional requirement

Use Case No:1	Use Case Name: Take Formative Test	Rating: Must Have
Purpose: Students should be able to choose and take tests and answers questions. (Formative tests)		
Main Actor: Student	Secondary Actors:	
Description: • Students must choose their test type before starting the test. (Formative tests) • The system should record how many times a question was answered correctly / incorrectly. • The test should have questions in random orders as to make sure the tests aren't repetitive. • The test should provide with the flexibility of having single or multiple attempts (in a single sitting) and should only provide with the answers following student's final attempt. • In each test, student can exit the test only after they have completed all the questions. Taking the tests should be intelligible and picking the answers and submitting the work should be clear and simple. System checks student's current account to ensure they can only login in their own account.		

Use Case No: 2	**Use Case Name:** Access Mark and Feedback	**Rating:** Must have
Purpose: The student can **access their feedback** for both the **formative** and **summative** tests. They should also see the overall **average score** of the class for the **summative test**		
Main Actor: Student	**Secondary Actors:**	
Description: • The student must be able to see their individual mark and feedback for both the formative and the summative tests They must not be able to see any other individual's mark (this can be implemented by using separate log-ins for each user) • For the **formative** mark: the student must be able to access this after completing the formative test • For the **summative** mark: the student must only be able to access this after the deadline for completing the summative test has passed • The student should also be able to see the **average mark** of the class (for the whole test) for the **summative** test • They could also see the **average percentage** of students who got each question correct The system must be programmed to save all the results and calculate the average percentage of correct answers • They could also be able to see their own position in the class, but nobody else's.		
Author: Hannah Rosser **Date:** 4/1/2018	Approved:	Date:

Use Case No: 3	Use Case Name: Take summative tests	Rating: Must Have
Purpose: Students should be able to choose and take tests and answers questions.		
Main Actor: Students	Secondary Actors:	
Description: • Students MUST be able to choose and take summative tests from within a module. • They MUST also be able to answer the questions in the summative tests. • The student MUST be able to submit their answers at the end of the test. • The test should only be taken once and once submitted it cannot be accessed again. • The test can only be taken before the deadline. Taking the tests, picking the answers and submitting the work should be clear and simple.		

Use Case No: 4	Use Case Name: Create test	Rating: Must have
Purpose: The lecturer must be able to create a test and choose whether it is formative or summative.		
Main Actor: Lecturer	Secondary Actors:	
Description: • Lecturer must be able to create a set of questions for a test, answers, marks and feedbacks, but students can't see the answers and feedbacks before the appropriate time. • Lecturer must be able to choose the type of test it is. • Test must be released to the students on the relevant module.		

Use Case No: 5	Use Case Name: Manage Tests	Rating: Must have
Purpose: The lecturer must be able to manage a formative or summative test		
Main Actor: Lecturer	Secondary Actors:	
Description: • Lecturer must be able to modify the Module ID, name, questions, answers, marks, feedback and deadline of the test. • Test must be released to the students on the relevant module.		

Use Case No: 6	Use Case Name: View test results	Rating: Should have
Purpose: Lecturer should be able to see the stats of the test results and to have them arranged in a variety of orders and ranges (module, semester, year…). It also should allow printing the test results and statistics on paper or exporting them to Microsoft Excel		
Main Actor: Lecturer	Secondary Actors:	
Description: • Lecturer should have an "See Test stats" option in the application; • The stats should be accurate; • Stats should be typified by Pass and Fail, but also provide information about lower grades, higher grades and to have them arranged by Honours in undergraduate and Merits or Distinctions in post graduate. • Information could be presented using graphics (pie charts, bars, and so forth); • Lecturer should be able to see previous test stats. • Lecturer should be able to arrange the graphs by order and select different ranges. • Lecturer should be able to print or export to Microsoft Excel the results or graphs;		

Risk Management

Risk	Likelihood	Impact	Strategy
Time required to develop a module is underrated	High	Moderate	Prioritise more important modules (prioritising developing the 'must have' use case descriptions first then the 'should have' then 'could have') and allow more time for larger modules.
Using different versions of programming languages	Low	Large	Make sure all software everyone is using is updated to the latest version, so we are all using the same version.
Required knowledge/skills for implementation is not yet known	High	Large	Study and research a lot, to gain the knowledge needed. Speak to other team members for assistance as some members are more experienced with coding than others.
With the team members initially allocated, the skills may vary a lot	High	Moderate	Allocate tasks based on skill level and allow time for lower skilled members to learn and develop their modules.
Team member ill at critical time	Low	Large	Plan ahead so that work isn't left until the last minute. Extenuating circumstances in extreme cases.
Data loss / system failure	Low	Large	Backup regularly (daily) onto multiple platforms (GitLab/External hard drives.)
The time of the project is so limited and none of the members can be fully occupied by the project that the schedules may often get delayed for all kinds of reasons	High	Large	Start early and make sure we have enough time to make changes. If one person runs into trouble with their part, they must voice their problem so that other teammates can help.
During development team members may waste time developing unnecessary features which waste development time	Low	Moderate	Develop the basic and required code first and only add extra features if there is enough time. We can also use a Scrum framework to make sure we're making enough progress as a team and to review and make changes based on other team members feedback.
Misunderstanding each other due to language barriers	Medium	Moderate	Team members should always ask and discuss things which they may not have understood. And all members should try and be clear with their ideas.
Serious design error is discovered during implementation of other modules	Low	Large	Focus on every step to ensure correct implementation. Communicate to team to ensure that everything that is required has been developed correctly.
User interface proves unattractive to the client	Medium	Moderate	Capture client's preferences in user interface in the early stage.
Personal data leaks	Medium	Large	Using encryptions and passwords to hide sensitive personal information

Leaving system open/ logged in	Medium	Large	Include a timeout (10 minutes) on the system for inactivity.
Software is incompatible with accessibility systems	Low	Large	Ensure accessibility validation and check it is compatible with accessibility software.
Software crashing	Medium	Large	Implement error detection and recovery and test code fully and repetitively to ensure stability.
Plagiarism	Low	Large	Use referencing and do not copy code.
Rounding errors	Medium	Small	Ensure accuracy of the operations and rounding of floats (if any) is correctly implemented.

		Likelihood of occurrence		
		High	Medium	Low
Potential scale of impact	Large	(2)	(3)	(6)
	Moderate	(2)	(2)	(1)
	Small		(1)	

Design Model

➢ Detailed Specification:

Detailed Use Case Description of a Functional requirement

Use Case No:1	Use Case Name: Take Formative Test	Rating: Must Have
Purpose: Students should be able to choose and take tests and answers questions. (Formative tests)		
Main Actor: Students	Secondary Actors:	
Pre-conditions: Before conducting the test, the lecturer should ensure that all test content and the release has been completed. Students should complete the authentication before entering the test and enter the system to wait for the test with the correct module.		
Trigger; Students click 'Formative Test' button from within the module they need to take the test.		
Description: step by step • Students must be able to click on the formative test from within the module which has a test available. (Student clicks on certain test and the app loads the test). The Formative test that students take must come from his lecturer creation. • If the student wants to start the test, they must click the start button. • Because this test is a formative test, every time a student tries, he should only be provided with the answers following his final attempt. • After completing the last question, the student can exit the test by clicking the Exit button. • This test can be tested multiple times. • Students can click on the feedback button to see the feedback from the tutor (After final attempt). • System checks student's current account to ensure they can only login in their own account. • Taking the tests should be simple through a clear interface and picking the answers and submitting the work should also be clear and simple. Alternative flow: • Test isn't available to be taken. The system checks to make sure the test is open to the lecturer before each test. If the test is found to be not ready, the system will prompt the student that the test is not ready. • Student chooses the wrong answer. A confirmation button appears each time the student chooses an answer. And a window appears showing the answers the student has chosen. Through this kind of operation, I can guarantee that every answer of the students is the one they want to choose. • Students do not complete the test. Before the test is finished, the system will detect if the student has answered all the questions. If the student is found to have not answered all the questions, the system will reject the student's		

request to quit the system. At the same time, the system will display a prompt box telling students that they must complete all the questions before they can exit the test.

- Student can't get the answers.

Student should be provided with the flexibility of having single or multiple attempts (in a single sitting). The student should only be provided with the answers following their final attempt. At the end of each test, the system displays a text box with the words "Is this your last attempt?" If the student chooses yes, then he will get the final answer. In other cases, they will not get the final answer. Unless it is the last attempt, the system will not give back the correct answer to the student.

Extensions: If the formative test has already been done the last time then the student should get the answers about this test.

Related Use Cases:

2. Access Marks &Feedback

4. Create Test

5. Manage Tests

6. View Test Results

Post-conditions: Students will know their grades and deficiencies. And students and lecturers will know how many times a question was answered by students correctly / incorrectly.

Author: Chaoran Wang	Date:04-01-2019	Approved:	Date:	Version:1.2

Use Case No: 2	**Use Case Name:** Access Mark and Feedback	**Rating:** Must have

Purpose:

The student can **access their feedback** for both the **formative** and **summative** tests.

They should also see the overall **average score** of the class for the **summative test**

Main Actor: Student	**Secondary Actors:**

Pre-conditions:

The student must complete and submit their formative/summative test (their results, along with the class', must be saved in a database)

The results of the summative test must be released (This must happen either automatically after the test deadline, or the lecturer must manually be able to release results using the programme.)

The student must be logged into their account on the system. (The system should store different account information where the student can log in with a username and password)

Trigger;

The clicks to view their results from the homepage

Description:

The Student should:

- Select whether they want to view their **formative** test results, or their **summative** test results
  Select Formative:
  - See their test results
  - See their feedback: which answers they got correct/incorrect and their total mark

  Select Summative:
  - Select to see their own test results & feedback
    - ❖ See their total mark and their feedback, along with which answers they got correct/incorrect
  - Select to See Average Mark
    - ❖ See the average mark of the class (for the whole test)
    - ❖ See the average percentage of students who got each question correct
      The average mark would be calculated by the system once all students have taken the test (if a student has not taken the test they are awarded a 0 but this does not contribute to the average mark)
    - ❖ See their position/rank within the class but nobody else's

Alternative Flow:

- The student selects to view the wrong test (i.e. intended: formative, clicked: summative)
  The student can select to return to the homepage and make their selection again
- The system times out due to inactivity
  If the student is inactive for too long, the system will automatically log them out in order to ensure data security

Extensions: See Average Mark

Related Use Cases:

Take Formative Test (1)

Take Summative Test (3)

Post-conditions:

The summative mark will be counted towards their final grade for the module.

Author: Hannah Rosser	**Date:** 4/1/2018	Approved:	Date:	Version

Detailed Use Case Description of a Functional requirement

Non-functional requirements for this Use Case Description:

- Security and privacy – separate logins
  - They *must not* be able to see any other individual's mark, only the average of the whole class (this can be implemented by using separate log-ins for each user)
- Usability – the system should have a clear interface and should be easy to use and navigate through
- Reliability
  - The system must calculate the average mark accurately and the system should be tested so as to eliminate risk of crashing or malfunctioning

***Use Case No**:* 3	***Use Case Name:*** Take summative tests and answer questions	***Rating:*** MUST HAVE
Purpose: Students should be able to choose and take summative tests and answer the questions.		
Main Actor: Student	***Secondary Actors:*** None	
***Pre-conditions*:** Summative tests must have been written by the lecturer for the module and the tests must have been made available for students to answer. Student is logged into the system and is in the correct module they need to take the test for. The test deadline must not have passed.		
Trigger: Students click 'Summative Test' button from within the module they need to take the test.		

Description:

Student can choose to take a summative test from within a module, ***before the specified deadline*** (see alternative flow if student tries to access a test once deadline has passed) and must be able to answer the questions and submit the work at the end of the test.

Basic flow:

- Students *must* be able to click on the summative test from within the module which has a test available. (Student clicks on certain test and the app loads the test).
- Students *must* be able to answer from pre-defined multiple-choice questions, with options for the students to pick from. (App loads multiple choice answers).
- Students *must* be able to submit their answers once they have completed a test. As summative tests can only be taken once, this submit button will have a confirmation screen to allow the student to be sure they are ready to submit their final answers. (click on 'submit' button where the app saves the data the student has submitted).
  Taking the tests should be straightforward through a clear interface and picking the answers and submitting the work should also be straightforward.

Alternative flow:

1. **Test isn't available to be taken**

Student tries to take a summative test from within a module. If the summative test has not been created and/or has not been made available to be taken yet, then the system should reply with a message that states, 'This test is not yet available, please contact your lecturer if the test is supposed to be available at this current time.' Ending the user case.

2. **Deadline for test has passed**

If the student tries to access a summative test once the deadline has passed then the system should reply with a message stating, 'This test can no longer be taken as the deadline has passed.'

3. **Logout timeout for inactivity**

If the student is inactive for a period of time, the system will log them out to ensure their data is secure. Ending the user case until the student logs back on.

4. **Lose internet connection**

If the student loses internet connection, the system should save where the student was in the test until the student then logs back in to resume the test. Ending the use case.

5. Summative test has already been taken

Student has already taken the test but tries to take it again. If this occurs, the system should display a message stating, 'You have already taken this test, please see the test feedback section if available.'

Extensions: If the summative test has already been taken then the student should access the feedback/results section.

Related Use Cases:

Access feedback from tests (3).

Create Summative tests (5).

See test results (6).

Post-conditions:

Receive their result (once the deadline has passed and the results have been released). Receive feedback based on the questions in the test (also once deadline has passed).

Author: Zak Kyriakou	**Date:** 12-Dec-18	**Approved:**	**Date:**	**Version: 1.1**

Use Case No: 4	Use Case Name: Create test	Rating: Must have
Purpose: The lecturer must be able to create a test and choose whether it is formative or summative.		
Main Actor: Lecturer	Secondary Actors:	
Pre-conditions: Logging into the system.		
Trigger: The lecture wants to create a test or a series of tests to test students on the module.		

Basic flow:

1. Lecturer clicks on create test.
2. Lecturer chooses the type of test.
3. If the lecturer chooses the right type of test then the lecturer proceeds to the next step.
4. Lecturer creates a test name through a clear interface.
5. If the lecturer does not log out of the system after creating a name for the test, then the lecturer proceeds to the next step.
6. Lecturer creates the test ID through a clear interface. Test ID can be modifiable.
7. If the lecturer does not log out of the system after creating the test ID, then the lecturer proceeds to the next step.
8. Lecturer adds a relevant module ID through a clear interface. Module ID can be modifiable.
9. If the lecturer does not log out of the system after adding the relevant module ID then the lecturer proceeds to the next step.
10. Lecturer creates the first question through a clear interface. Question can be modifiable.
11. If the lecturer does not log out of the system after creating the first question then the lecturer proceeds to the next step.
12. Lecturer creates the answers for the question through a clear interface. Answers can be modifiable.
13. If the lecturer does not log out of the system after creating the answers then the lecturer proceeds to the next step.
14. Lecturer adds the mark to the question through a clear interface. Mark can be modifiable.
15. If the lecturer does not log out of the system after adding the mark to the question then the lecturer proceeds to the next step.
16. If the lecturer does not want to create another question then the lecturer proceeds to the next step.
17. Lecturer creates the feedback for the test.
18. If the lecturer does not log out of the system at all when creating the feedback then the lecturer proceeds to the next step.
19. The lecturer sets up how many times the students can do the test.
20. If a deadline does not need to be added then lecturer proceeds to the final step.
21. Lecturer releases the test to the students on the module and must make sure that they don't also release the answers and feedbacks at the same time. The test must be released securely.

Alternative flow:

3.a. If the wrong test type is selected then the lecturer goes back to the previous page and the use case continues.
5.a. If the lecturer logs out of the system after creating a name for the test then use case finishes.
7.a. If the lecturer logs out of the system after creating a ID for the test then the use case finishes.
9.a. If the lecturer logs out of the system after adding a relevant module ID for the test then the use case finishes.
11.a. If the lecturer logs out of the system after creating the first question then the use case finishes.
13.a. If the lecturer logs out of the system after creating the answers for the question then the use case finishes.
15.a. If the lecturer logs out of the the test after creating the mark for the question then use case finishes.

16.a. If the lecturer wants to create another question then the lecturer creates another question and the use case continues from step 10. 18.a. If the lecturer logs out of the system before finishing creating the feedback then the use case finishes. 20.a. If the lecturer needs to add a start date and deadline then the lecturer adds the start date and deadline and the use case goes to the final step.
Extension: Create questions which are not just multiple-choice type questions.
Related Use Cases: Manage test(5).
Post-conditions: Test is saved or released into the system.

Author: Spencer Du	Date: 04/01/19	Approved:	Date:	Version

Use Case No: 5	Use Case Name: Manage Tests	Rating: Must have
Purpose: The lecturer must be able to manage a formative or summative test		
Main Actor: Lecturer	Secondary Actors:	
Pre-conditions: Logging into the system		
Trigger: The lecturer clicks 'Manage Test' button to manage the test.		

Basic flow:

- Lecturer clicks on manage test.
- Lecturer chooses the type of the test (formative or summative).
- Lecturer chooses the module of the test.
- Lecturer chooses the name of the test.
- Various pieces of information that the lecturer can modify will be presented to the lecturer on a single form.
- Lecturer can have the ability to modify the name of the test.
- Lecturer can have the ability to modify the type of the test.
- Lecturer can have the ability to modify the module ID of the test.
- Lecturer can have the ability to modify the question of the test.
- Lecturer can have the ability to modify the answer of the question.
- Lecturer can have the ability to modify the mark of the question.
- Lecturer can have the ability to modify the feedback of the question.
- Lecturer can have the ability to modify the start date time and end date time of the summative test.
- Lecturer can have the ability to save the modification after modifying anything.
- Lecturer can have the ability to release the test to the students and must make sure that they don't also release the answers and feedbacks at the same time. The test must be released securely.

Alternative flow:

1. **No test is available to be managed**

Lecturer tries to manage a test. If the test has not been created or saved into the database yet or all tests have been released, then the system should reply with a message that states, 'No test is available to be managed, please create a test first.'

2. **The ID and name of the test cannot repeat**

Lecturer tries to modify the ID and name of the test. If the ID and name of the module already existed, then the system should reply with a message that states, 'The ID or name of the module cannot repeat, please change it.'

3. **The question of the test have not been set when the test is released**

Lecturer tries to release the test, but the question of the test has not been set, then the system should reply with a message that states, 'The question of the test should be set before releasing.'

4. **The answer of the test have not been set when the test is released**

Lecturer tries to release the test, but the answer of the test has not been set, then the system should reply with a message that states, 'The answer of the test should be set before releasing.'

5. **The mark of the test have not been set when the test is released**

Lecturer tries to release the test, but the mark of the test has not been set, then the system should reply with a message that states, 'The mark of the test should be set before releasing.'

6. **The feedback of the test have not been set when the test is released**

Lecturer tries to release the test, but the feedback of the test has not been set, then the system should reply with a message that states, 'The feedback of the test should be set before releasing.'

7. The time of the summative test have not been set when the test is released Lecturer tries to release the summative test, but the start date time and end date time of the test has not been set, then the system should reply with a message that states, 'The time of the formative test should be set before releasing.' **8. Deleting the test** Lecturer can have the ability to delete the test. **9. Discarding a modification** Lecturer can have the ability to discard a modification he made just now.
Extension: Modify questions which are not just multiple-choice type questions.
Related Use Cases: Create test (4)
Post-conditions: Test is saved or released into the system.

Author: QIFA CAI	Date: 04/01/19	Approved:	Date:	Version

Use Case No:06	Use Case Name: View test results	Rating: Must have
Purpose: Allow lecturer to see detailed statistics about the tests.		
Main Actor: Lecturer	Secondary Actors:	
Pre-conditions: Students must have submitted the final attempt for 'formative' tests for the app to calculate and display the results. In Summative tests, students answering the tests should have made their final submission and the deadline must have been reached.		
Trigger: Lecturer instructs the App to show the statistics.		

Basic flow: step by step

1. **Calculate the results:**
   The module will compare each test answer with the pre-loaded correct answer sheet by the lecturer and calculate a mark. Results must be stored in a database

2. **Calculate stats:**
   This module will use the test results and calculate median average mark, highest and lower mark and modal average and store them in the database. Stats must be stored in a database.

3. **Error detection:**
   Module will check that the previous processes finished without errors and that the information is complete, it will count the items submitted and the items counted for the stats, will check that the average mark fall within a range (0-100), All errors detected will be appended to /log.

4. **Show test results:**
   Lecturers can click on the "show test results". The app will provide the list of tests made by the lecturers, and will show which tests have their results and statistics available for viewing and which tests have not been calculated yet and a brief explanation why the stats have not been calculated (Test not yet released, no student have submitted an answer yet, etc). Lecturers can select which test they want to see the test results for.

5. **Plot:**
   App stats module will load the results, results statistics and plot graphs.

6. **Print Out / Export to excell:**
   A button should allow the lecturers to export the test results and statistics to Microsoft Excell or to print out the document on paper.

7. **Choose Range:**
   Lecturers can choose a different range of test results (all year, all module, and so forth). All previous stats are stored in database.

8. **Plot:**
   App will load the previously calculated stats from the database and plot a graph.

9. **Exit:**
   Lecturers will close the app or module.

Alternative flow:

1. **Deadline not reached for Summative Tests:**
   If the deadline has not been reached, and the lecturer asks the module to calculate the stats for said test, the module will display "Deadline not reached". However it will still allow the lecturer to choose a different range of tests to plot.

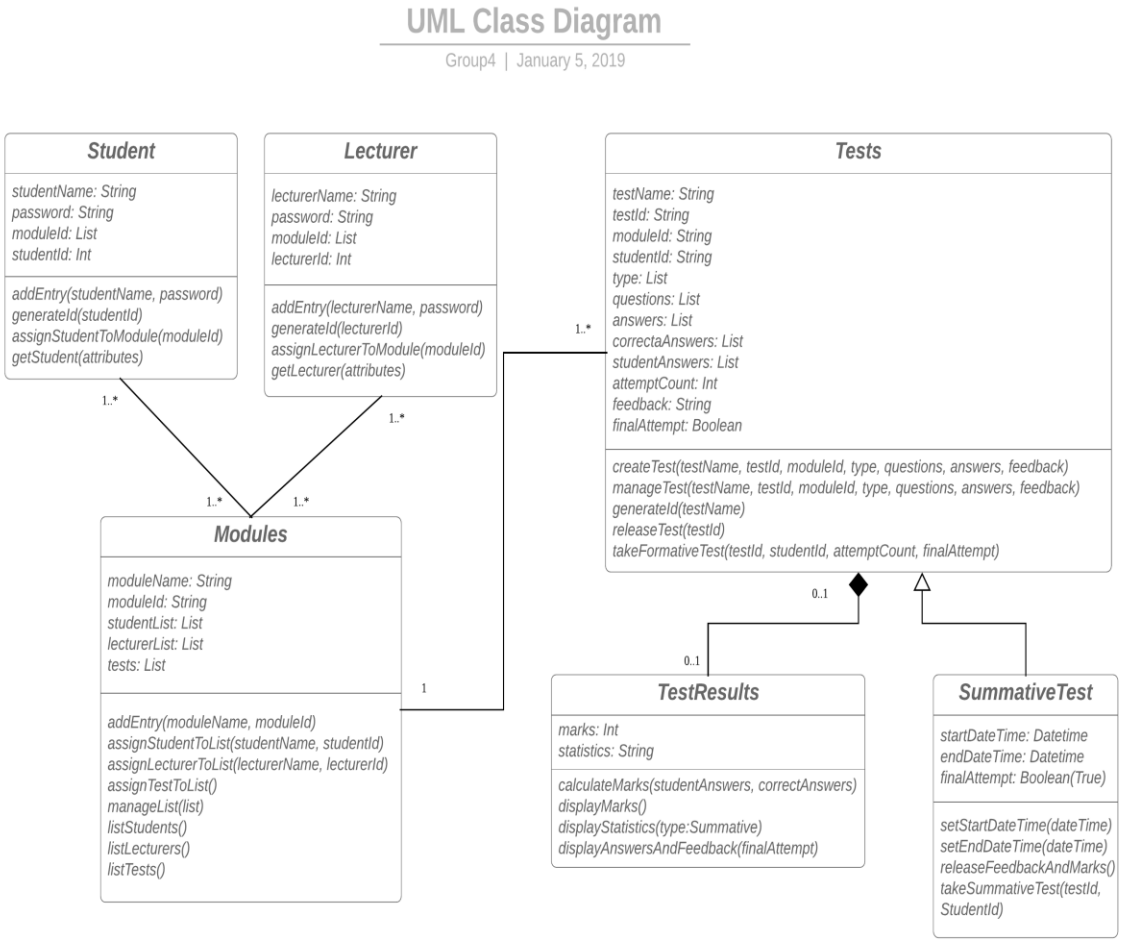
2. **Unanswered test submitted:**
   In the Calculate the results step of the basic flow, if the test that is being market has not being answered at all (opposed to has no right questions answered); the module will flag it as "Did not answer" and award it a mark of 0. The module will report on the Show stats step how many students did not answered the test. The flag "Did not answer" will be stored in database.

3. **All tests submitted left blank:**
   If an unusual high amount of tests were submitted blank the Module will report a warning with the percentage of them that were left blank, but will still calculate and plot the results. Tests left blank will have the "Did not answer" flag on database.

4. **Error detected:**
   If the module detects that the averages does not fall within normal range, or the number of tests submitted is not the same numbers of tests that has been taking into the statistics calculations, the module will attempt to re-load the test marks and run the calculations again. If the error persist, the module will display "Error calculating stats, contact your system administrator" and will refuse to calculate the stats for current test. The system will write on a server log the errors occurred and its details.

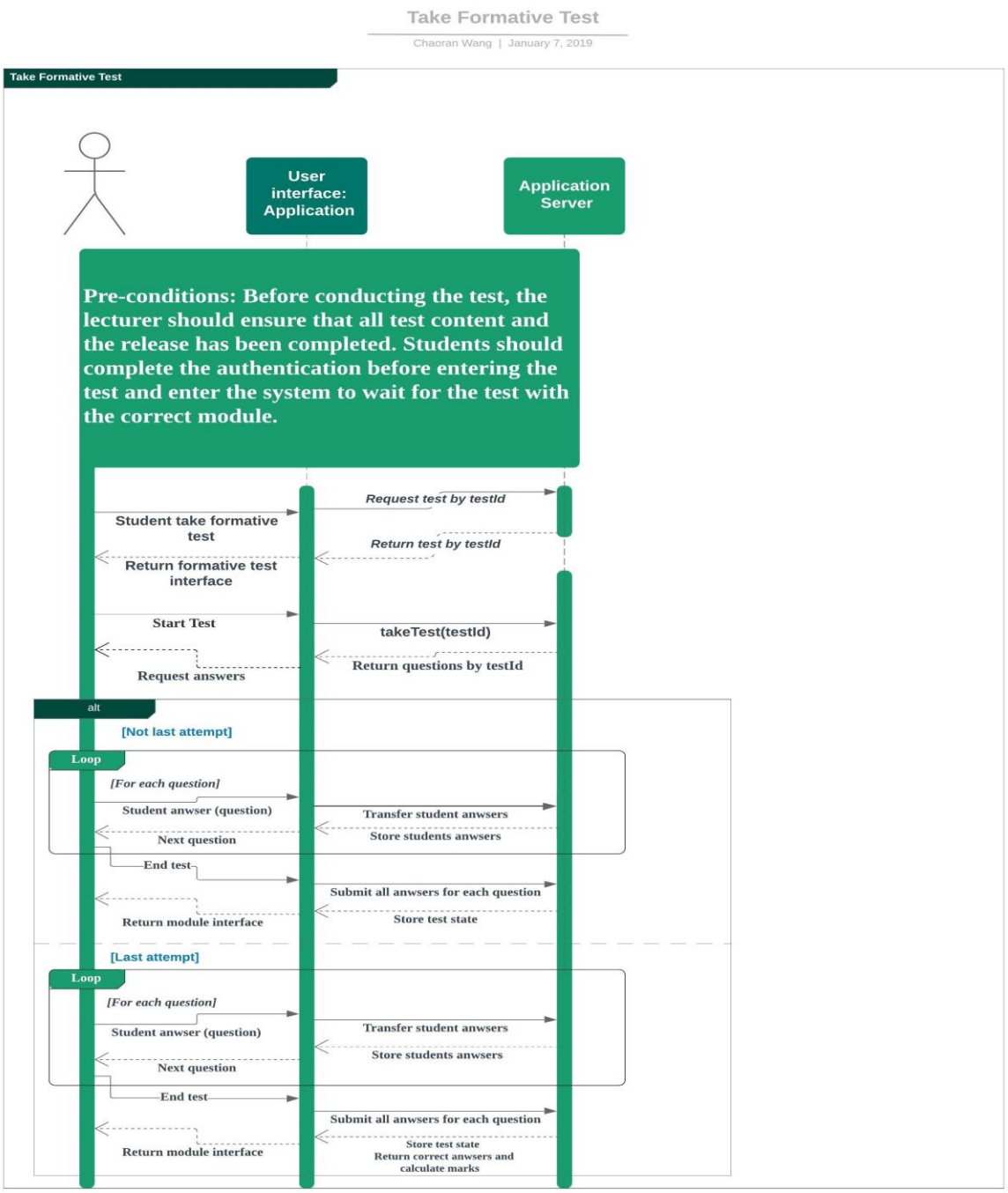
Extensions: Print Out, Export to Excel				
Related Use Cases: Create test, Take formative Test, Take summative test, Access Mark and Feedback				
Post-conditions: Results and statistics will be stored in the database				
Author: Armando Castany	Date: 4- Jan-2018	Approved:	Date:	Version: 0.2

➢ UML Class diagram



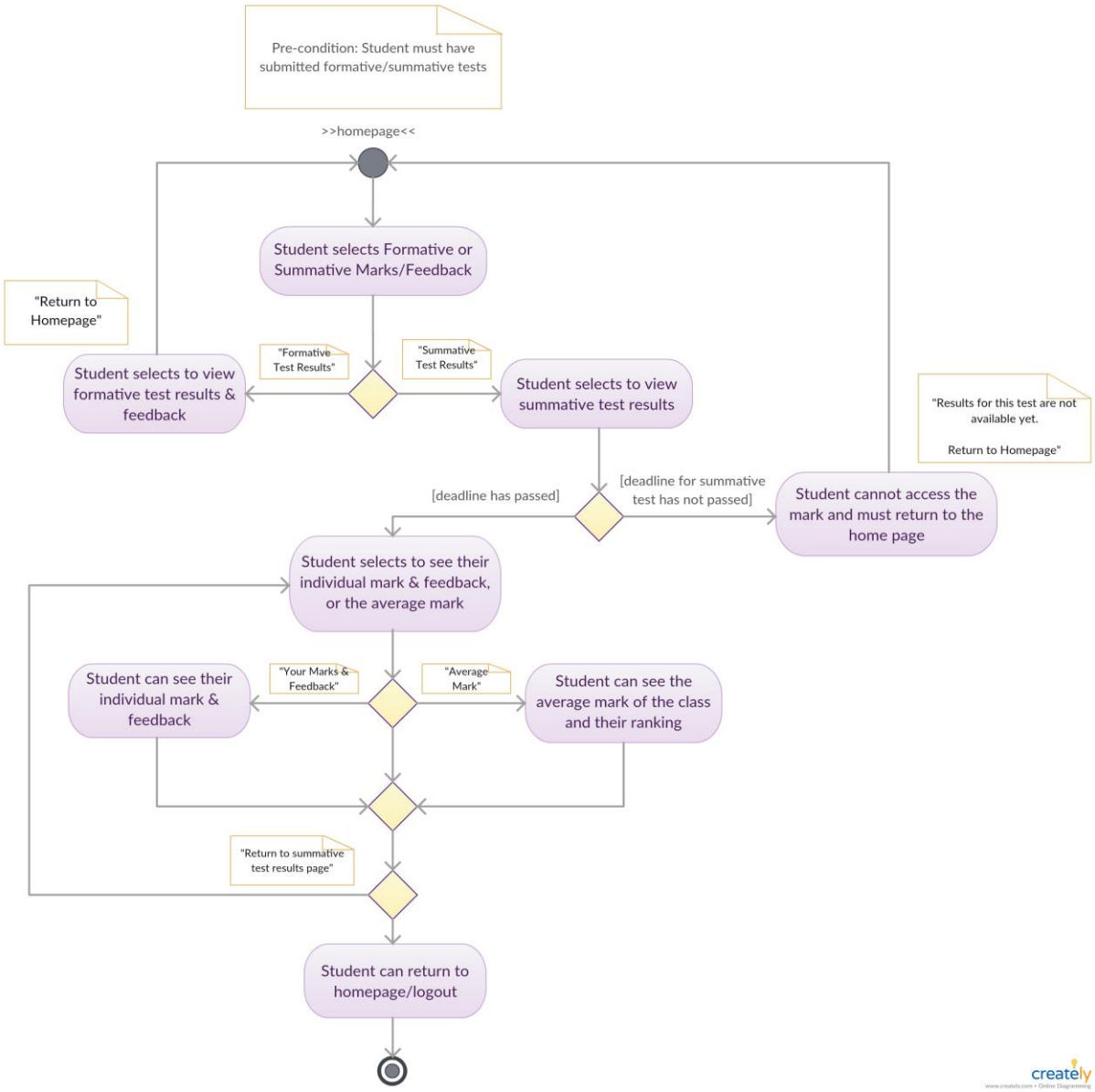
➢ Functionality Use Case UML Diagrams

✧ Use Case 1-Author: Chaoran Wang



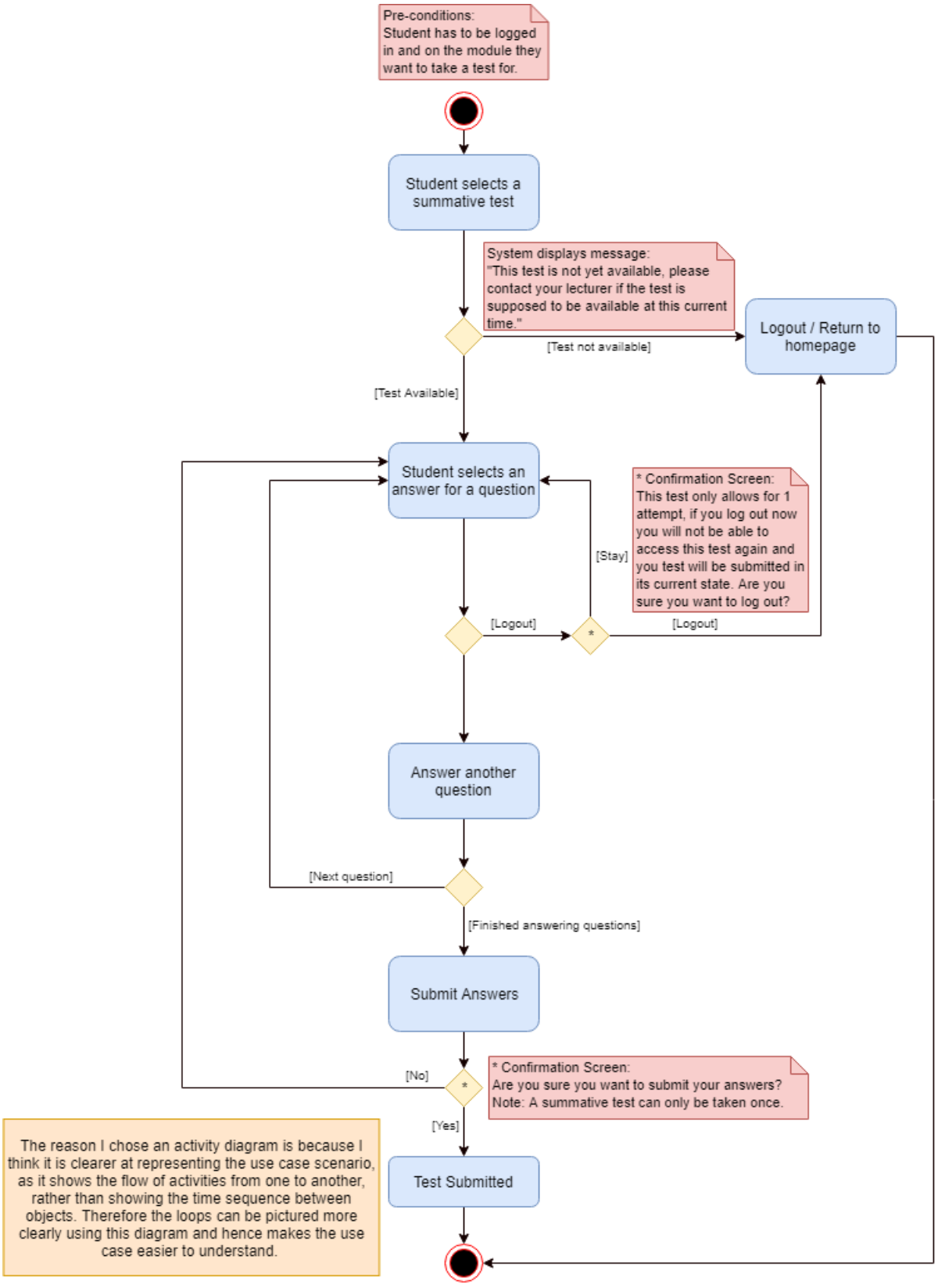
Sequence diagrams are mainly used to more intuitively represent the time sequence of interactions between objects. The emphasis is on time-based, the time sequence in which each object sends and receives messages, processes messages, and returns messages, also known as timing diagrams. The program involves the interaction of three projects: user, platform, and server. So, I think it's a great choice to choose a sequence diagram. Also, because my use case is very close to Zak's, we plan to use different graphs to distinguish them.

✧ Use Case 2-Author: Hannah Rosser

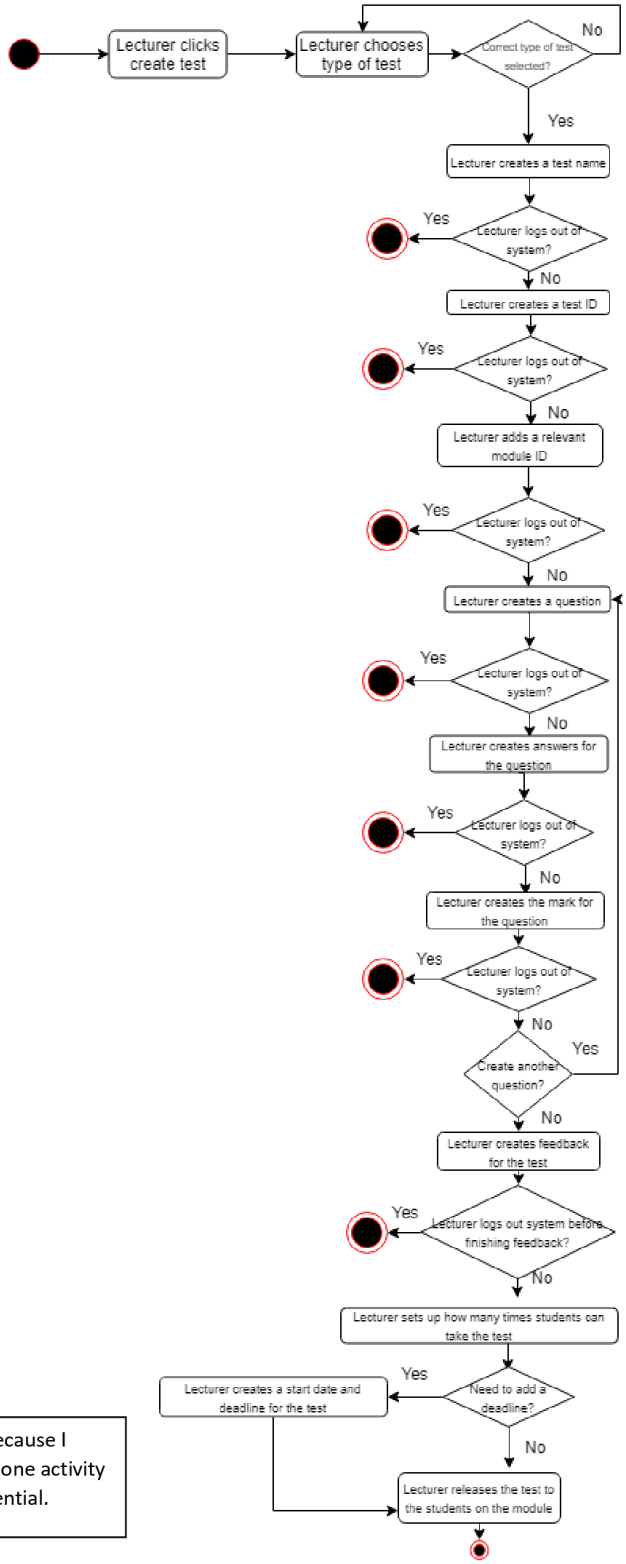


I decided to portray this use case as an activity diagram as it fits this model better. I felt that the use case could be more clearly conveyed when portraying the flow of events and the actions the user takes, rather than showing the interactions between the user and the system.

✧ Use Case 3-Author: Zak Kyriakou

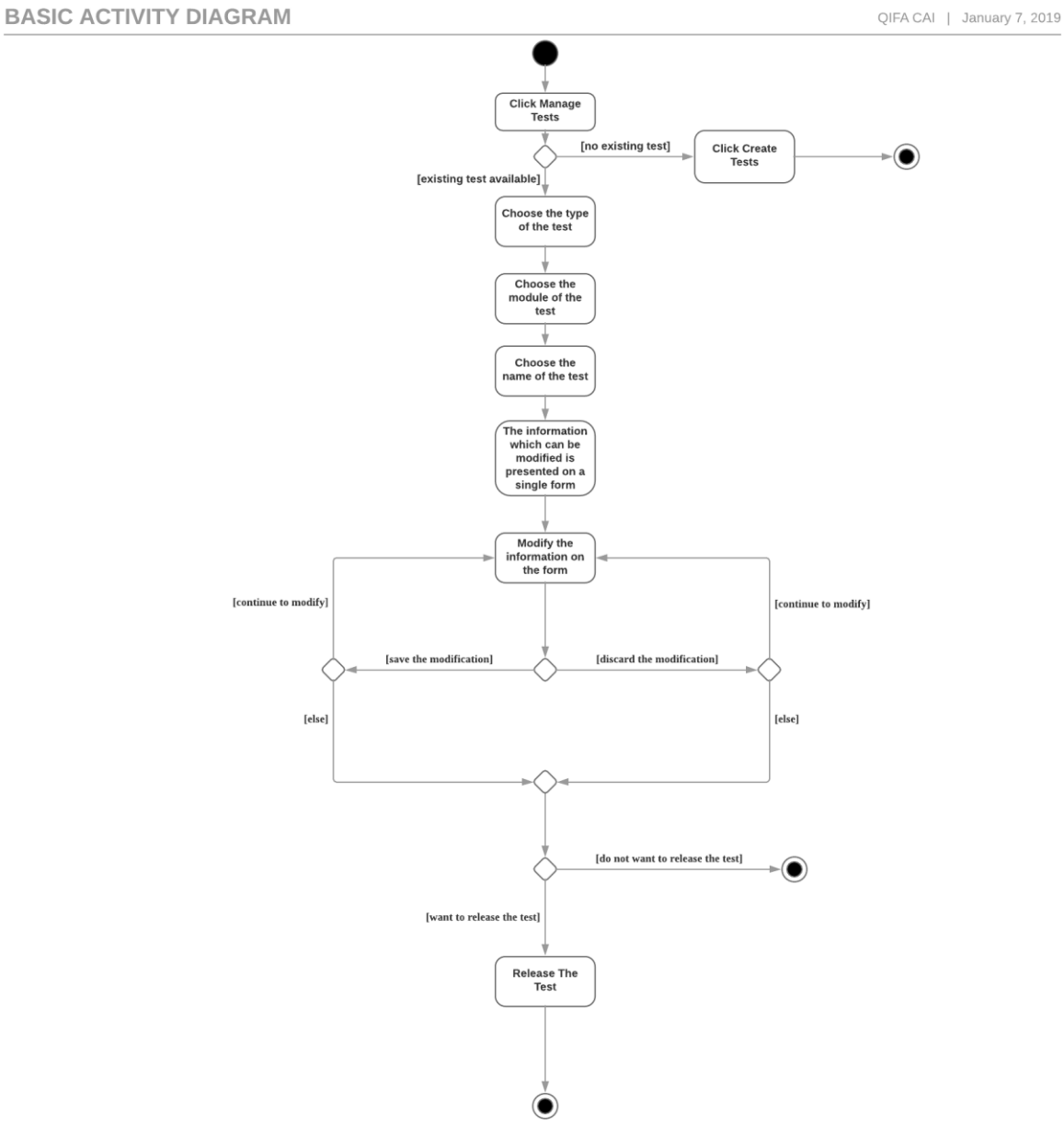


✧ Use Case 4-Author: Spencer Du



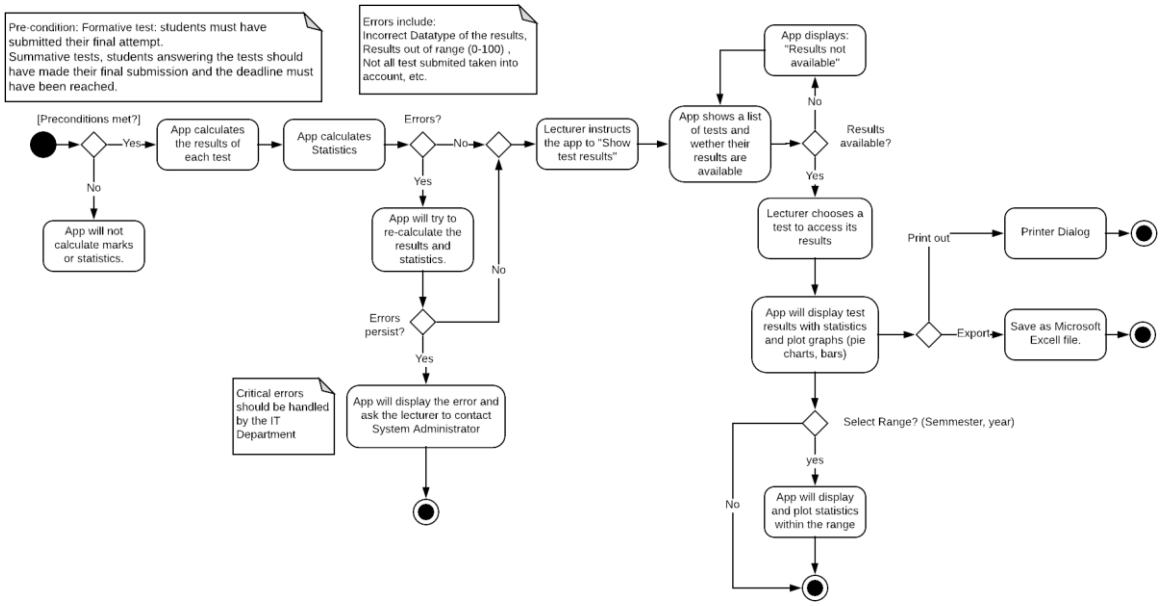
I have chosen to draw a activity diagram because I wanted to show the flow of activities from one activity to another. Also, the activities are all sequential.

✧ Use Case 5-Author: Qifa Cai



I choose activity diagram to show steps of the use case clearly. It also explains the sequence of every steps well. Furthermore, the activity diagram also shows the association between each step.

✧ Use Case 6-Author: Armando Castany



I have chosen to use the activity diagram to emphasize the algorithm of the module, the error detection and the way operate within the module. The activity diagram also shows the steps the user takes for retrieving the desired output of the module and it's easier to visualize than the sequence diagrams.