The University of Melbourne
School of Computing and Information Systems
SWEN90007 Software Design and Architecture
Semester 2, 2020

Project Overview

In this subject, we will discuss a variety of design principles and architectural patterns for enterprise applications. In this project, students will have the opportunity to gain practical experience in the implementation of these patterns by developing an examination platform to facilitate online subject delivery.

The project aims to give students the experience to:

- Produce an architectural design for an enterprise system.
- Choose suitable design patterns to be applied for the different architectural layers.
- Implement an application for your design.
- Develop employability skills: teamwork, good communication, collaboration, problem-solving, critical-thinking, adaptability, initiative and leadership.

Application Domain

In this project you and your team are tasked with implementing an online examination application. Through the application, students can take exams created by subject instructors. Instructors can then use the application to mark exams and publish the results to students.

The application has a single administrator. The administrator has the ability to add subjects to the application. A subject must have one or more instructors and one or more students. The administrator can also view the list of all subjects and their associated details: instructor(s), student(s), and exam(s) (if any). Administrators do not have the ability to create exams.

The application gives instructors access to all the subjects they teach (an instructor can be associated with one or more subjects). Instructors can view all the exams for a particular subject, they can also create, delete, and update exams. A newly created exam is an “unpublished” state by default, this means the exam is visible to subject instructors but not to students. Instructors can then choose to publish the exam, which makes it available to all the students in the subject. Once an exam has been published, it cannot be unpublished.

An instructor can only delete or update an exam if the exam has not been submitted by any student and if there is no student currently taking the exam. Updated exams should only be saved once the instructor has finished applying all the changes, e.g., by clicking a “Save Exam” button after updating all the questions (the instructor should not be able to save updates to individual questions).

Exams have one or more questions and the system will initially support two types of questions: short answer questions and multiple-choice questions. A short answer question allows students to enter their answer in a text box. Multiple-choice questions allow students to select a single answer from at least two choices. Each question has a set number of marks allocated to it. Initially, the system will not support automatic marking of multiple-choice questions.

All exams allow a single attempt, meaning students can only start and submit their exam once. When a student submits an exam, the answers to the questions are saved as part of the student’s submission.

Instructors have the ability to manually close exams, this will associate an exam submission to every student in the subject, even if the student has not submitted the exam. Specifically, when an instructor closes an exam:

- If the student has already submitted the exam, the answers associated with the submission remain unchanged.
- If the student is currently taking the exam (and has not yet submitted), a new submission is created with all the answers recorded as "unanswered".
- If the student has not attempted the exam, a new submission is created with all the answers recorded as "unanswered".

There is no need to immediately inform students currently taking the exam about the exam closure, but a message should be displayed when the students attempt to submit their exam. Exams that have been closed cannot be attempted by students anymore.

Instructors are given two options to enter students' marks into the system:

1. A table view of all the students and the exams in the subject. The instructor can enter the total mark for a student's exam submission in the table entry corresponding to the student and the exam. The decision on when and how to save changes made to the marks in this view is left up to you (e.g., automatically save changes made to any entry in the table, save changes to a single row in the table by clicking a row-specific save button, save changes to the entire table by clicking table-level save button, etc.).
2. A detailed view of a student's submission. In this view, the instructor can see the student's answers and can assign a mark to each question. The total mark is automatically calculated when the instructor saves the changes by adding the individual questions' scores. The system must keep a record of the individual question's mark as well as the total mark.

The detailed view must be accessible from the table view, this can be done by having an extra column for each exam with a link to the detailed view. Marks can be updated in both views. In the detailed view, a marked submission must display the current mark for each question. The instructor can update any of the marks and submit the changes (examples of the table and detailed views are shown in Appendix A).

Students may have access to one or more subjects. For each subject, the student should be able to see and access all of the subject's exams. When attempting an exam, students must be presented with one question at a time. The student then has the ability to answer the question and move on to the next question. The student can go back already answered questions to update them. Updated questions should only be saved once the student submits the exam (the student should not be able to save individual questions).

Programming Language and Architecture

The system that you design and develop must be a web-based system.

The back-end (i.e., server-side components) must be implemented in Java (no exceptions). The use of frameworks to facilitate any infrastructure is not allowed. The idea is for you to implement the patterns from scratch, rather than using a framework with the patterns built in. This will give you hands on experience, strengthen your ability to design and architect applications, and deepen your understanding of the patterns.

For the front-end (i.e., UI/client-side components) of your application, you are free to use any technology/framework/library of your choice. Keep in mind that all workshop and lecture materials will focus on the use of JEE technology, namely JSPs + Servlets. We will provide resources and support on how to set up and use this technology during the lectures and the workshops. We also ask that you consider the fact that integrating JavaScript web UI frameworks with Java-based back-ends may be a non-trivial task. It is your responsibility to design and implement the integration between the front and back ends of your application (e.g., by exposing your application services via a REST API and accessing them using AJAX). We will not cover, nor provide support, on how to do this.

All design diagrams (domain model, class diagram, use case diagrams, sequence diagrams) must be in UML.

Plagiarism

All code and documentation submitted as part of this project must be your own team's work, it must be based on your own knowledge and skills. Collaboration between teams is not allowed, if we find that a submission has been copied from the work of another team, all students submitting the copied work will receive zero marks for the assignment. Similarities will be determined by Turnitin (reports) and Moss (for Measure of Software Similarity – source code).

Teams

This assignment will be carried out in teams of 3, no exceptions. Team members must be enrolled in (and attend) the same workshop. This is because the workshops have been structured to support your teamwork, e.g., by allowing time for the team to work in the project and consult queries with the tutor.

This is a semester long project that will require constant interaction and contribution from all team members, we strongly encourage you to create your own teams and make a conscious choice of whom you will be working with. You are free to team up with friends or peers you are already acquainted with. Students without a team will be randomly allocated into teams by Eduardo and Maria.

A survey to submit your team was created in Qualtrics and a link was posted on Canvas announcement (password: SWEN90007_2020).

Use of Code Repositories and Collaborative Tools

For this project, you are required to create a private Git repository in https://github.com/.

Read and write access must be granted to all of the subject staff:

- Eduardo (agogear)
- Maria (mariaars)
- Eman (eman-kh)
- Najam (najamnazar)

This repository must be used for the duration of your project to manage source code and related configuration items. You should also use it to store a copy of your design documentation (e.g., class diagrams) and the reports submitted for each stage.

Keep in mind that your Git repository will be used to assess individual contributions as part of the peer review process. This means that it is very important that you commit and push your work frequently. It is also good practice to do this as it protects your work from accidental crashes or deletions!

For your submission reports, you are free to choose a collaboration mode that works best for your team. We suggest using a collaborative tool such as Google Docs, it facilitates teamwork, stores copies of your work, and keeps a log of individual contributions to the document. This may come in handy during the peer review process if disputes about contributions arise in the team.

Peer Review

During this journey, students will be asked to participate in two peer assessment survey (Week 6 and Week 11).

This is an individual mandatory task. Everyone needs to participate, assess and provide feedback to peers. Regardless of having a great or bad team experience, students will be encouraged to provide feedback. The survey shouldn't take students longer than 10 minutes to be completed. Students' guidelines for peer review will be provided in the LMS.

The project deliverables (Part 1, Part 2 and Part 3) will be weighted by students' peer review scores. These are continuous assessments focused on communication, teamwork, engagement, commitment and so on. It is essential that everyone can contribute equally to group work. The peer assessment form will give us (teaching team) an indication when this is not the case.

We will apply the following policy to incorporate peer assessment feedback:

- convert each question to a numerical score of between 0 and 5;
- compute the group and individual member averages;
- if a member's score is less than 95% of the group average, your marks will be multiplied by a score weight;
- if you do not participate in this, you'll be scored below the group average (no late submissions will be accepted).

Example: let's say the mark of a group for a deliverable is 20 (out of 20), the group's average peer assessment score is 4.75, member A's score is 4.25, and member B's score is 4.82, then member A gets a final score of: 20*4.25/4.75=17.9 [A's score is less than 95% of group average], while B gets 20 [B's score is above group average].

Teamwork during Workshops

Each workshop will have a certain amount of time dedicated for project teamwork. You are encouraged to think of this time as your team's weekly meeting. You can use the time to plan activities for the week ahead, work collaboratively on a task, discuss any issues that need to be resolved, etc. Most importantly, your team should use this time to seek clarification and guidance from the tutor.

You are required to take minutes during these workshop team meetings. The minutes should contain:

- Workshop date/week.
- Team members that attended.
- A few bullet points summarizing the team's activities during the workshop meeting.

These minutes must be stored in your Git repository and will be used as part of the peer review process.

Detailed Project Description

Part 1 [5 marks]

Task:

Given the Application Domain description, identify the use cases necessary to meet the requirement specification for the **instructor** and **student** users.

Deliverables:

1. A report, in pdf format, that includes the following:

   - Your team's name, team members' names, student ids, unimelb usernames and emails.
   - A use case diagram that illustrates the interaction between the actors and the use cases, as well as any relevant relationship between use cases.
   - A detailed description (see Appendix B for a sample template) of each use case. Each use case must contain: the list of actors associated with the use case, a high-level description of the use case, the typical course of events, and the main alternative courses of action.
   - A link to your private Git repository. Before the submission deadline, all the teaching team must have read and write access to your repository. A copy of this stage's report should already be stored in your repository.
   - A Git release tag: you must create a release tag in your Git repository for this deliverable in the following format: SWEN90007_2020_Part1_<team name>[1]. The tag must be created before the submission deadline as this will be used to assess your deliverable (no exceptions). The created release tag must be added to your report.

Part 2 [22 marks]

Task:

In this part you will design and implement the examination app as described in Application Domain. At this stage, you will assume that there is a single instructor per subject. This means that for each subject, a single instructor has the ability to create, update, and mark exams.

You must include the following patterns in your design and implementation (see Appendix C for more details):

- Domain model
- Data mapper
- Unit of work
- Lazy load
- Identity field
- Foreign key mapping
- Association table mapping
- Embedded value
- One of the inheritance patterns

Deliverables:

[1]Git Basics – Tagging: https://git-scm.com/book/en/v2/Git-Basics-Tagging

1. The application coded and deployed. The source code of your application must be committed to your Git repository. The application must be deployed in Heroku.
2. You must create a tag in your Git repository for this deliverable in the following format: SWEN90007_2020_Part2_<team name>. The tag must be created before the submission deadline as this will be used to assess your deliverable (no exceptions).
3. A Software Architecture Design (SAD) report with the following items:
    - The domain model of your application.
    - The class diagram of your application.
    - A description of all patterns used. The description must be contextualized, this means it must include details of how the pattern was implemented as part of your particular application design. A sequence diagram illustrating the use of the pattern in your application must be included as part of your description.
    - Design rationale for unit of work and lazy load: an explanation on where the patterns were used and why.
    - A link to your Heroku deployed app, including a populated database with a range of realistic data samples/information that are necessary for the teaching team to test your application. You also need to provide additional instructions on how to use the existing data that you created in your system, for example: the administrator username and password. We do not expect to create any test data from scratch. Again, this is your project. Make sure you test your deliverable before your submission and that you have real/appropriate data in the deployed system. Meaningless data such as abc, 123, blah blah, subject ASDFG and so on will compromise the assessment of your project (and your final marks). See Appendix D for more details.
    - The Git release tag.

Note that for the UML diagrams, it is your responsibility to decide the level of detail required to appropriately convey the design and implementation of your application. Use your judgement and knowledge in software engineering to make this decision; another software engineer, new to your team, should be able to implement your design based on the information contained in the SAD report.

Part 3 [13 marks]

Task:

In this part, the ability to have multiple instructors per subject will be enabled. This means that for a particular subject, multiple instructors can be creating, updating, and marking exams simultaneously. This gives place to concurrency issues that you must address. In this stage you will also secure your platform.

Concurrency: There are multiple scenarios that will give place to concurrency issues (some easier to manage that others):

- Multiple instructors can create exams for the same subject simultaneously.
- Multiple instructors can update the same exam simultaneously.
- Multiple instructors can enter marks for the same student cohort simultaneously (in the table view).
- Multiple instructors can enter marks for the same exam and the same student (in the detailed view) simultaneously.

To manage these concurrency issues, you are required to make use of the concurrency patterns covered in the lectures. You can choose to use a pessimistic or an optimistic online lock (or both). You are required to present a convincing argument for your choice of pattern(s). In this stage, you can make any changes you deem necessary (but that abide by the specification) to the way in which instructors are allowed to save marks in the table view.

Security: You are asked to implement the following security patterns: Authentication, Authorization, and Secure Pipe. You are free to use any library/framework to implement these patterns.

Deliverables:

1. The application coded and deployed. The source code of your application must be committed to your Git repository. The application must be deployed in Heroku.
2. You must create a tag in your Git repository for this deliverable in the following format: SWEN90007_2020_Part3_<team name>. The tag must be created before the submission deadline as this will be used to assess your deliverable (no exceptions).
3. A Software Architecture Design (SAD) report with the following items:
    - The updated class diagram of your application, with any updates made to it highlighted.
    - A description of all patterns used. The description must be contextualized, this means it must include details of how the pattern was implemented as part of your particular application design. A sequence diagram illustrating the use of the pattern in your application must also be included as part of your description.
    - An explanation and justification supporting your choice of concurrency pattern(s).
    - A link to your Heroku deployed App including data/information that we need in order to be able to test your application.
    - The Git release tag.

NOTE: There is not only one correct answer in terms of pattern choice. Patterns should be chosen based on the specific needs of the enterprise system (context) and justified accordingly. Discuss differences, pros and cons in regard to your specific project.

Submissions

Submission of reports will be via the LMS, under the Assignments menu. All reports must be submitted as a PDF file (no exceptions). Only one member of the team should submit reports via the LMS.

For the source code, we will assess your RELEASE TAGS for marking purposes. Make sure you generate the tags and add them to your PDF report before the submission deadline.

Following are the project deliverables (submissions) throughout the semester, and the marks associated with the different submissions (adding up to a total of 40 marks).

Deliverable	Marks	Due Week	Due Date
Part 1	5	3	Sunday 23 Aug, 11:59pm
Part 2	22	9	Sunday 4 Oct, 11:59pm
Part 3	13	12	Sunday 1 Nov, 11:59pm

Feedback will be provided for each stage within two weeks of submission.

Extensions

Extensions will only be granted if students are able to provide appropriate supporting documentation. If an extension is needed you must contact the lecturer as soon as possible, before the submission date. It will not be possible to apply for an extension after the submission date.

For further details please see:
https://ask.unimelb.edu.au/app/answers/detail/a_id/5667/~/applying-for-an-extension

Late Submissions

Unless an extension is in place, a 10% penalty will be applied to every day you delay your submission (PDF and/or RELEASE TAG).

Example: Part 2 - Deadline October 4, 11.59PM

Scenario 1: For Part 2 deliverable, you can score 22 maximum. You and your team submitted the PDF report on Canvas before the deadline (the day before, for example). However, two days later you were having a coffee together and realised that you did not create a release tag to your source code. You finish your coffee quickly, go to GitHub and generate a release tag to your source code. Unfortunately, as that happened two days after the deadline, this means we will need to work with the calculator: 2 days x 10% = 20%. Your maximum possible mark for this deliverable is now 17.6.

Scenario 2: You and your team submitted the PDF report on Canvas before the deadline (the day before, for example). On Sunday afternoon (the BIG deadline day), you are watching the footy on TV and realise that you did not create a release tag to your source code. As it is still lunch time, you finish watching the game, celebrate your team's victory, go to GitHub and generate a release tag to your source code. Luckily, as much as the resources were submitted and generated in different days, everything happened before Part 2 deadline. No calculator needed for you and your team. Your maximum possible marks remain unchanged.

Assessment Criteria

The goal of this project is for you to apply and demonstrate a clear understanding of the design principles and architectural patterns covered in the subject. Your project will be assessed based on the following high-level criteria:

- Complete implementation of the functionality specified in the application domain.
- Clear, complete, and correct presentation of the application's design and architecture using UML.
- Correct implementation of the patterns.
- Design quality and rationale. This includes appropriate selection of patterns (if applicable), sound reasoning behind their selection, and accurate explanations of the design principles and patterns used.

Note that proper use of UML notation is expected. Inaccuracies in your UML diagrams will affect your mark. In particular, you are expected to produce class diagrams, use case diagrams, and sequence diagrams. Also, your implementation must match the design presented in the documents. Any discrepancies between code and documentation will impact your mark.

Please be aware that simply producing an application that provides the required functionality is not enough to pass the assessment.

Assessment breakdown

Part 1

Criterion	*Description*	*Marks*
Use case diagram	Correctly identified use cases and their relationships. Correct use of UML notation.	1
Detailed use cases	Typical course of events matches behaviour described in specification. Alternative courses match behaviour described in specification. Appropriate level of detail and abstraction. Required sections are present.	4
Total		*5*

Part 2

Criterion	*Description*	*Marks*
Domain model diagram	Correct use of UML notation, appropriate level of detail and abstraction.	1
Class diagram	Correct use of UML notation, appropriate level of detail and abstraction.	2
Pattern description	Correct, complete, and contextualised implementation details for each pattern, correct sequence diagrams with an appropriate level of detail.	9
Design rationale (UoW and Lazy Load)	A clear and convincing rationale for the use of unit of work and lazy load patterns in the design of the application.	2
Functionality and pattern implementation	A correct and complete implementation of the architectural design, including a correct implementation of the patterns and application functionality.	8
Total		*22*

Part 3

Criterion	*Description*	*Marks*
Class diagram	Correct use of UML notation, appropriate level of detail and abstraction.	2
Pattern description	Correct, complete, and contextualised implementation details for each pattern, correct sequence diagrams with an appropriate level of detail.	3.5
Design rationale (concurrency pattern(s))	A clear and convincing rationale for the use of concurrency patterns in the design of the application.	2
Functionality and pattern implementation	A correct and complete implementation of the architectural design, including a correct implementation of the patterns and application functionality.	5.5
Total		*13*

Appendix A – Screenshots of Canvas Quizzes UI

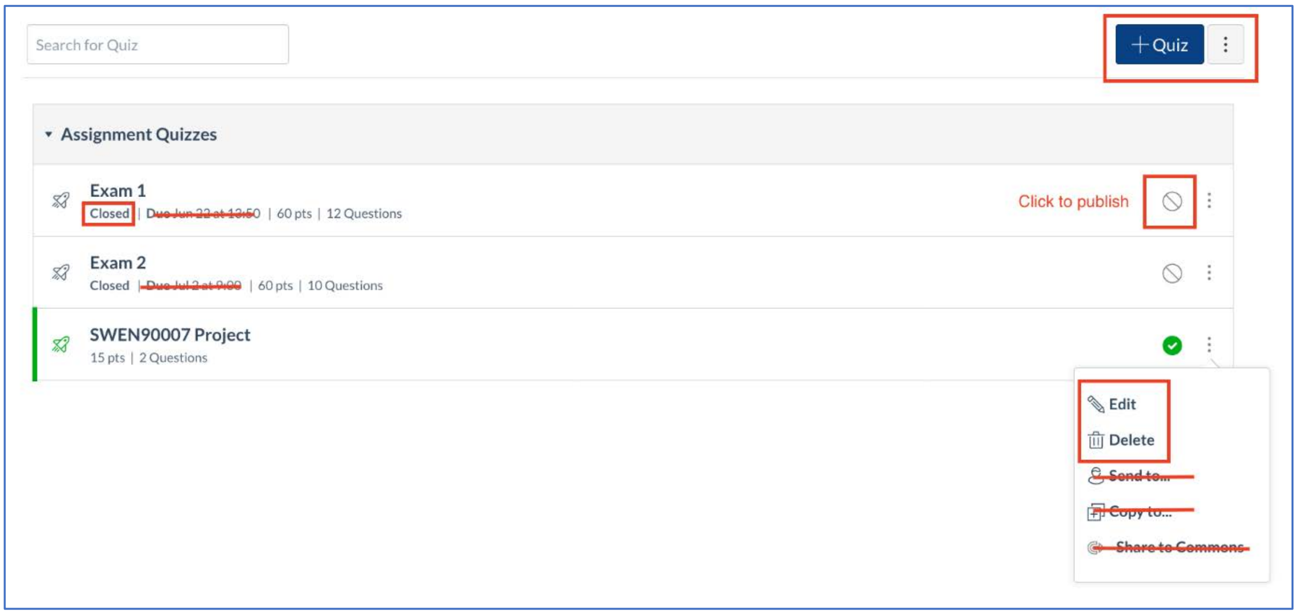
The goal of these screenshots is to clarify the application requirements. They are an **example** of a user interface that provides some of the required functionality. **You are not expected to mimic this interface in your project, you are free to design your application in any way you want**.

Please keep in mind that:

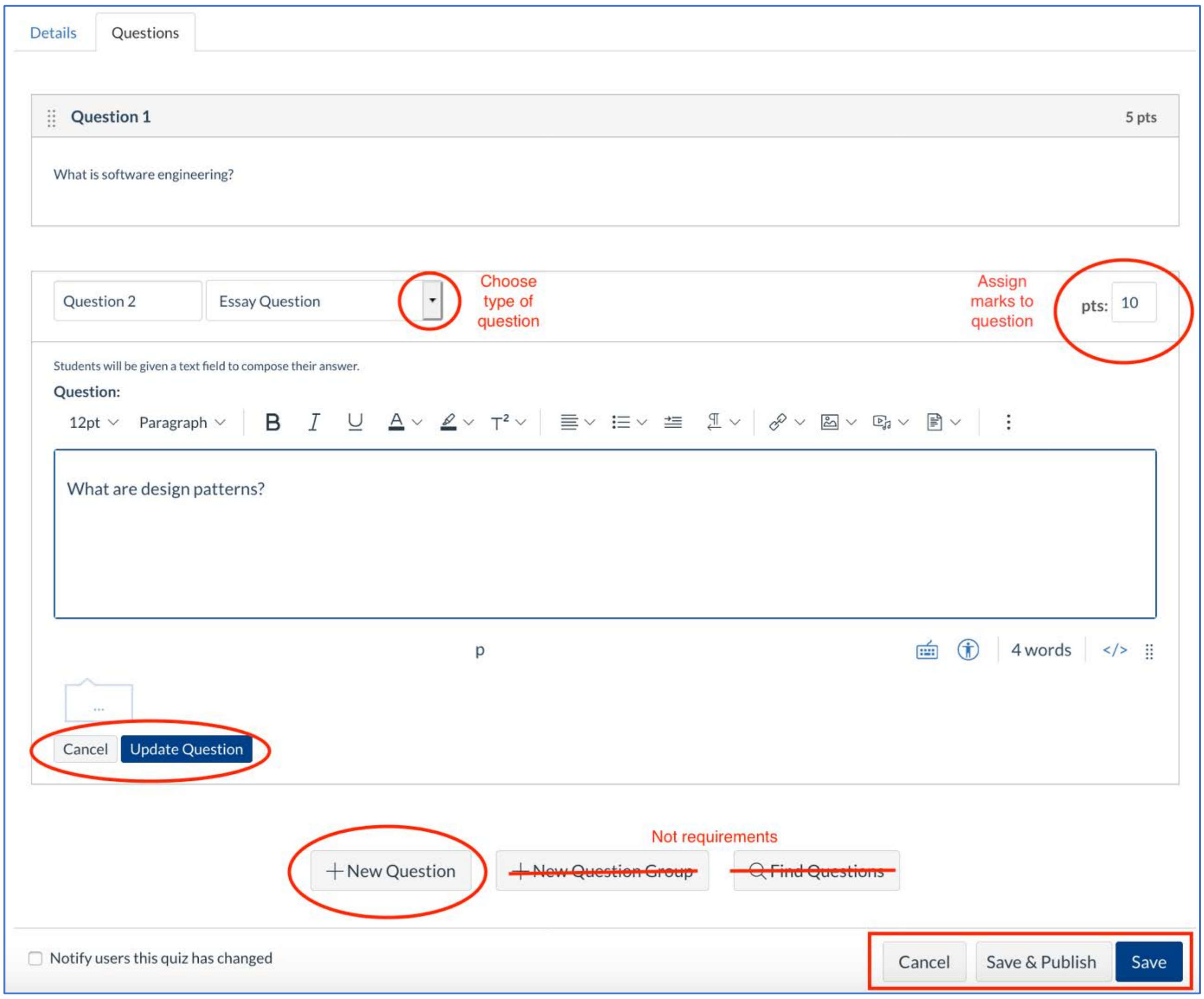
- The screenshots cover some of the functionality you are required to implement, but not all of it.
- The screenshots may depict additional functionality that you are not required to implement, the red boxes/circles highlight the functionality you should focus on.

Creating Exams

Adding, editing, deleting, and publishing exams:



Adding questions to an exam:



Question 1 5 pts

What is software engineering?

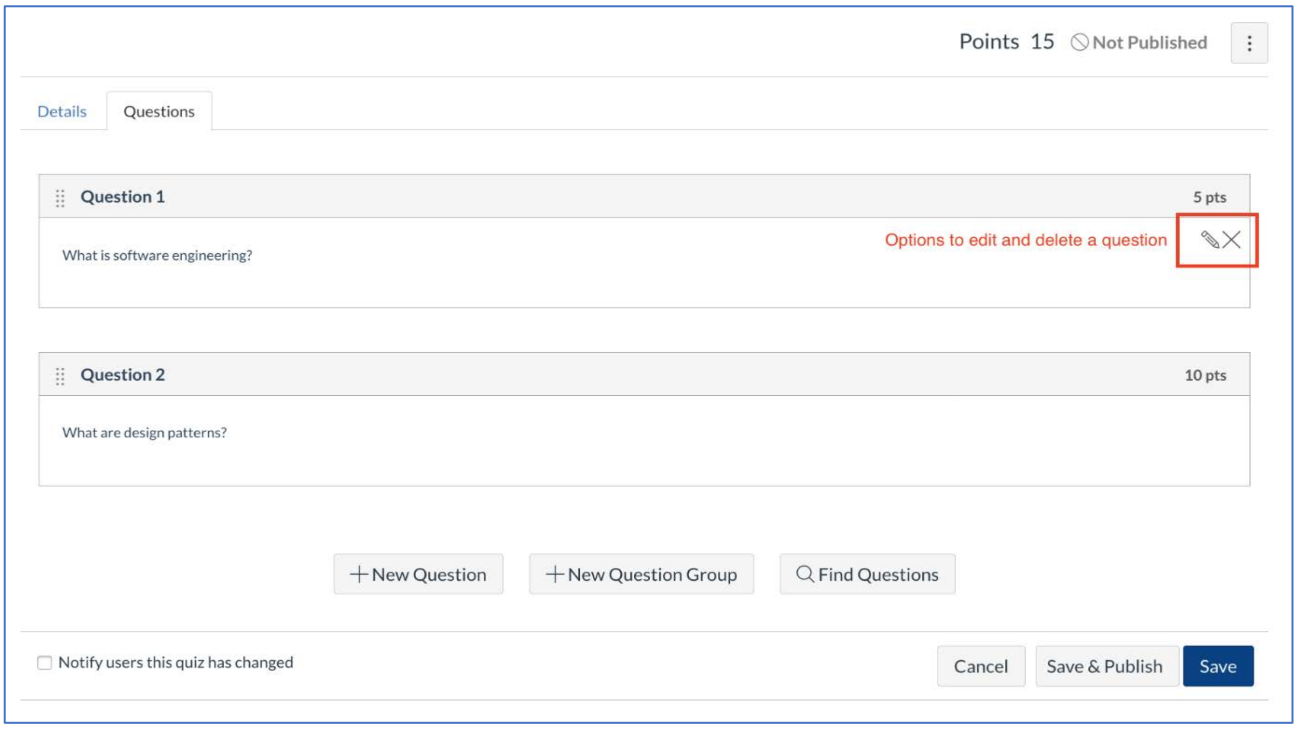
Question 2 | Essay Question | pts: 10

Students will be given a text field to compose their answer.

Question:

What are design patterns?

Editing and deleting exam questions:


Points 15 Not Published

Details Questions

Question 1 5 pts

What is software engineering?

Options to edit and delete a question

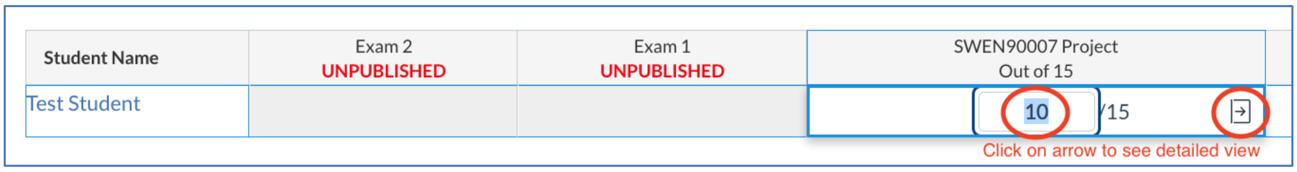
Question 2 10 pts

What are design patterns?

Notify users this quiz has changed

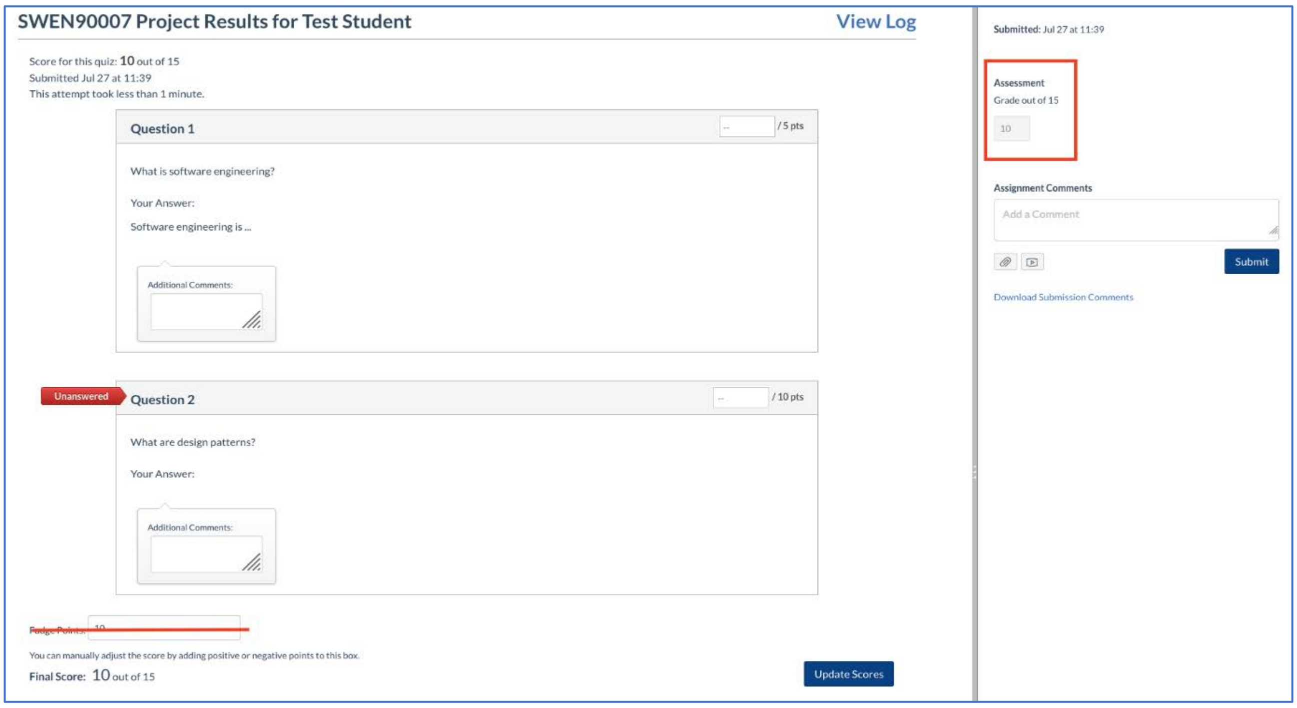
Marking

Table view:


Student Name	Exam 2 UNPUBLISHED	Exam 1 UNPUBLISHED	SWEN90007 Project Out of 15
Test Student			10 /15

Click on arrow to see detailed view

Detailed view (after entering marks in table view):


SWEN90007 Project Results for Test Student

View Log

Score for this quiz: 10 out of 15
Submitted Jul 27 at 11:39
This attempt took less than 1 minute.

Question 1 -- / 5 pts

What is software engineering?

Your Answer:

Software engineering is ...

Additional Comments:

Unanswered

Question 2 -- / 10 pts

What are design patterns?

Your Answer:

Additional Comments:

~~Fudge Points: 10~~

You can manually adjust the score by adding positive or negative points to this box.

Final Score: 10 out of 15

Update Scores

Submitted: Jul 27 at 11:39

Assessment
Grade out of 15
10

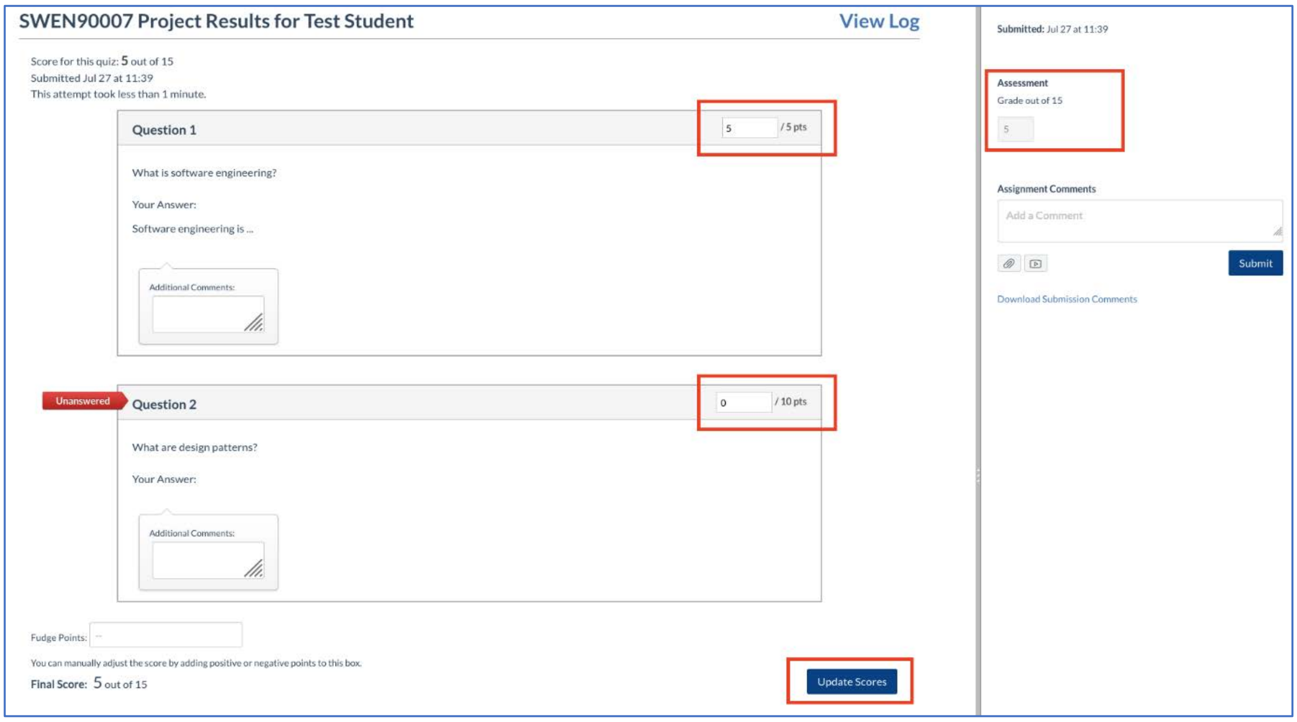
Assignment Comments

Add a Comment

Submit

Download Submission Comments

Detailed view (after entering marks for each question and updating scores):


SWEN90007 Project Results for Test Student

View Log

Score for this quiz: 5 out of 15
Submitted Jul 27 at 11:39
This attempt took less than 1 minute.

Question 1 5 / 5 pts

What is software engineering?

Your Answer:

Software engineering is ...

Additional Comments:

Unanswered

Question 2 0 / 10 pts

What are design patterns?

Your Answer:

Additional Comments:

Fudge Points: --

You can manually adjust the score by adding positive or negative points to this box.

Final Score: 5 out of 15

Update Scores

Submitted: Jul 27 at 11:39

Assessment
Grade out of 15
5

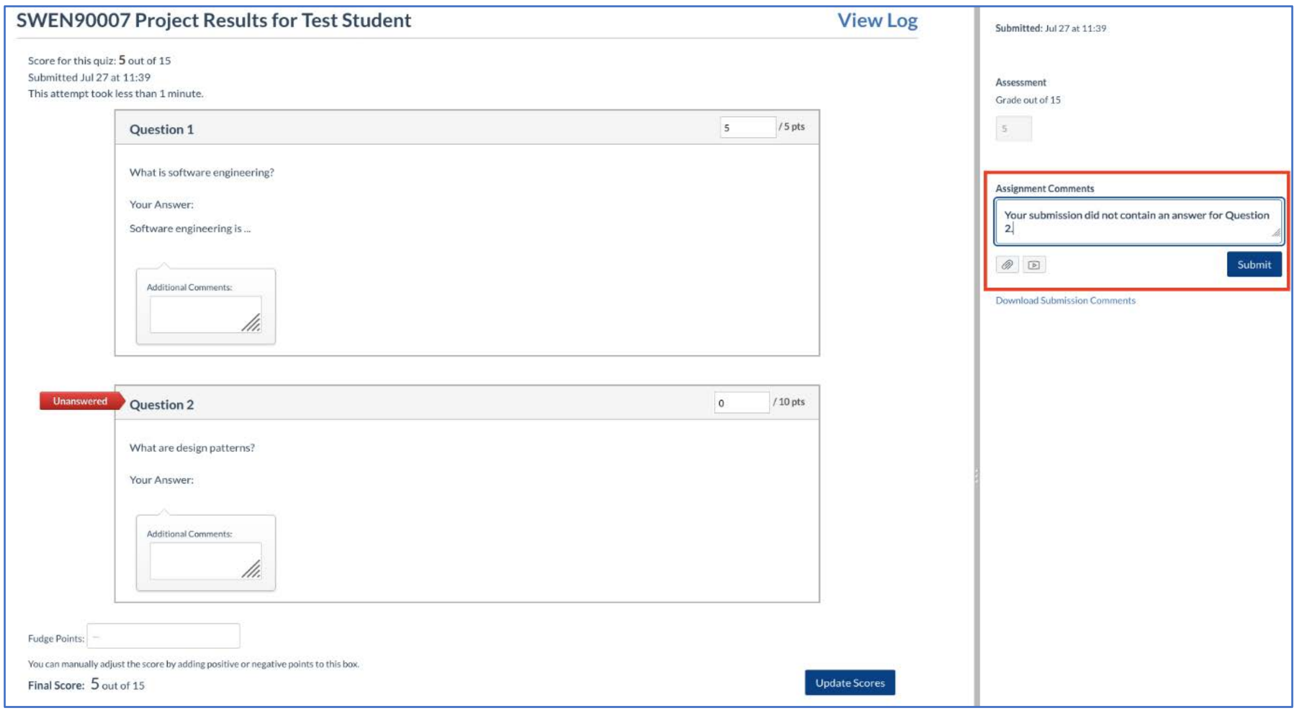
Assignment Comments

Add a Comment

Submit

Download Submission Comments

Entering comments in detailed view:



SWEN90007 Project Results for Test Student

View Log

Score for this quiz: 5 out of 15
Submitted Jul 27 at 11:39
This attempt took less than 1 minute.

Question 1 — 5 / 5 pts

What is software engineering?

Your Answer:

Software engineering is ...

Additional Comments:

Unanswered — Question 2 — 0 / 10 pts

What are design patterns?

Your Answer:

Additional Comments:

Fudge Points: --

You can manually adjust the score by adding positive or negative points to this box.

Final Score: 5 out of 15

Update Scores

Submitted: Jul 27 at 11:39

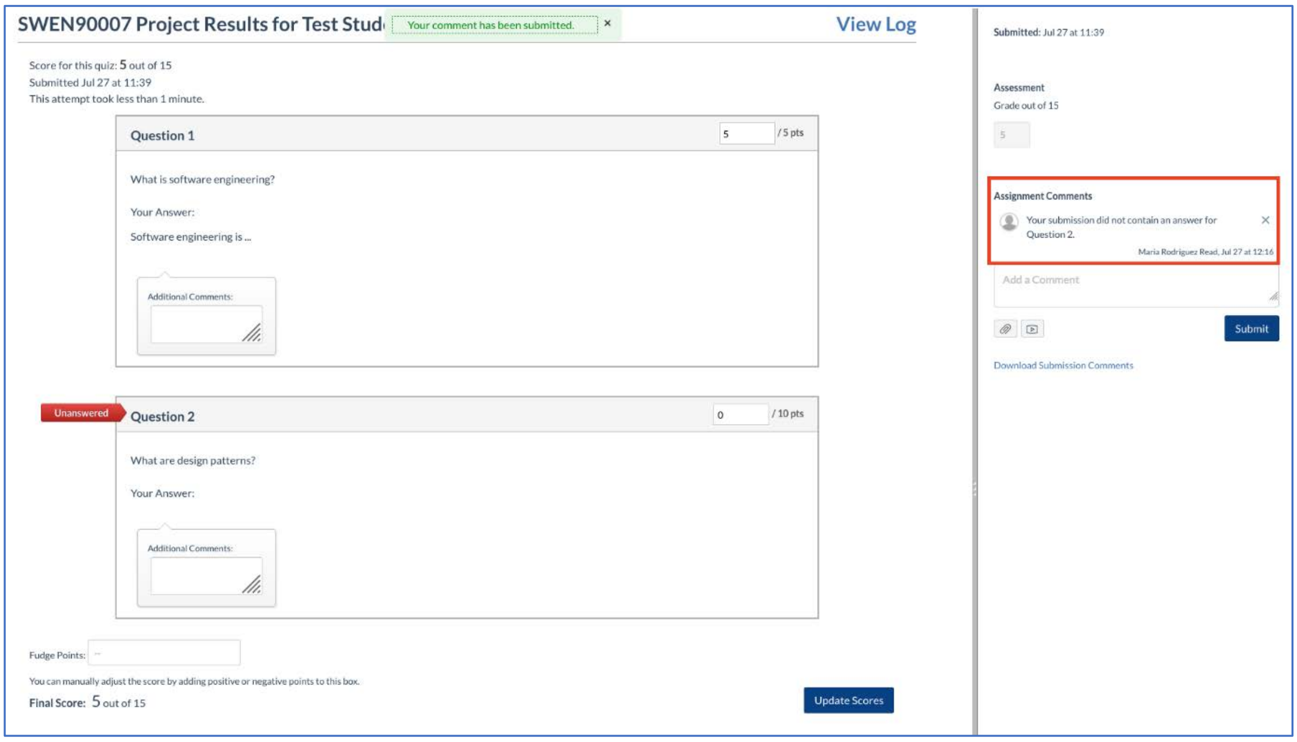
Assessment
Grade out of 15
5

Assignment Comments

Your submission did not contain an answer for Question 2.

Submit

Download Submission Comments



SWEN90007 Project Results for Test Stud

Your comment has been submitted.

View Log

Score for this quiz: 5 out of 15
Submitted Jul 27 at 11:39
This attempt took less than 1 minute.

Question 1 — 5 / 5 pts

What is software engineering?

Your Answer:

Software engineering is ...

Additional Comments:

Unanswered — Question 2 — 0 / 10 pts

What are design patterns?

Your Answer:

Additional Comments:

Fudge Points: --

You can manually adjust the score by adding positive or negative points to this box.

Final Score: 5 out of 15

Update Scores

Submitted: Jul 27 at 11:39

Assessment
Grade out of 15
5

Assignment Comments

Your submission did not contain an answer for Question 2.
Maria Rodriguez Read, Jul 27 at 12:16

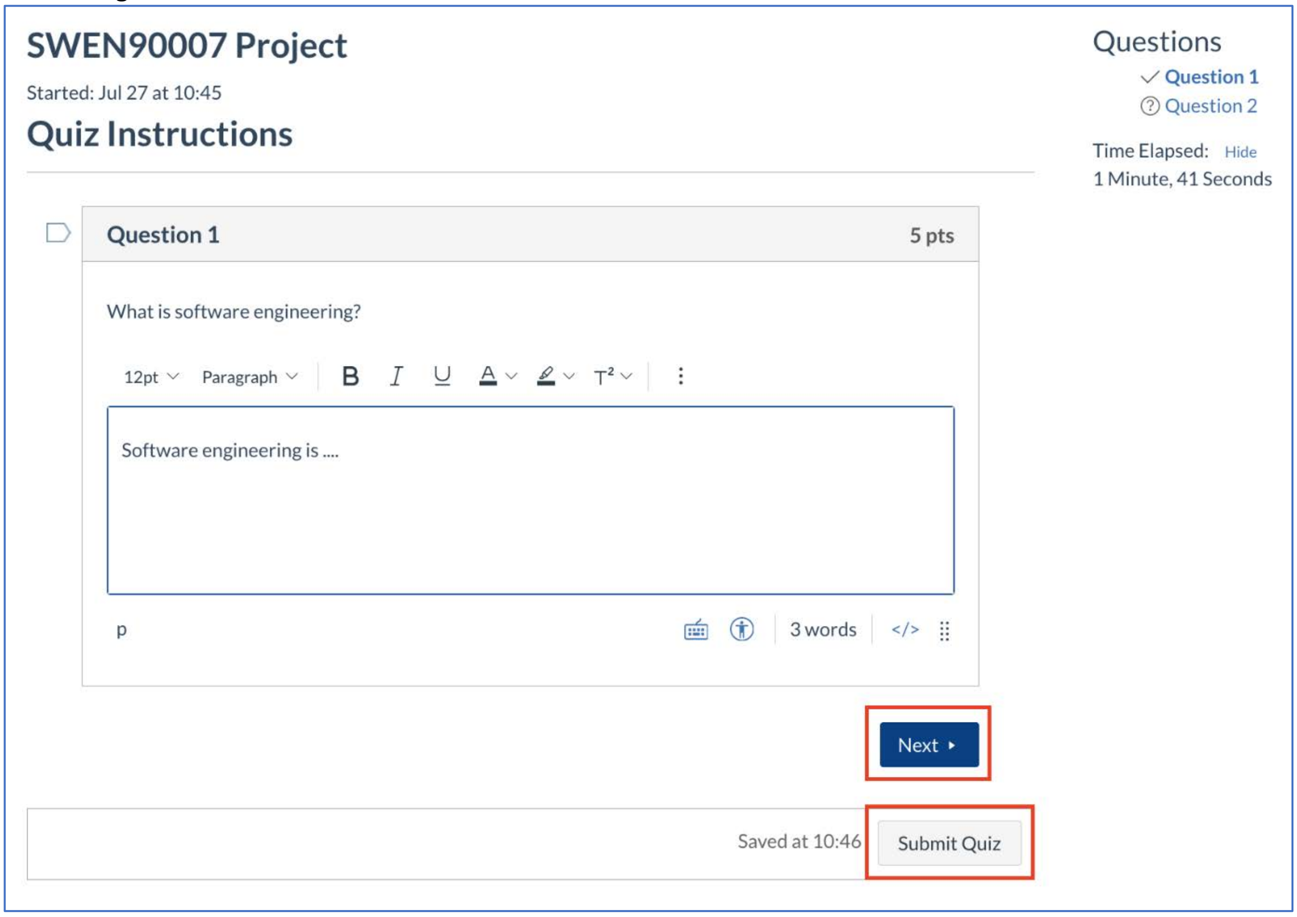
Add a Comment

Submit

Download Submission Comments

Student Submissions

Answering an exam:



SWEN90007 Project

Started: Jul 27 at 10:45

Quiz Instructions

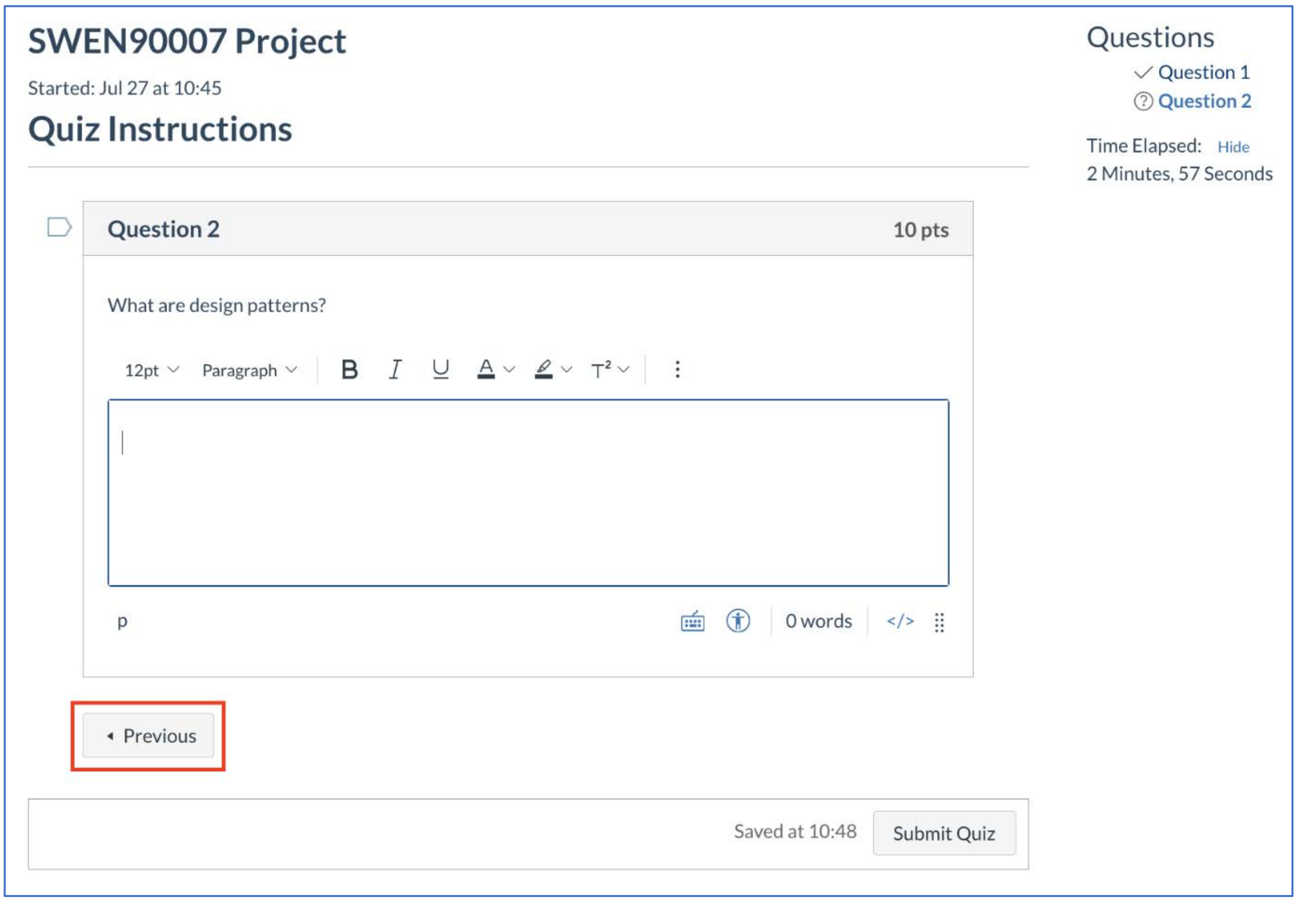
Question 1 5 pts

What is software engineering?

Software engineering is ....

3 words

Saved at 10:46



SWEN90007 Project

Started: Jul 27 at 10:45

Quiz Instructions

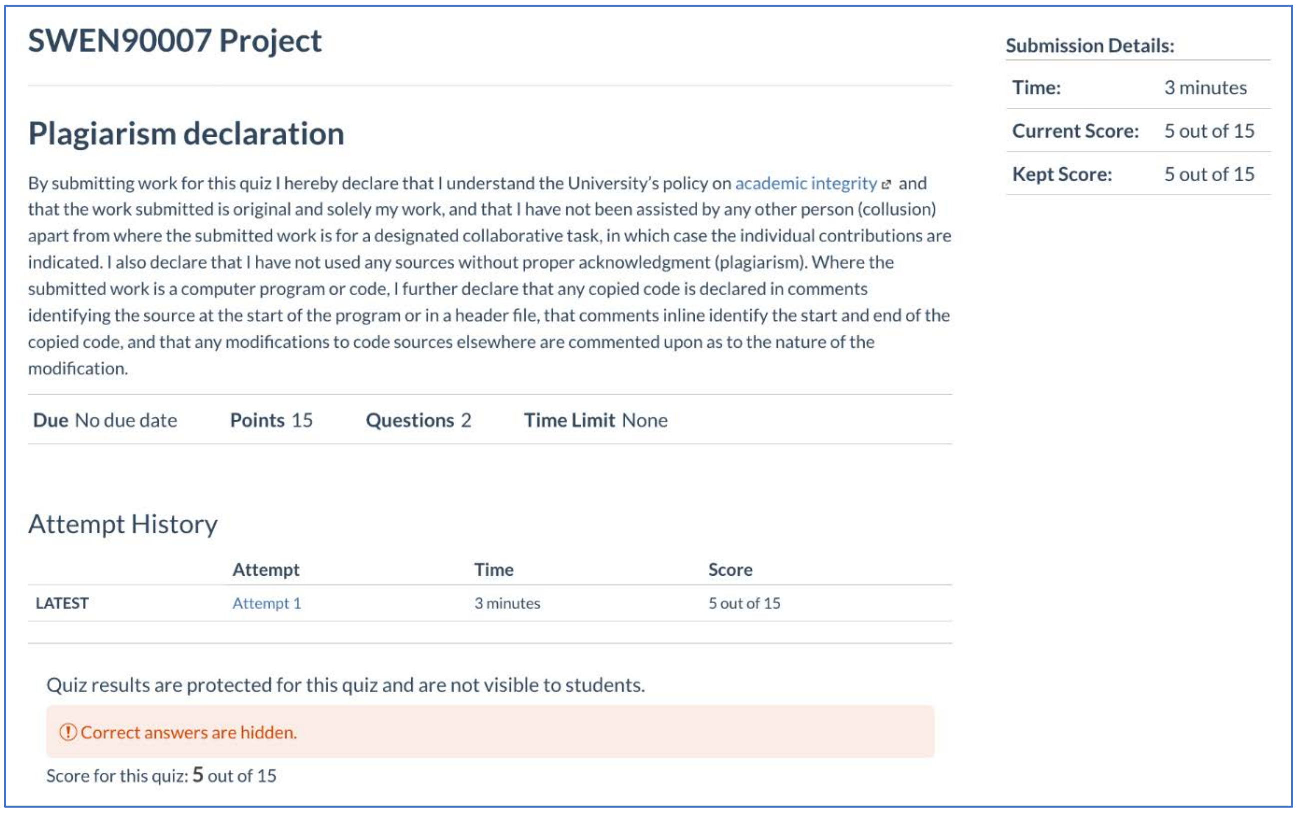
Question 2 — 10 pts

What are design patterns?

0 words

Saved at 10:48

Viewing exam results for a marked exam:



SWEN90007 Project

Plagiarism declaration

By submitting work for this quiz I hereby declare that I understand the University's policy on academic integrity and that the work submitted is original and solely my work, and that I have not been assisted by any other person (collusion) apart from where the submitted work is for a designated collaborative task, in which case the individual contributions are indicated. I also declare that I have not used any sources without proper acknowledgment (plagiarism). Where the submitted work is a computer program or code, I further declare that any copied code is declared in comments identifying the source at the start of the program or in a header file, that comments inline identify the start and end of the copied code, and that any modifications to code sources elsewhere are commented upon as to the nature of the modification.

Due No due date **Points** 15 **Questions** 2 **Time Limit** None

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	3 minutes	5 out of 15

Quiz results are protected for this quiz and are not visible to students.

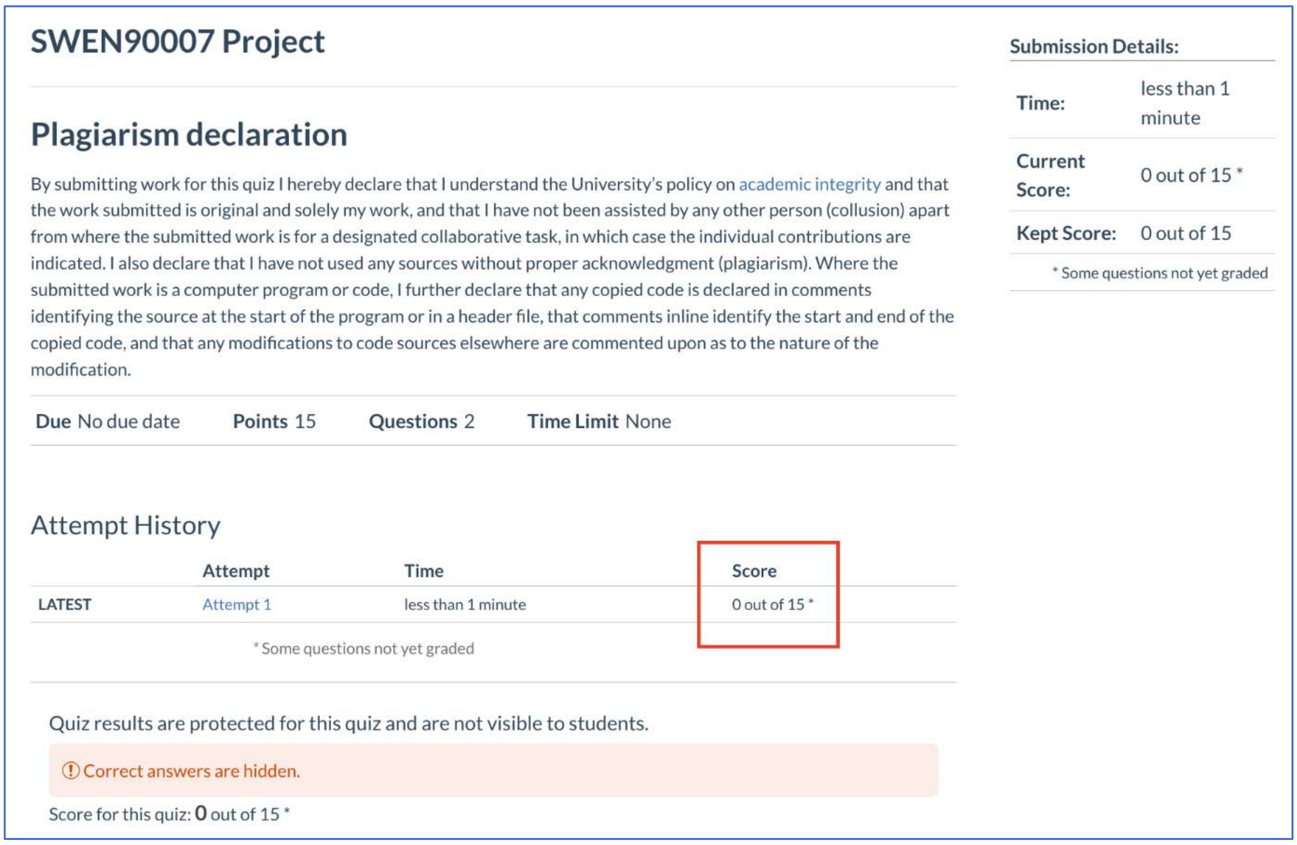
Correct answers are hidden.

Score for this quiz: **5** out of 15

Submission Details:

Time:	3 minutes
Current Score:	5 out of 15
Kept Score:	5 out of 15

Viewing exam results for an unmarked exam:



SWEN90007 Project

Plagiarism declaration

By submitting work for this quiz I hereby declare that I understand the University's policy on academic integrity and that the work submitted is original and solely my work, and that I have not been assisted by any other person (collusion) apart from where the submitted work is for a designated collaborative task, in which case the individual contributions are indicated. I also declare that I have not used any sources without proper acknowledgment (plagiarism). Where the submitted work is a computer program or code, I further declare that any copied code is declared in comments identifying the source at the start of the program or in a header file, that comments inline identify the start and end of the copied code, and that any modifications to code sources elsewhere are commented upon as to the nature of the modification.

Due No due date **Points** 15 **Questions** 2 **Time Limit** None

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	less than 1 minute	0 out of 15 *

* Some questions not yet graded

Quiz results are protected for this quiz and are not visible to students.

Correct answers are hidden.

Score for this quiz: **0** out of 15 *

Submission Details:

Time:	less than 1 minute
Current Score:	0 out of 15 *
Kept Score:	0 out of 15

* Some questions not yet graded

Appendix B – Expanded use cases

We designed this template based on real world/professional documents. You are welcome to use and to modify the given template accordingly. Please use this template as a reference to describe your use cases.

Available on Canvas: https://canvas.lms.unimelb.edu.au/courses/89935/modules/items/2167959

Appendix C – Architecture Document

We designed this template based on real world/professional documents. You are welcome to use and to modify the given template accordingly. Please use this template as a reference to describe your system (architecture design, patterns and so on).

Available on Canvas: https://canvas.lms.unimelb.edu.au/courses/89935/modules/items/2167968

Appendix D – Test Cases and Data Entry Document

We designed this template based on real world/professional documents. You are welcome to use and to modify the given template accordingly. Please use this template as a reference to describe your test cases and data entry.

Available on Canvas: https://canvas.lms.unimelb.edu.au/courses/89935/modules/items/2168338