ASSESSMENT 3 BRIEF	
Subject Code and Title	ISE102 Introduction to Software Engineering
Assessment	Audio-Visual Presentation Video of the Final Software Solution
Individual/Group	Group
Length	5 to7-minute presentation and relevant supporting elements
Learning Outcomes	The Subject Learning Outcomes demonstrated by successful completion of the task below include: a) Identify user requirements in computer-driven software applications. b) Devise software specification using a standardised modelling language based on technical and resource requirements. c) Apply software programming methodologies including object-oriented programming (OOP) paradigms to build software solutions. d) Test and validate software solutions that satisfy project requirements. e) Communicate the software solution to both technical and non-technical audiences.
Submission	Due by 11:55 pm AEST/AEDT. Wednesday of week 12
Weighting	35%
Total Marks	100 marks

Assessment Task

In this assessment, your group will demonstrate your understanding of object-oriented design principles using a standardised modelling language. In Assessment 2, your group had solved a business case study. In Assessment 3, your group will be extending the solution with added features. In addition, you will complete the object-oriented design of the business application using unified modelling language (UML). Your group will be required to demonstrate the extended programming features and object-oriented design in a 5 to 7-minute recorded audio-visual presentation video. Assessment 3 extends the work of Assessment 2.

Please refer to the Task Instructions for details on how to complete this task.

Context

A computer application solves real-world problems. However, before the application is released to the client, it is important to ensure the code works for all different types of inputs and generates correct output and terminates safely and without errors.

In Assessment 2, your group had completed coding for an application by applying object-oriented programming to solve the business case project. The programming structures included object-oriented principles, such as classes and constructors, in addition to conditionals, nested loops, functions and concepts of modular design.

In this assessment, you will demonstrate your group's understanding by describing and justifying your code choices and recording it in an audio-visual presentation that is 5- to 7- minutes in length. You will be required to explain your code by executing it. As testing and validating of the code is an important measure to ensure the code is fee of bugs and errors, you will be required to run test cases by providing valid and invalid inputs to the program.

Note: This is not a PowerPoint slide presentation; instead, this is an audio-visual recording of the demonstration of a programming product.

Refer to these instructions on how to record and submit videos from this link on Submitting Assessments. Your Learning Facilitator will also describe to you the methods /tools to record a video demonstration.

Task Instructions

The following tasks must be completed as part of this assessment. It is recommended that each group member present a different part of your group's final software solution to ensure equal participation in the audio-visual presentation.

1. Create a 5 to 7-minute audio visual presentation video of the Assessment 2 code with added features

a) Introduction: Record your group members and yourself in the video. Each group member should introduce themselves by stating their name and Torrens' student ID and show their student card (if issued) at the beginning of the recording.
b) Summarise the problem solved using the programming product.
c) Explain the code choices used to solve the case project. Note: uncomment your code before recording the audio-visual presentation video.
d) Select any two classes and explain the working and returns of the selected classes.
e) Highlight at least one or more features your group has implemented as part of Assessment 3; the extended features that were not present in the Assessment 2 submission. For example, you may have added a new function or changed your implementation using inheritance.
f) Run and record at least 5 test cases with valid and invalid inputs. For example, for an expected character or string input, what happens if an integer value is used? How does the code handle this instance, and will it terminate gracefully? (You may require enhancing your case project solution to handle such instances).

2. Object-oriented design of the business application using modelling language

In Assessment 2, your group had solved a business case problem using OOP knowledge you had learnt in Modules 5 to 8.

In Modules 9 to 12, you have learnt a modelling language (UML) for object-oriented design. In this assessment, you will use that knowledge to develop the use case diagram and class diagrams for the business case study provided as part of Assessment 2. The use case diagram needs to address the user requirements and the class diagrams must demonstrate how the OOD is applied in your software solution.

Please refer to the Assessment Rubric for the assessment criteria.

Submission Instructions

Final Submission by: **11:55pm AEST/AEDT on the last day (Sunday) of Module 12**.
Submit your audio-visual presentation video via the **Assessment 3** link in the main navigation menu in the *ISE102 Introduction to Software Engineering* Blackboard portal. Only one group member should submit the final project to the Learning Facilitator.

Note: This assessment builds on top of Assessment 2. For Assessment 2, your Learning Facilitator had provided you the base code and worked through part of it in class. It is critical that you attend or watch class streams and work along. Keep track of Blackboard announcements/emails and reach out to your Learning Facilitator with questions or issues well before the final due date.

Feedback will be provided via the Grade Centre in the LMS portal and can be viewed in My Grades. Before you submit your assessment, please ensure you have read and understand the conditions outlined in the Academic Integrity Code Handbook. If you are unsure about anything in the Handbook, please reach out to your Learning Facilitator.

Please note that ad-hoc university tech support for external apps and platforms is not available.

Academic Integrity

All students are responsible for ensuring that all work submitted is their own and is appropriately referenced and academically written according to the Academic Writing Guide. Students also need to have read and be aware of the Torrens University Australia Academic Integrity Policy and Procedure and subsequent penalties for academic misconduct. These are viewable online.

Students also must keep a copy of all submitted material and any assessment drafts.

Special Consideration

To apply for special consideration for a modification to an assessment or exam due to unexpected or extenuating circumstances, please consult the Assessment Policy for Higher Education Coursework and ELICOS and, if applicable to your circumstance, submit a completed Application for Assessment Special Consideration Form to your Learning Facilitator.

TORRENS UNIVERSITY AUSTRALIA

Assessment Rubric

Assessment Attributes	**Fail (Yet to achieve minimum standard) 0-49%**	**Pass (Functional) 50-64%**	**Credit (Proficient) 65-74%**	**Distinction (Advanced) 75-84%**	**High Distinction (Exceptional) 85-100%**
Introduction, uncommented code, problem summary **Total Percentage for this Assessment Attribute = 15%**	Demonstrates little or no attention to detail and understanding of programming. Reasons may include: • Has not provided an introduction using ID and no appearance in the video • Code is not uncommented, and it is difficult to decide if a student is reading the written explanations in comments • Problem summary is missing.	Demonstrates some attention to detail and understanding of programming. Reasons may include: • Has provided an introduction using ID but no appearance in the video • Code is uncommented but it is difficult to decide if a student is reading the written explanations in comments • Problem summary is partly explained.	Demonstrates good attention to detail and understanding of programming. Reasons may include: • Has provided an introduction using ID but appearance is not clear in the video • Code is uncommented but not explained well • Problem summary is adequately explained.	Demonstrates excellent attention to detail and understanding of programming. Reasons may include: • Has provided an introduction using ID and appearance is clear in the video for most of the time • Code is uncommented and well explained • Problem summary is well explained.	Demonstrates outstanding attention to detail and understanding of programming. Reasons may include: • Has provided an introduction using ID and appearance is clear in the video • Code is uncommented and thoroughly explained • Problem summary is comprehensively explained.
Two additional features are added **Total Percentage for this Assessment Attribute = 25%**	No additional features have been added.	One additional feature has been added but not supported with the justification.	One additional feature has been added and excellent justification for its inclusion has been provided.	Two additional features have been added but the justification lacks clarity and focus.	Two additional features have been added and excellent justification for their inclusion has been provided.

Testing of program using different scenarios **Total Percentage for this Assessment Attribute = 20%**	Five test cases are not presented using the following: • Discussion of test case scenario • Use of valid and invalid inputs • Termination of the test case gracefully.	All five test cases are presented but with some bugs using the following: • Discussion of test case scenario • Use of valid and invalid inputs • Termination of the test case gracefully.	All five test cases are presented but with minor bugs using the following: • Discussion of test case scenario • Use of valid and invalid inputs • Termination of the test case gracefully.	All five test cases are presented but with minor or no bugs using the following: • Discussion of test case scenario • Use of valid and invalid inputs • Termination of the test case gracefully.	All five test cases are presented but with no bugs using the following: • Discussion of test case scenario • Use of valid and invalid inputs • Termination of the test case gracefully.
Object-oriented design using modelling language **Total Percentage for this Assessment Attribute = 25%**	No evidence of the object-oriented design has been submitted.	Use case diagram has been developed and submitted but not supported well with the justification of its design elements.	Use case diagram has been developed and submitted and supported with good justification of its design elements.	Use case diagram and class diagram have been developed and submitted and supported with excellent justification of its design elements.	Use case diagram and class diagram have been developed and submitted and supported with outstanding justification of its design elements.

Use of collaborative programming *Effective Communication (Presentation/Oral)* **Total Percentage for this Assessment Attribute = 15%**	• Presentation is difficult to follow. • There is no logical or clear structure, with a poor flow of ideas and the arguments lack supporting evidence. • Stilted, awkward and/or oversimplified delivery. • Collaborative programming is not used.	• Presentation is sometimes difficult to follow. • Information, arguments and evidence are presented in a way that is not always clear and logical. • Correct, but often stilted or awkward delivery. • Basic collaborative programming is used.	• Presentation is easy to follow. • Information, arguments and evidence are well presented, demonstrating a mostly clear flow of ideas. • Correct, but occasionally stilted or awkward delivery. • Good collaborative programming is used.	• Information, arguments and evidence are effectively presented. Engages audience interest. • The presentation is logical, clear and well-supported by evidence. • Clear and confident delivery. • Excellent collaborative programming is used.	• Information, arguments and evidence are expertly presented. Engages and sustains audience's interest. • The presentation is logical, persuasive and well-supported by evidence, demonstrating a clear flow of ideas and arguments. • Clear, confident and persuasive delivery. • Outstanding collaborative programming is used.

The following Subject Learning Outcomes are addressed in this assessment	
SLO a)	Identify user requirements in computer-driven software applications.
SLO b)	Devise software specification using a standardised modelling language based on technical and resource requirements.
SLO c)	Apply software programming methodologies including object-oriented programming (OOP) paradigms to build software solutions.
SLO d)	Test and validate software solutions that satisfy project requirements.
SLO e)	Communicate the software solution to both technical and non-technical audiences.