

Assignment 2

Learning Outcomes

This assignment is intended to develop and assess the following unit learning outcomes:

- ✓ **LO1.** Iteratively apply object-oriented design principles to design small to medium-size software systems, using standard software engineering notations, namely UML class diagrams and UML interaction diagrams.
- ✓ **LO2.** Describe the quality of object-oriented software designs, both in terms of meeting user requirements and the effective application of object-oriented design concepts and principles.
- ✓ **LO3.** Apply object-oriented programming constructs, such as abstraction, information hiding, inheritance, and polymorphism, to implement object-oriented designs using a programming language (namely, Java).
- ✓ **LO4.** Apply effective programming strategies to refactor and debug object-oriented implementations systematically and efficiently using available programming language tools.
- ✓ **LO5.** Apply principles of software engineering practice to create object-oriented systems with peers using tools including integrated development environments (IDEs), UML drawing tools, and version control systems.

To demonstrate your ability, you will be expected to:

- read and understand UML design documentation for an existing Java system
- propose a design for additional functionality for this system
- create UML class diagrams to document your design using a UML drawing tool such as [diagrams.net](#), [UMLet](#) or [plantuml](#) – you are free to choose which one
- write a design rationale evaluating your proposed design and outlining some alternatives
- implement the features of the system that you designed
- use an integrated development environment to do so
- use git to manage your team's files and documents

The marking scheme for this assignment will reflect these expectations

Learning Materials

The base code will be automatically available in your group's repository (Gitlab).

Repeat this mantra: Design, write code, test, fix design, fix code, repeat



Note: You **must NOT follow** demo apps' design decisions; they only show how to use the engine, **NOT** how to design a proper system with object-oriented principles.

Introduction



Heavy penalty will apply for editing (add, modify, delete) any classes in the game engine as it breaks the purpose of the whole learning experience.

Before working on assignment 2, you and your team members need to agree on which design diagram, rationale and implementation will be used for the starting point for assignment 2.




You need to push the diagram, rationale and implementation to the new group repository.



IMPORTANT: You may change your design if you find that you need to. It is normal for the design to evolve alongside the codebase as the developers' understanding of the requirements improves. If your design changes, you should document the changes. This includes your design rationale as well as your diagrams.

REQ1: The moon's (hostile) fauna II: The moon strikes back

Requirement

In addition to the hostile Huntsman Spider, the moon's craters () can spawn other kind of creatures, which include Alien Bug () and Suspicious Astronaut (). Each crater can spawn an Alien Bug with a 10% chance at every turn of the game, while a Suspicious Astronaut can be spawned with a 5% chance at each game turn from each crater.



Each crater instance can only spawn one type of creature. For example, a crater at (2, 0) can only spawn a Huntsman Spider at every turn, while a crater at (9, 9) can only spawn an Alien Bug at every turn.

Please read the rest of the requirements (REQ2-4) thoroughly for the details of each new creature.

Testing Instruction

Play several turns. Pay attention to one of your craters. ~~At different turns, you should be able to see three different kinds of creatures spawn from the same Crater, randomly.~~ (EDIT: see the info callout above). Then, they should walk around the map.


REQ2: The imposter among us

Story

Going deeper into the moon, the Intern manages to scan another creature that can be found there. The scan result shows that the creature is a form of an Alien Bug named "Feature-XXX", where XXX is three random digits (e.g., "Feature-483"). It is capable of stealing resources/scraps from the Intern, resulting in the Intern having to retrieve them back.

The Intern also finds a document describing that the moon houses creatures that caused the moon to be abandoned by the factory due to how dangerous they are. They can shapeshift into an astronaut, acting as an imposter among the factory staff inhabiting the moon in the past.

Requirement

Alien Bug () can wander around the map. This creature spawns from a crater (see REQ1), so there will be many of them in the game. Unlike the hostile Huntsman Spider, the Alien Bug cannot attack the Intern. It can pick up scraps found on the ground where they are currently standing. To retrieve the scraps stolen by the Alien Bug, the Intern must attack and defeat it. When it is defeated, it will drop all scraps in its possession. This creature has 2 hit points. If, by any chance, this creature enters the Intern's spaceship, it can steal the scraps on the ground. If the Intern is within the surroundings of the bug (i.e. one exit away), it will start following the Intern. The bug will follow the Intern until it dies or the game ends.



SPOILER ALERT

► Expand




If there are multiple scraps on the ground where the bug is standing, it will randomly take a single scrap.



If the bug is following the Intern and it stands on top of a ground with items on it, the bug will prioritise picking up an item. Once it has picked up all items on the ground, it starts following the Intern again.



Although alien bugs cannot attack the Intern, it is still considered a hostile creature since it steals valuable scraps that should belong to the factory.

Similar to the hostile Huntsman Spider, the Suspicious Astronaut () is a creature that can wander around the map if the Intern is not within their surroundings (i.e. one "exit" away). However, when the Intern enters its surroundings, it will instantly kill the Intern, regardless of the Intern's health, with 100% precision. It cannot attack any other creatures hostile to the Intern. Suspicious Astronaut has 99

hit points, and it cannot enter the Intern's spaceship.



Think about what happens if the Intern's maximum health can be increased in the future.



Testing Instruction

Walk around to pick up some scraps. Find and stand next to an Alien Bug. You should be able to see an attack action with its name, such as "Feature-744" (the number doesn't need to be 744; it should be random). Do not attack it. Walk back to your spaceship, and it must follow you. Drop some scraps on the ground (more than 1). Step aside to let this creature stand on top of your scraps and pick them up. Try to walk away from that spot to let Alien Bug keep picking up scraps. It must not follow you until it finishes picking up all scraps on that ground. Once it finishes picking up all scraps, and when it stands next to you (back to follow again), attack it until it is defeated and drops all scraps.

To test Suspicious Astronaut's feature, you can walk to stand beside it. The game must end because "YOU ARE FIRED!" (it hits you with massive damage -- [bonk!](#)).

REQ3: More scraps

Requirement


The Intern finds other scraps on the moon. These include a Jar of Pickles () and a Pot of Gold (). Consuming a Jar of Pickles has a 50% chance of being past its expiry date. If it's past its expiry date, the Intern will be hurt by 1 point. Otherwise, the Intern will be healed by 1 point. The Intern can also take out the gold from the pot and put it in their wallet, increasing their wallet amount by 10 credits.



Similar to the previous assignment, you can randomly put several jars of pickles and pots of gold in the game map manually before the game runs.



Don't forget to display the Intern's current balance on the console! If you are wondering about the usage of these credits, read REQ4.

The Intern can also drink the water from the Puddle of water (). This increases the Intern's maximum health by 1 point permanently. Since the Intern forgot to bring their water bottle, they should be able to consume the water from the ground directly without having to add anything to their inventory.




The Intern can only drink water from the puddle that they are standing on.



Each puddle can be consumed infinite times.

Testing Instruction

Find and consume a Jar of Pickles. Test both cases (heal and damage). Next, find a Pot of Gold and interact with it. You should see your balance goes up by 10. Then, walk to a puddle of water () and stand on top of it. You can interact with it to drink. Once executed, your maximum hitpoints should increase by 1, but it won't increase your current hitpoints. For example, your hitpoints from the first drink will be 4/5, the second drink 4/6, and so on.

REQ4: Static factory's staff benefits

Requirement

In the spaceship, the intern can access a Computer Terminal (=) that can print items from the factory to help them explore the abandoned moon. The intern needs to pay with credits to print the items.

EDIT: The purchase fails if the intern's balance is less than what the computer terminal asks for.



Note that all of the following items can be picked up and dropped off similar to the other items so far.

The items that can be printed from the computer terminal include:

- An Energy Drink (*), which costs 10 credits each. If the intern drinks it, it heals them by 1 point. If the intern attempts to get this item from the computer terminal, there is a 20% chance that the computer terminal will ask the intern to pay double the original price. ~~The purchase fails if the intern's balance is less than what the computer terminal asks for.~~ (EDIT NOTE: moved up, applied to all items)
- A replica of the Dragon Slayer Sword (x), which is a massive sword once used by a legendary swordsman who went berserk. It costs 100 credits each. The intern can use this to attack hostile creatures, dealing 50 damage with 75% accuracy. Due to how powerful the weapon replica is, if the intern attempts to get this item from the computer terminal, there is a 50% chance that the computer terminal will throw an error, taking the intern's credits without printing out the weapon.
- A Toilet Paper Roll (s), which costs 5 credits each. Due to oversupply, if the intern attempts to get this item from the computer terminal, there is a 75% chance that the computer terminal will only ask the intern to pay 1 credit. The intern cannot do anything with a toilet paper roll.



You can assume that the factory has an unlimited supply of items that they can sell to the intern.

Testing Instruction

You can start the game with a massive amount of credits in your wallet. Since you have placed a Computer Terminal, walk and stand on or next to it. You should be able to see three actions to purchase each item (i.e., three actions). Make some purchases and test each random scenario of the item purchases (i.e., overcharged or undercharged).

Submission Instructions



Within this class, you are welcome to use foundation models (ChatGPT, GPT, DALL-E, Stable Diffusion, Midjourney, GitHub Copilot, and anything after) in a totally unrestricted fashion, for any purpose, at no penalty. However, you should note that all large language models still have a tendency to make up incorrect facts and fake citations, code generation models have a tendency to produce inaccurate outputs, and image generation models can occasionally come up with highly offensive products. You will be responsible for any inaccurate, biased, offensive, or otherwise unethical content you submit regardless of whether it originally comes from you or a foundation model. If you use a foundation model, its contribution must be acknowledged in the submission (in the form of a separate file titled `FoundationModelsUsed.txt`); you will be penalised for using a foundation model without acknowledgement.

Having said all these disclaimers, the university's policy on plagiarism still applies to any uncited or improperly cited use of work by other human beings, or submission of work by other human beings as your own. Moreover, missing contribution logs/GIT commits and/or failing any handover interview will be treated as a potential breach of academic integrity which will be further investigated.



We will mark your assignment based on the latest commit in the `main` branch in your GitLab repository by the due date, **not** in the `master` branch.



Heavy penalty will apply for editing (add, modify, delete) any classes in the game engine as it breaks the purpose of the whole learning.



A Reminder:

Before working on assignment 2, you and your team members need to agree on which design diagram, rationale and implementation will be used for the starting point for assignment 2.

You need to push the diagram, rationale and implementation to the new group repository.

You are expected to produce the following artefacts in Assignment 2:

1. Implementation (the code) of the system.
2. Contribution Log (mandatory, useful for markers) - see the "Assignment Rules" module, specifically the "Contribution Logs Template" section
3. Revised (based on feedback received on Assignment 1 or if you consider changes were needed) + New UML design documents, which include
 - 4 UML class diagrams (1 for each A2 requirement, similar to Assignment 1)
 - 4 Interaction diagrams (1 for each A2 requirement: sequence diagrams or communication diagrams are equally acceptable)
4. Design rationale, which should include (4 of them, 1 for each requirement)
 - Justification for any changes made to the design proposed for Assignment 1
 - Justification for the new requirements



For class diagrams, please make sure to distinguish between existing classes and new classes with different colours. Additionally, you should colour revised classes and newly added classes differently.

Additionally, you will be asked to complete the following forms:

- Peer Assessment Form (mandatory, useful for markers and for you to reflect on your group dynamics)
- ~~Feedback Request Form (to be completed in order to upload your files to Moodle).~~ [NOT REQUIRED ANYMORE 06/05/2024]

As mentioned above, you can (and should) **update** the artefacts previously marked in Assignment 1 since the **code must align with the UML diagrams**.

We will mark your Assignment 2 on the state of the `main` branch of your Monash GitLab repository by the due date at your local time (see above). If you've done any work in other branches, make sure you merge it into main before the due time.



The FIT2099 Assignment Rules apply to this assignment. Please read this document and make sure you comply, especially as regards the Work Breakdown Agreement.

You do not need to create a new copy of your work for Assignment 2. Git lets us easily roll back to any previous version of your repository, so keeping a separate "Assignment 1" folder only creates unnecessary clutter. If you like, you can [add a tag](#) to your final Assignment 1 commit before starting on Assignment 2, so you can find that version easily.

You may update your design if you find that you need to. We expect consistency between your design documents and implementation. Regarding the contributions document, please use the prepared tab to separate the contribution from the previous assignment. We will take your contribution logs into account when marking.

It is highly recommended that you explicitly Indicate in the **design rationale** what changed in the design between Assignment 1 and Assignment 2. Explaining the changes would help you organise your ideas to explain how the design changed during your handover interview. A quick summary of changes with some explanation regarding why you applied such changes would be really valuable.

Contribution Log

We require you to have a document to know how you plan to divide the work between team members and simultaneously capture individual contributions. We have prepared a template you can copy to your university's Google Drive and share with your team. Please follow the detailed instructions in the "Assignment Rules" module.



Since your markers need to access this information, make sure that you create a share link and put it in the README of your repository. Failure to do so will result in 0 mark.

In this matter, everyone must contribute a **FAIR amount of code AND documentation**. It means you cannot work only on the code or only on documentation. In other words, your team **MUST NOT** split the work by artefacts: a person working on a "class diagram", another on a diagram alone, and another working on design rationale alone. We will give a heavy penalty to your team as a whole.

You will meet up with your team ideally once a week for about 1 hour for team discussion and another 1 or 2 hours for integration and resolving conflicts (if any).

Peer Assessment Form

You should also complete the peer assessment form, which you can find here:

<https://forms.gle/BETvSkCoPCx4briHA>, to help us understand and evaluate your (and other team members') contributions to the assignment better.

About Late Submissions

It is all team members' responsibility to ensure that the correct versions of the documentation and code are present in the repository by the due date and time.

Marking Rubric

Assignment 2 Rubric

Prerequisite for marking the whole assignment: Design documents (UML diagrams and design rationale) and implementation need to be submitted for the assignment to go through the marking process. Missing any of these will result in 0 marks.



Please, kindly note that the **handover presentation/interview (see details below)** needs to be completed to get the marks awarded.

Overview:

Basic requirements are worth 100% of the marks (REQs1-4)

Total: 26 points

- Feature implementation completeness (6 points)
- Implementation quality: design principles (8 points)
- Design rationale (4 marks)
- Integration with the existing system (2 points)
- UML syntax and clarity (2 points)
- Alignment and design consistency (2 points)
- Style & Javadoc (1 point)
- Format and directory structure (1 point)

Detailed rubric:

Feature implementation completeness (6 points)

6 marks - The system runs and perfectly meets **all functional expectations** as per the relevant requirement(s) with no runtime errors. All required classes and relationships are implemented, and the program's behaviour aligns perfectly with the specification.

5 marks - The system runs and meets **all functional expectations (with some minor errors or punctual omissions)** as per the relevant requirement(s). All necessary classes and relationships are included, and the program's behaviour largely matches the specification, except for **one or two minor unexpected behaviours**. No runtime errors occur.

4 marks - The system runs and **partially meets all functional expectations, displaying several**

minor functional errors or omissions as per the relevant requirement(s). Despite this, the majority of necessary classes and relationships are included, and **only a few minor unexpected behaviours occur**. No runtime errors are present.

3 marks - The system runs and **all functional expectations were addressed to some extent**, (with one or two major functional errors) as per the relevant requirement(s). Most important classes and relationships are included but **at least one or two major unexpected behaviors** are observed. A major unexpected behaviour is that which affects considerably the completeness of at least one requirement. No runtime errors occur.

2 marks - The system runs but **several functional expectations were not addressed** as per the relevant requirement(s) (e.g. the notion of Behaviour is not included even though it is a part of the requirement). Alternatively, runtime errors occur during the execution of the system but they can be easily fixed.

1 mark - The system runs but **addresses only some functional expectations and shows major omissions** as per the relevant requirement(s) (important classes or relationships are missing). Alternatively, runtime errors occur during the execution of the system without a clear idea of the cause or cannot be easily fixed.

0 marks - The system runs but it **poorly addresses or doesn't address the functional expectations** as per the relevant requirement(s). Alternatively, the system might not run at all.

Implementation quality: design principles (8 points)

This item applies across all functional expectations as per the relevant requirement(s)

8 marks - The implementation of all functional expectations as per the relevant requirement(s) **flawlessly adheres to good design principles and concepts** (e.g., DRY, SOLID principles and connascence), making the design easy to extend and maintain. Any punctual violations are **convincingly** justified in the design rationale (e.g., using singleton to implement a feature). All relevant requirements have been addressed.

7 marks - The implementation follows good design principles **nearly perfectly** making the design is easy to extend and maintain (if some punctual principles are violated, the trade-off is **somewhat** justified in the design rationale. All relevant requirements have been addressed.

6 marks - The implementation involves **one or two minor violations of design principles** (could be easily fixed) across all functional expectations as per the relevant requirement(s) (e.g., some attributes are not set to private without any justification). All relevant requirements have been addressed.

5 marks - The implementation involves **minor violations of design principles in multiple places** (could still be easily fixed) across all functional expectations as per the relevant requirement(s). All relevant requirements have been addressed.

4 marks - The implementation involves **one or two non-severe violations of design principles that could be implemented in a better way** (no trade-offs are convincingly provided in the design rationale) across all functional expectations as per the relevant requirement(s) (e.g. some code repetitions are found in the implementation that would require some refactoring to be fixed). All relevant requirements have been addressed.

3 marks - The implementation involves **non-severe violations of design principles in multiple places** (no trade-offs are convincingly provided in the design rationale). All relevant requirements have been addressed.

2 marks - The implementation involves **one or two severe violations** of design principles that could be implemented in a better way across all functional expectations as per the relevant requirement(s), e.g. violating the SOLID principles. Fixing them would require substantial refactoring. Alternatively, not all relevant requirements have been attempted.

1 mark - The design/implementation can be considered **hacky or various instances of procedural programming are found**. For example, the implementation uses downcasting and 'instanceof' in various cases without a convincing justification. Alternatively, it can also be the case that abstraction is not used, making the design **difficult to extend and maintain**. Alternatively, only some relevant requirements have been attempted.

0 marks - The implementation mainly follows a non-OO paradigm; or the UML class diagram(s) or the implementation is missing. Alternatively, most relevant requirements have not been attempted.

Design rationale (4 points)

4 marks - The design rationale includes a description of **what has been done and why**, focusing on the principles and concepts taught in the unit such as DRY, SOLID principles, connascence etc., and not in terms of game design. The rationale discusses the **advantages and disadvantages** of the design (pros and cons), **the reasons** for choosing the current design, and **ways in which it can be easily extended** (e.g. my design achieves OCP because if a new character is added in the future ...). All relevant requirements have been addressed.

3 marks - The rationale describes **what has been done and why**, based on the principles and concepts taught in the unit. It also includes **some discussion** of the pros and cons of the design and the reasons for the current design choice but **lacks examples of future extensibility**. All relevant requirements have been addressed.

2 marks - The rationale describes **what has been done and why**, aligning with the principles and concepts taught in the unit. However, it **lacks a discussion on the pros and cons of the design and/or examples of future extensibility**. All relevant requirements have been addressed.

1 mark - The rationale **primarily describes what has been done** without a thorough explanation of the decision-making reasons. For instance, it may mention design principles and concepts without providing convincing explanations, discussion of pros and cons, or examples of how the system can

be extended. Alternatively, not all relevant requirements have been attempted.

0 marks - The design rationale is either very challenging to read, is missing, or the submitted work omits more than one relevant requirement.

Integration with the existing system (2 points)

This item applies across all relevant requirements.

2 marks - The implementation **effectively uses the engine classes** (e.g. the submitted work demonstrates that the students understand the difference between actions and behaviours). All relevant requirements have been addressed.

1 mark - The implementation **does not use some engine classes as intended** (e.g. behaviours are created for some actions that do not need it, such as actions performed by the player) or includes **custom classes for functionality that could be implemented with the engine class** (e.g. created a custom ground class instead of using the ground class given in the engine package). Most relevant requirements have been addressed.

0 marks - The engine has been modified in a minor or significant way OR the UML class diagram OR the implementation is missing

UML syntax and clarity (2 points)

This item applies across all relevant requirements.

2 marks - Relevant (static and/or dynamic) diagrams are **perfect in terms of syntax**, with no missing multiplicities, correct arrowheads for all relationships, and appropriate usage of realisation for classes implementing interfaces, generalisation for classes or interfaces inheriting others, etc. Diagram formatting is **consistent and clear**. All requested diagrams have been submitted.

1 mark - Diagram(s) contain **some minor syntax errors**, such as missing multiplicities for several associations, inappropriate use of generalisation for classes implementing interfaces, realisation for classes extending others, or classes extending multiple classes, etc. Nonetheless, the design can still be understood by the TA with a little effort. Alternatively, there are several **inconsistencies across diagrams**. All necessary diagrams have been submitted.

0 marks - Diagram(s) are significantly hard to understand or show major inconsistencies in formatting and clarity, OR more than one required diagram is missing, OR they do not resemble a UML diagram as required.

Alignment and design consistency (2 points) - incl. Design rationale, code and UML diagrams

2 marks - The design rationale and UML diagrams are in **perfect alignment** with each other and

with the code implementation across all functional expectations, as per the relevant requirement(s). All relevant requirements have been addressed or attempted.

1 mark - There are **some small inconsistencies** (that can be easily fixed) between the design documents and the implementation, but all relevant requirements have been addressed or attempted.

0 marks - There are **major inconsistencies** (which would necessitate significant changes for correction) or **numerous smaller discrepancies** distributed throughout the design documents and the implementation. Alternatively, more than one required diagram or implemented requirement is missing, or some submitted design documents do not resemble the required UML diagram format.

Style & Javadoc (1 point)

Applicable to basic requirements (1-4) only

1 mark - The code is properly documented with Javadoc across all functional expectations as per the relevant requirement(s), including class-level and method-level documentation. The Google Java Style guide is followed properly (e.g. package names are written in lowercase, attributes and variables names are written in lowerCamelCase, class names are written in UpperCamelCase, etc.). All relevant requirements have been addressed or attempted.

0.5 marks - Some classes and methods are missing Javadoc documentation OR some classes do not follow the Google Java style guide

0 marks - Most classes and methods are missing Javadoc documentation OR The Google Java style guide is not followed.

Project formatting and directory structure (1 point)

Applicable to basic requirements (1-4) only

1 mark - If the directory structure suggested in the specification has been followed. All the documents are easily found where expected.

0 marks - The suggested directory structure was not followed and some files may be hard to find.

Individual Contribution (100%) & Handover Presentation/Interview (compulsory)

We will, in principle, assume all team members equally contributed to the assignment. Yet, this individual contribution component will be used to weigh the marks awarded at a group level (see above) for cases where individual students did not contribute to the group task fairly (see details below) or did not show evidence that they understand their own work during the handover

presentation/interview (see details under Handover Presentation/Interview, below).

If you contributed to the group task and demonstrate that you understand your own work you should get (100%) the full marks awarded at a group level as per the rubric above.

Individual Contribution (100%) Only for A2 and A3

100-80% - The student has made substantial contributions to the submitted work, evident by numerous **commits** (over 10) with meaningful commit comments (note: default comments from web UI are not considered). For full marks, there should be ~~at least one commit per week leading up to the submission deadline, and~~ **(Update: 9/05/24) multiple contribution log entries** (over 10 entries) with comprehensible explanations for an external reader (e.g. the TA).

Additionally, the student's contributions reflect a reasonable understanding of OOP principles, whether in programming or designing UML diagrams. The student's input spans **both design and code**.

Additionally, other team members have given positive feedback or there are no complaints regarding the student's contributions (as **per peer assessment**).

70-40% - The student has made some contributions to the submitted work, evident by **some commits** (between 5 and 9) with meaningful commit comments ~~OR the student made more than 10 commits but~~ **all of them on the same week of the submission deadline (Update: 9/05/24)** or there may be **commits without any comments**.

Alternatively, there are **some contribution log entries** (between 5 and 9) with comprehensible explanations for an external reader (e.g. the TA).

Alternatively, the student's input focused **only on the design or the implementation**.

Alternatively, there are **some complaints** regarding the student's contributions (as per peer assessment).

30-20% - Peer assessment, contributions logs, and git logs show there are **insufficient contributions from an individual**; the student received **a negative review** from other team members which usually happened due to last-minute work, difficult communication outside of labs, or any other kind of difficulties (evidence are collective from peer assessment, emails, contribution logs, and git logs). The student committed to git, with **at least four commits** and meaningful comments OR there are various commits that are not meaningful and that do not explain the purpose. **Some contribution log entries** (at least 4 entries) with meaningful explanations that can be understandable by a new team member (e.g. the TA).

10-0%% - There is no or little evidence of contribution to the team from Git logs or contribution logs. If applicable, the student only writes comments (inline or Javadoc) without any appropriate programming contribution. Clear evidence of academic misconduct of an individual would be marked

as zero. Note: submitting files as attachments to the release notes would be marked as zero.



Make sure you communicate effectively within your team. If a student works on or completes a task specifically assigned to another individual without formal approval or reassignment, this will result in a penalty. It is essential to respect assigned responsibilities and communicate with your team members before taking over someone else's task. This rule is intended to encourage proper communication and respect for the collaborative process.

Handover Presentation/Interview (compulsory)

The aim of the handover group presentation is to demonstrate the team's understanding and implementation of the four requirements assigned to your project. Each team member will be responsible for presenting one requirement, including its UML diagram and implementation details. The presentation will be followed by a Q&A session, where peers and the teaching assistant can ask questions related to the project.

Duration: The total duration of the presentation will be 30 minutes, divided as follows:

- Presentation: 20 minutes (5 minutes per requirement)
- Q&A Session: 10 minutes

Presentation Requirements:

- **Introduction:** Start with a brief overview of the project and its objectives.
- **Requirement Presentation:** Each team member will present one requirement. The presentation should include:
 - A brief description of the requirement. This includes running the application and showcase its correct functionality.
 - The UML diagram that represents the design of the requirement.
 - A walk-through of the implementation, highlighting key code snippets and functionalities.
 - Any challenges faced and how they were overcome.
- **Integration:** Briefly explain how the individual requirements integrate to form a cohesive system.
- **Conclusion:** Summarize the key points and state the readiness of the system for handover.

Q&A Session: After the presentation, the team will engage in a 10-minute Q&A session. Peers and the teaching assistant can ask questions related to the project. The team should be prepared to answer questions on both the technical and conceptual aspects of the project.

Presentation COMPLETED: Satisfactorily completing the presentation, including effectively addressing questions during the Q&A session, will be equivalent to passing the handover interview.

Presentation NOT COMPLETED: A student may be unable to join the team presentation due to special considerations. Alternatively, if the teaching assistant (TA) leading the session determines that an individual student's presentation or their responses during the Q&A session do not provide

sufficient evidence of understanding the requirement they presented, that student will be asked to undergo a regular interview. The TA will organise this with the student.



For the cases where the presentation cannot be completed during the applied session (for example, due to a special consideration), an individual handover interview will be organised on subsequent days after the applied session. The completion criteria for this interview will be the same as in A1, as follows:

COMPLETED: If the student can satisfactorily answer all (or at least 2) questions during the handover interview. The responses need to demonstrate that the student understands the various parts of the assignment, including both parts created by the student (as per the logged commits) and at least some awareness of those parts created by other team members (for Assignments A2 and A3).

NOT COMPLETED: If two or more questions are not responded to adequately and sensibly. The remaining question(s) are partly responded to, but it is unclear whether the student understands their own work.



IMPORTANT: Failing to have meaningful commits (i.e. showing that the task was progressively completed) and/or failing the handover interview would automatically flag this as a potential case of plagiarism, it will be further investigated using a similarity check software, and **zero marks would be awarded**.



IMPORTANT: Any inquiry (e.g. potential conflict within a team) should be submitted via the emails below (not your AdminTA, Lecturer nor CE). Emails sent in other ways will not be processed in time.

FIT2099.Clayton-x@monash.edu if you are based in Clayton
FIT2099.Malaysia-x@monash.edu if you are based in Malaysia

Assignment 2 Q&A Session

Direct link: <https://monash.au.panopto.com/Panopto/Pages/Viewer.aspx?id=db2d1b82-a0c4-4c84-b76f-b164008557d6>

Meeting chat:



[GMT20240502-070951_RecordingnewChat.txt](#)