COMP3850 Computing Industry Project

Read this document carefully and by the due date/time. All team members should be familiar with and refer to this Project Deliverables Definition Document. The team Quality Manager should ensure the team follow the instructions correctly.

As part of COMP3850 you will be formed into groups of 4-6 students. You will work together to understand what your client requires and go through a number of phases from assessing the feasibility, requirements, design, implementation and testing of a prototype or other artefact with associated documentation, as appropriate for your project. You can expect requirements and client needs to evolve over the period of the project.

The project is worth 75% of the course assessment.
The other assessment for the course is the **final exam** worth 25%.

The final marks for Deliverables 1-5 may differ for each member of a group if a certain student does not contribute equally to the group's deliverables according to the contributions submitted by team members. At the two checkpoints identified in the deliverables schedule, each individual must fill in an online individual contribution form to state the percentage contribution of each team member, including themselves, to the deliverables included in that contribution form (see Deliverables Schedule).

The project will follow an agile life cycle and is made up of a number of deliverables that will be iteratively updated and expanded.

Contents

Summary of Group / Individual Assessment Expectations

- **1) List of assessment items**
- **2) List of individual contribution form submission checkpoints**
- **3) List of communication and involvement expectations**
- **4) Semester 1 2023 – Deliverables Schedule**

Deliverable 1 Details

- **Deliverable 1 marking scheme**

Deliverable 2 Details

Project Plan including Quality Manual
Marking scheme for project plan and quality manual sections
Requirements Document/ Scoping Document
A note to all groups
Details for CYBERSECURITY projects
Details for DATA SCIENCE projects
Details for GAME projects
Details for SOFTWARE projects
Marking scheme for scoping / requirements sections
Deliverables 3 & 4 (Increments 1 and 2)
All Teams: Deliverable 3 & 4 submission instructions
Common sections for all groups
Updated Project Plan and Quality Manual
Updated Requirements / Scoping Document (0.25 marks Del 3 and 0.1 marks for Del 4)
List of Assumptions
Prototype / MVP
Prototype Feedback Adjustment
Analysis, Design + Testing Documentation for DATA SCIENCE projects
Analysis, Design + Testing Documentation for CYBERSECURITY projects
Analysis, Design + Testing Documentation for GAMES projects
Analysis, Design + Testing Documentation for SOFTWARE / WEBSITE projects
All Teams: Deliverable 3 and 4 marking schemes
Deliverable 5 Details
Deliverable 6 Details
Deliverable 7 Details

Summary of Group / Individual Assessment Expectations

1) List of assessment items

(D1 to D5 must include a deliverables certificate starting from the first page)

Deliverable 1 (D1):
Feasibility Study

Deliverable 2 (D2):
Project Plan + requirements/scoping document

Deliverable 3 (D3):
Updated D2 documents + design documentation + test documentation + Prototype/MVP

Deliverable 4 (D4):
Updated D2 documents + updated D3 documents + user / training manual

Deliverable 5 (D5):
Final Group Reflective Report

Deliverable 6 (D6):
Project presentation and demonstrations

Deliverable 7 (D7):
Delivery of product to sponsor

Deliverable 8 (D8):
Final exam

2) List of individual contribution form submission checkpoints

(marks may be deducted if the following are not completed)

Individual Contribution form 1: *covers deliverable D1 and D2*

Individual Contribution form 2*: covers deliverable D3, D4, and D5*

3) List of communication and involvement expectations

(marks may be deducted if the following are not completed)

Induction in week 1 (C1)

Team training in week 2 (C2)

Weekly reports on the group forum (C3)

Meetings Register (for off campus meetings ONLY) (C4)

Deliverables Certificate and Time/worksheets (C5)

Essential Sponsor Communications (C6)

4) Semester 1 2023 – Schedule

Wk	Deliverable	Product	Marks	Due - Date and Time
2	**C2**	Attend and participate in Week 2 team training	**1**	**5-7pm, Thursday 02/03/2023**
3	**D1**	Feasibility Study*	7	Thursday 09/03/23
6	**D2**	Project Plan and Requirements/Scoping Document*	13	Thursday 30/03/23
6	**Individual Contribution Form 1 For (D1 + D2)**	**Checkpoint to complete first half Individual Contribution form on iLearn		Saturday 1/04/23
8	**D3 (Inc1)**	Updated Deliverable 2 documents, plus Design, Test Cases, Prototype/MVP*	13	Thursday 27/04/23
11	**D4 (Inc2)**	Updated Increment 1 deliverables, plus user/training manual *	13	Thursday 18/05/23
13	**D5**	Final Group Reflective Report*	8	Thursday 1/06/23
13	**D6**	Project Presentation / Demonstration	10	Thursday 1/06/23
13	**Individual Contribution Form 1 For (D3 + D4 + D5)**	**Checkpoint to complete second half Individual Contribution form on iLearn		Saturday 3/06/23

13+	**D7**	Final Delivery of the Product to the sponsor. This should occur by Week 13 but can be extended to end of exam period or Team grade will be (I) Incomplete. Upload of all project documentation (including other files are optional)	10 (from sponsor)	Thursday 1st June up to Thursday of Week 16, 22nd June.
14+	**D8**	Final Exam	25	Week 14/15 Time/ location TBA
*For deliverables 1-5: **One** team member in the group should **submit a single pdf named 'Group[num]_Del[num].pdf'** that includes the Deliverable Certificate (which includes timesheets) at the start of the document. Place content in the order used in this document. For Deliverable 7, on iLearn upload a zip file that includes the latest version of all documents created over the semester.				

Additional Activities – penalty may apply for non-compliance		
C1 Induction	Attend and participate in Week 1 to learn about unit structures and meet team and sponsor.	Week 1 lecture/ class
C3 Weekly reports	Start a new thread for each week with Group # Week# in the thread title. One per team. Stating – weekly activities, progress, any meetings (date, time, location, attendees), issues, work to be done in coming week by each individual. Marks for the group may be deducted from the overall grade for < 8 satisfactory weekly reports.	Via iLearn Forum – post to team only

C4 Meetings Register	Each time you meet off campus with your client, before the meeting record the upcoming event. Create/Add a post to Meeting Register thread in forum with the following information: Date, time, location, 2 phone numbers, names of all students attending, sponsor name	Via iLearn Forum – post to team only
C5 *Deliverables Certificate and Time/work sheets	Each student from each group will have to sign the front page of the deliverables certificate. One person from each team will upload the deliverables submission to iLearn by the due date/time. On the reverse side of the deliverables certificate include time/worksheet · Group by student, · Task · Duration (Hours) · Complexity (high-medium-low) · ADD a total row for each individual that shows their TOTAL and grand total that shows the GRAND TOTAL of hours for the whole team. Include ALL tasks/activities (e.g. meeting attendance, research/reading/training, etc) not just those spent on the deliverable. Marks for a submission (i.e. deliverable) may be deducted for missing/incomplete/incredible and far-fetched timesheets	Template on iLearn under Project in [Deliverables Certificate/Cover sheet]
All email correspondence MUST be via the MQ student account. other formats **may** be ignored. Please **include group number** in all correspondence.		

C6 ESSENTIAL SPONSOR COMMUNICATION

In order to deliver valuable outputs within the semester, you are expected to follow agile processes including regular and frequent communication with your sponsor to obtain their feedback on your ideas and work. Your sponsor is not responsible for editing or assessing your documents. Make sure what is presented to them is professional and businesslike.

Deliverable 1: Discuss the Feasibility Report with your sponsor to ensure any misunderstandings are clarified, particularly concerning **their** problem, opportunity, mandates and assumptions. **After** updating based on your improved understanding and feedback from marking, discuss with the sponsor what parts of the Feasibility Report they are interested in and provide them with either an Executive Summary or the full report, as described in the Deliverable 1 section.

Deliverable 2: **Before** Deliverable 2 due date, you must obtain sponsor feedback to the Requirements/scoping document. This feedback should be included in the submission.

Deliverables 3 and 4: **Before** Increment 1 and 2 due dates, you should have demonstrated your prototype/minimum viable product (MVP) to your sponsor. In the Prototype section, your submission should include when/how the demonstration was done, the feedback you received and your response to it. You will lose marks for your prototype/MVP if you do not include information about the prototype in your Increment 1 and 2 documentation and regarding the meeting with your client, the feedback you received from them and how you will respond.

Deliverable 1 Details

Deliverables Certificate
Feasibility Report (4.5 marks) approx. 8-15 pages
Team Manual (2.5 marks) approx. 3-6 pages

Please liaise with your industry sponsor about their needs, problem or opportunity description and the solution sought. There may have been supporting documentation you received. The team should do further research to understand better the organisation, domain, problem, task, solution options, etc. Based on the information given and your research, discuss the problem, the activity/task, the domain and issues with your team so that you get a bigger picture of the context the problem fits in. Particularly consider the IT needs related to that activity and develop an initial proposed IT solution to one or more of the problems. You will need to describe the problem, alternative solutions (even if a specific solution has already been requested) and justify the solution you recommend.

Use the template on iLearn under resources for your Feasibility Report. In your Problem Identification section, explicitly identify the nature of your project: software, game, data science or cybersecurity. Please explain why it is that type of project. In the rare case that a project might cover more than one category, please provide a longer explanation. Assumptions –It is important to include assumptions. If those assumptions are wrong, then your plans and project will be compromised. Assumptions could include access to certain resources, scope of the system, support that will be provided, platform that will be used, etc. It is good to elaborate these so the sponsor can see them and correct any false assumptions.

After you receive your feedback, update your submission and discuss with the sponsor what sections of the Feasibility Report are of interest to them. As a minimum you need to provide them with a 1-page Executive Summary stating the sponsor's problem and opportunity.

Use the headings in the rubric below for your Team Manual. Discuss team structure and roles. Review together the ACS Code of Professional Conduct (see iLearn). Describe the team's values and how you will adhere to the Code. Clarify and justify the approaches and/or tools that will be/are being used for project management, team communication and to manage change or conflict.

Deliverable 1 marking scheme

Team _____

Problem Identification & Project Classification	0.5
Opportunities	0.5
Mandates	0.1
Current Situation	0.25
Tangible & Intanbile Benefits	0.5
Alternative Solutions	1.75
Recommended Solution	0.75
References/Abbreviations/Assumptions	0.15
TOTAL	**4.5**
Team Manual	
Team Organisation and Structure	0.5
Team Values & ACS Code of Professional Conduct	0.5
Project Management Approach and Tools	0.75
Communication Plan and Meeting Schedule	0.5
Change Management and Conflict Resolution	0.25
TOTAL	**2.5**

Deliverable 2 Details

Deliverables Certificate

Updated Team Manual (0.75 mark)
Project Plan (6.25 marks)
Requirements Document / Scoping Document (6 marks)

Project Plan

The Project Plan should not exceed (15) A4 sides, excluding appendices. See the marking rubric for content to be included and follow these headings and order.

Include the following in the *Project Plan*, in this order:
Statement of purpose [for plan] and scope [of project]
Risk Management:

- Development risks and their management, including a risk matrix/table that includes the risk, description, probability/likelihood, effect/consequence and risk level. Mitigation strategies must also be included.

Resource Management

- Project resources - people, hardware, software, other resources

Change Management

- Changes to requirements and scope
- Change to documents, code and data (i.e.version control).

Quality Management

- Discuss how quality will be monitored and assured

Schedule

- Using the project management tool of your choice (e.g. Trello, MS Project[1], ClickUp) include chart/screens that clearly show planned tasks, deliverables, time-line, resource/task allocations for the entire project.

Assumptions

- Assumptions around resources, availabilities, tools, techniques, standards, communication, expectations and anything else your project is based on and assumed to be true.

NOTE: For software development and games projects, we are using an agile software development life cycle (SDLC) because the requirements are often not fully known and will emerge as the project progresses based on the client's feedback. Data science projects should follow the intent of an agile data science process (http://www.datascience-pm.com/what-is-agile-data-science/).

Agile approaches are widely used in industry because agile methods allow all stakeholders to communicate and validate their ideas sooner. Examples of agile methodologies include SCRUM and Kanban. You will want to research what agile methodology best suits your project. Your project plan must be consistent with the deadlines and process life cycle you are using.

Marking scheme for project plan

Updated Team Manual	**0.75**
Team Organisation and Structure	0.25
Team Values & ACS Code of Professional Conduct	0.25
Communication Plan and Meeting Schedule	0.15
Conflict Resolution/Negotiation	0.1
Project Plan	
Statement of Purpose/Scope of Project/Description	0.5
Risk Management	1
Resource Management	0.75
Change Management	
Managing requirements and scope change	0.5
Version Control	0.5
Quality Management	0.75
Project Schedule	
Tasks/Deliverables/Process	1
Timeline	0.5
Resources Allocated	0.25
Assumptions	0.25

Standards/Templates/Appendices/Forms	0.25
Plan TOTAL	**6.25**

***** Continue on with the following pages for the Requirements / Scoping Document specifications and rubrics for this deliverable.***

Requirements Document/ Scoping Document

(6-20 pages)

A note to all groups

Overview: you must provide the requirements/scoping document to your client before the due date for their feedback. I will assume that what I receive has been confirmed by the client. You can expect to make changes to your document based on the client's review. It is best, though not essential, to send your draft document to the clients AND also arrange to meet them to obtain face-to-face comments. Your client may also like to see your project plan. Ask them. Make sure you clarify and confirm with them any resources you identified in the project plan that you expect from the client.

Include at the END of your Requirements/Scoping Document a page that describes (1 mark loss if not done):

Meeting date and time:
Feedback received
Team response/action points

Choose which of the following rubric sections to follow for your scoping document / requirements document:

- Cybersecurity
- Data Science
- Game Development
- Software (Default if unsure / not specified otherwise)

Details for CYBERSECURITY projects

Cybersecurity majors should have gained experience with a range of methods, tools and documentation in your previous units. Your client is also likely to have examples and preferred methods, tools and documentation to be used in your project. Please discuss these with your client and describe them in the section on "Deliverables Due" in your project plan. You will need to extend the project plan to a more comprehensive scoping document. Please refer to the marking rubric for content/sections in your scoping document.

The project plan put together the timeline and expected deliverables, now it is time to focus on what your project is about. This scoping document is intended as a way of identifying your analysis of the problem space.

The scoping document for Cybersecurity projects is aimed at outlining / communicating the following things for your team, the staff in The PACE unit, and your sponsors / stakeholders:

Part 1: analysis overview (intent / motivating factors)

1. The purpose and scope of activities being considered
    - What systems are being investigated / profiled / attached / guarded and why?
    - What kind of project are you undertaking?
        - An attacking team? (red team)
        - A defensive team? (blue team)
        - An overall security assessment analysis with recommendations?
        - …
2. The understanding (high level overview) of attacks and vulnerabilities that are going to be investigated
    - What attack vectors are being considered / reviewed / researched / attempted?
3. The types of adversaries that will be considered / profiled as part of any defensive or offensive strategy the team will be undertaking
    - This is akin to grouping adversary profiles into different groupings (e.g. what are the different kinds of bad actors and what would their goals be?). This can help to consider how to guard against certain types of attackers and preventing their intended outcomes.
    - What resources would each of these different adversary groups have at their disposal to achieve their outcomes?
4. The detection methods that are being considered / are available for investigation. You need to understand the important role of threat detection in any security system/solution. Even if your project is a simple application/plugin, if you don't pay attention to detection, then if an incident happens, you will have no clue how it happened. And even worse, your system/solution has no embedded mechanism to

respond to threats before they become incidents. If this has not been part of your considerations in the final product, it is highly recommended to include Threat Detection function. For instance, in providing the proposed end-to-end encryption and storage solution, is there any chance that someone would be able to intercept or tamper the data transmission? If so, how would your solution respond to that? What do you do to prevent these kind of threats from happening again?

5. Monitoring of assumptions. Description of how the identified assumptions will be monitored in your project to ensure they have not been invalidated.
6. Document conventions being followed in this document (and what each of the sections cover – this is useful for anyone who doesn't know what to expect in this document)

Part 2: analysis activity details / specifics (mechanisms for investigation being considered)

1. Security feature extraction
    - This should expand on the points in (Part 1: 1). and (Part 1: 2).
    - What security features / weaknesses are initially being considered?

Part 2.1 focuses on your solution, i.e. what security features you will provide in the final product.

2. Attack / vulnerability analysis
    - This should expand on the points outlines in (Part 1: 2).
    - In Part 1, you outline attacks / vulnerabilities being investigated
    - Here in Part 2, is a chance to answer questions like
        - "What does this look like in the systems we are focusing on"?
        - Are there any indicators that might be a sign that this kind of attack / intrusion is currently occurring?
        - Are there any impacts / results which can show that such an attack / intrusion has occurred?

Part 2: 2 answers why you want to provide these security features by outlining the attacks/vulnerabilities that you have investigated in the current system (or in other solutions that you have investigated but didn't choose).

3. Defence countermeasures
    - Identifying those points in Part 2: 2. … now it is time to consider what defence countermeasures that could be used / developed as part of any strategy.
        - If you are a red team project, then consider how these countermeasures might impact your approach.
        - If you are doing an overall security assessment / providing recommendations, then consider how any current defence mechanisms may impact your investigations / obscure other issues / limit the investigation capabilities.
    - This is also a chance to have more details on the points outlined in Part 1: 4.
4. Architecture, Algorithms, and models

- This is an expansion / elaboration on (Part 1: 4). In that section, you've outlined the potential detection methods being considered, here is where you elaborate on these in more detail.

5. Security testing
   - What are the different kinds of tests / assessments being considered? (this may change over time as more is discovered about the systems under focus).
6. Deployment / monitoring / reporting
   - Intended approach for deploying / running any tests or analysis scripts
   - How the monitoring of progress and results will be undertaken
   - What reporting / results / recommendations are being provided to the client.

Details for DATA SCIENCE projects

Data science projects will need to extend the above project plan to a more comprehensive scoping document by adding the extra sections as described below. Also, see the marking rubric. For Data Science projects, you will need to follow a project management method such as CRISP http://www.datascience-pm.com/crisp-dm-2/. For phase 1 (Business Understanding) you have already been asked to create a feasibility report and a project plan.

In the scoping document, your team is aiming to communicate the intended scope of work. However, these might change or evolve as you go through your project… and that is ok, because this is a living document and should be kept up to date / reviewed with stakeholders. In the scoping document, include each of the following sections:

1. Data Understanding: collect initial data; describe data; explore data; verify data quality
   - What are the initial data sources being used / provided?
   - What data collection methods are being used?
   - What is the assumed data quality / structure / completeness?
   - What mechanisms will be undertaken to evaluate the above points? Will you have all the data that you need or will there need to be more data collection methods done during the project?
2. Data Preparation (generally, the most time-consuming phase): select data; clean data; synthesize data; integrate data; format data
   - Describe the intended data preparation activities for each of the data sources listed above
   - Describe any data processing pipelines that will need to be done. E.g. noise filtering, smoothing, running averages, error corrections, removing outliers, anonymising any datasets, feature engineering procedures…

- If there needs to be different preparation pipelines for different models, then describe the processes needed for each model.
- Also, describe why these steps would make the data more usable for the intended purposes.
- What data storage mechanisms will need to be used during these processes?

3. Modelling: select modelling technique; generate test design; build model; assess model
   - Which modelling technique / techniques are being used / compared? Why? What is the intended outcome / question being considered and how does it relate to the chosen model?
   - How will the model effectiveness / validity be checked?
   - What test data needs to be provisioned / used for the above points?
4. Evaluation: evaluate results; review process; determine next steps
   - How will the results be reviewed? Are there expectations to be compared to, or is this more of an open exploration / unknown result? If so, how will these results be reviewed / validated?
   - What next steps or questions are being considered depending on the results / outcomes?
5. Deployment: plan deployment; plan monitoring and maintenance; produce final report; review project.
   - What is being given to the client? What training / documentation will be provided?

<u>Details for GAME projects</u>

A requirements document would have:

* High concept: A concise statement of what kind of a game it is, focusing on what the player does and what they experience.

* Core experience: What are the key experiential goals? Is it challenging and exciting? Relaxed and creative?

* Learning outcomes: For a serious game, what are the practical outcomes it is designed to produce? How does it change the player's understanding or ability? Does it motivate the player to do some task?

* Target audience: Usually expressed as a persona, with specific preferences, needs and requirements. E.g. "Malcolm plays puzzle games on his iPhone while travelling on the train to

work". This gives you some constraints on who your game appeals to, where/when they are going to play it, for how long, etc.

* Target platform/interface: The device it is going to be played on, including considerations such as screen-size, controllers (mouse/keyboard/touchscreen/joypad), graphical capabilities, etc, and why these are appropriate for the kind of game you want to make. These can change as you develop, but you should always be working with a target platform/interface in mind.

Please follow the following format:

1…...Introduction
1.1 Document Convention/Intended audience
1.2 Game High Concept and Scope
1.3 Definitions, Acronyms, and Abbreviations
1.4 References
2 .....Overall Description
2.1 Core Experience
2.2 Target Platform
2.3 Target Audience and other User Classes and Characteristics
2.4 User/Training Documentation
3 .....Requirements
3.1 Intended Learning Outcomes
3.2 Core Gameplay Features
3.3 Other non-gameplay Functional Requirements
3.4 Design and Implementation Requirements/Constraints
3.5 Usability Requirements
3.6 Other Non-functional Requirements

<u>Details for SOFTWARE projects</u>

If your project concerns software (e.g. desktop or mobile application, website, game) you will need to create a Requirements Specification. If the product already exists (from a previous team or off the shelf product) you should reverse engineer your own requirements document to understand the product specifications.

<u>System/Software Requirements Specification (SRS)/Requirements Document (6 marks)</u>

The SRS should follow the IEEE standard below and use these headings:

1…...Introduction
1.1 Overview/Document Convention/Intended audience

1.2 Purpose (of software)
1.3 Scope – including a context diagram (Level 0 Data Flow Diagram)
1.4 Definitions, Acronyms, and Abbreviations
1.5 References
2 .....Overall Description
2.1 Product Perspective
2.2 Product Functions
2.3 User Classes and Characteristics
2.4 Operating Environment
2.5 User Documentation
3 .....Requirements
3.1 Functional Requirements
3.2 Design and Implementation Requirements/Constraints
3.3 Usability Requirements
3.4 Other Non-functional Requirements

Follow the above headings. For guidance refer to the IEEE SRS Standard. Please do not pay for the IEEE standard which you can access for free if you are logged on to Macquarie University Networks. You must be signed into the MQ campus network for it to recognise the institution. (this does not mean you can login to IEEE with your MQ OneID).
To access off campus:
https://login.simsrad.net.ocs.mq.edu.au/login?qurl=https://ieeexplore.ieee.org%2fdocument%2f6146379
The unit website includes some SRS examples. If your client has a format they would like you to use, please use that one and check that it covers the above sections.

Marking scheme for scoping / requirements sections

Marking Scheme for Scoping Document (Data Science and Cybersecurity Only)

For Data Science projects	**For Cybersecurity projects**	
Document convention/intended audience	Document convention/intended audience	0.25
Purpose/Project Scope & Context Diagram/	Purpose/Project Scope	0.25
Data Understanding – describe the data	Attacks/Vulnerability Understanding	0.5
Sources	Adversaries	0.5

Capture	Attacks/Vulnerability Detection	0.5
Storage/Environment	Monitoring of Assumptions	0.25
Data quality	Data quality	0.5
Follows Format Specified/Well-presented	Follows Format Specified/Well-presented	0.25
Data Preparation - cleaning needed	Data Preparation	0.25
Integration needed	Data (packages, logs, …) Collection	0.25
Formats needed	Security Feature Extraction	0.5
Data Exploration – Modelling Modelling technique selection	Attacks/Vulnerability Analysis	0.5
Model design constraints	Defence countermeasures	0.25
Model assessments needed	Architecture, Algorithms, Models,	0.25
Evaluation Requirements & Process	Security Testing (e.g., penetration testing)	0.5
Deployment – monitoring, reporting, review, training/support)	Deployment – monitoring, reporting, review, training/support, security polices)	0.5
Customer Feedback	Customer Feedback	PENALTY
TOTAL REQUIREMENTS		6

Marking Scheme for Requirements Document (Software / Web and Games Streams Only)

For Application/Website	**For Games**	
Document convention/intended audience	Document convention/intended audience	0.25
Purpose/Project Scope & Context Diagram/	High Concept	0.5
Definitions/References/Overview	Definitions/References/Overview	0.25
Follows Format Specified/Well-presented	Follows Format Specified/Well-presented	0.25
Product Perspective/Features/	Core Experience	0.25
User Classes and Characteristics/	Target Audience & other User Classes	0.5
Operating Environment/	Target Platform	0.25
User Documentation & Help	User Documentation/Training & Help	0.25

Functional Requirements (FR)“The system shall …”	Learning Outcomes: description and measurement	0.5
FR Uniquely identified	Other non-gameplay Functional Requirements	0.25
FR Organized/structured/grouped		0.25
FR Covers functionality	Core Gameplay Features	0.5
FR Unambiguous/testable/consistent		0.5
NonFR (security, performance, access, etc)	NonFR (security, performance, access, etc)	1
Design/Implementation Req (platform, language, etc)	Design/Implementation Req (platform, language, etc)	0.25
Usability Requirements (ease of use, training, etc)	Usability Requirements (ease of use, training, etc)	0.25
Customer Feedback	Customer Feedback	PENALTY
TOTAL REQUIREMENTS		6

-Deliverables 3 & 4 (Increments 1 and 2)

Common sections for all groups:
Deliverables Certificate
Updated Project plan (with revision history + new “Handover Requirements” section)
Updated Requirements/ Scoping Document (with revision history)
Stream-specific sections (with different criteria):
Design Documentation
Testing Documentation
Prototype / Product (MVP)
+ deliverable 4 stream-specific document
(13 marks for Deliverable 3, 13 marks for Deliverable 4)

All Teams: Deliverable 3 & 4 submission instructions

PLEASE INCLUDE YOUR TEAM NUMBER in the file name.

The next submission contains multiple documents. However, for marking I would like one .pdf document in this order:

1. Revised Project Plan/Quality (complete plan, with your changes and revision history).
2. Revised Requirements/Scoping document (include complete document with your changes and revision history).
3. Analysis & Design Document including Assumptions at the end of this document.
4. MVP/Prototype Document
5. Testing Document
6. User Manual (**for Deliverable 4 only**)

Each of these documents will have their own Table of Contents - or if it suits your formatting better, you could make one TOC at the start.
You could create these pdf documents separately and then do a merge - if you have the right software. Include Deliverables Certificate as the first page.

Warning- Submission of documentation should be done by the due date via iLearn. Documentation must be consistent and relevant for the prototype demonstrated. The client typically is interested in the product, not the documentation. A poor prototype or redundant/inconsistent documentation will bring down the documentation mark.

Common sections for all groups

Add revision tables to the start of all documents just after the TOC to indicate the revision number, date, person and change to the previous version (be specific about what changed / why).

Updated Project Plan

Revised Project Plan (0.5 mark **Del 3 and 0.1 marks for Del 4**)

The following is required:

1. **Include the heading in the project plan: Handover requirements.** This section can be added after your schedule to explain in detail what the client needs. Handover will be different according to what the project is. It's about making sure the client gets what they were expecting and need before the team disappear. So any data, software, algorithms, scripts, test cases, testing tools/environments, outputs and documentation would be

handed over together with training/manuals that lets them go forward with it - including reports or test cases showing current output and/or what is fully functional or not.
For all projects, the point is to discuss with the client what they need at the end of the project. Make sure they have everything they need and know how to use what you have given them. There is no set format or requirement for the marker.

a. For games and software projects, think about who will do play testing and/or acceptance testing, asking questions such as when will testing start, when and what devices/sites does the final product need to be deployed on, who will maintain the system, how will the system be maintained, what are the user manual/training manual/installation/installation wizard/administration parameters requirements, what database/file setup is required, what documentation is required.

2. **A revised and updated version of your project *schedule*,** that takes into account any schedule revisions, covering the balance of the project, that have arisen since you submitted the previous schedule. Anything else that has changed such as team structure, resource allocation, risk evaluation, change management processes, communication strategy, etc should also updated in the plan and quality manual and included in the revision history. Resubmit the entire updated document.

Updated Requirements / Scoping Document **(0.25 marks Del 3 and 0.1 marks for Del 4)**

Add a revision history table at the start of your document to clarify all changes made from Deliverable 2. No new sections are required. Please review feedback to Deliverable 2 submission and consider any project changes or sponsor feedback and update the document accordingly.

List of Assumptions

(0.25 marks Del 3 and 0.15 marks for Del 4)

Provide a list of assumptions (relevant to design, requirements, project plan or any of the documentation). This will help you think about them and whether they are reasonable and also help the marker/review team to understand why you have done certain things. Please review the assumptions as a group before submission. A poor assumption will not be a valid reason for poor design decisions.

Prototype / MVP Document

(4.5 marks for increment 1-**Del 3** and 5.5 marks for increment 2-**Del 4**)

Before the due date/time you need to arrange a time with the client to demonstrate your system and receive feedback. The team must organise that meeting. All team members are expected to attend. A valid reason for non-attendance by a team member must be arranged before the meeting. When the demonstration/meeting is arranged please let the convenor know of the arrangement via the meetings thread on iLearn.

Under the heading "Prototype", include at least a page in your increment 1 and 2 documentation to present and discuss your prototype. Screen shots and how the product looks and works should be included.

Also include a subheading "Sponsor meeting, feedback and response to feedback" in the "Prototype section". Include details of the sponsor meeting and feedback. All sponsor feedback on any aspects of your project should be included in this section. Also clarify how the team have or will respond to that feedback.

If you have not been able to arrange a meeting before submission due to sponsor unavailability state that under this heading and when the meeting is scheduled. After the meeting, include the feedback and response in your next weekly report submitted on iLearn.

I will also be contacting the sponsor. The client will be asked to confirm the meeting and their feedback via a survey sent to them.

Prototype Feedback Adjustment

The marks for your prototype are made up of what you present in your document, the feedback reported and the client's feedback to me. (For Increment 1 marks are allocated as follows: 0 if no evidence or mention of a prototype, 1 if some evidence identified, 2 if evidence provided, 2.5 if brief, just screens and description or just feedback, 3 if shown with screens and feedback provided. 3.5 if shown in detail with feedback and responses to feedback, 4 if sponsor is happy 4.5 if sponsor is very happy.)

Analysis, Design + Testing Documentation for DATA SCIENCE projects

The main objective with the iterations is to build up as much of the data processing pipeline and describe the decisions / choices made - as well as how any choices and outputs will be evaluated. Someone else coming in late to the project should be able to use this deliverable to understand

what the team is doing and where they are up to with their deliverables - including the rationale behind any design decisions.

Students choosing the data science stream can leverage what was built up in data science units (like COMP257 / COMP2200) and discuss what has been done in their project. Much like in the project portfolio and project presentation from COMP257 / COMP2200, the aim here is to communicate different aspects of the project implementation including the data, the goals, the options, the decisions, the choice of models and starting configurations, and the resources available.

For DATA SCIENCE projects, you should seek to establish a baseline model for the project – the simplest model you can think of for the problem. This would be a good deliverable as a MVP since it encourages you to have an end-to-end processing pipeline and to establish a baseline that you can then try to beat with their refined model in the subsequent increments.

Your documentation should include the data manipulation process with clear instructions on how to run them to turn the raw data into the input for modelling. One possible approach could be to load / maintain them in a GitHub / Bitbucket / Firebase repository with appropriate permissions. However, this is something that should be discussed with your sponsor and the convenor.

You should present the results of the modelling via evaluations in the form of a report. The report should outline the different models used and the results observed. Interpreting the results and discussing implications for the business questions that prompted the study are also a key part of this. In many cases this can be structured like a traditional scientific paper – material, methods, results, discussion.

Feature engineering: (deciding what you are looking for and how to go about it)

- From the data, are there any trends / ranges to look for?

e.g. if you are analysing time-based movement data, what accelerometer and gyroscope ranges / trends do you want to look for to identify someone that has fallen / collapsed?
e.g. if you are looking at financial data, what does an "upward sale trend" look like?
e.g. if you are looking at population data, or education rates, what "characteristics" would you want to look at and why?

- Describe what data characteristics are being looked for, and how your data pipeline is being processed to generate these features.
- Give each different "feature" or "characteristic" a name and then attribute some form of data ranges / statistical definition.

Solution Architecture: (choice of macro architecture / pipeline)

- Provide an overall description of each section of the pipeline including the data in and out of each "stage". The data details can be properly described in the "Detailed data descriptions" section.
- This would most likely be a more detailed version of the overall pipeline presented in the team's scoping document as some of the stages would now be implemented / finalised.
- Include any resources available / resource processing constraints to each of the sections in the pipeline (e.g. processing / timing limits)

Algorithms / models methods: (detail what is in each part of the solution architecture, including models used and initial conditions / config settings)

- selection of model... there are many different approaches: predictive, supervised / unsupervised, classifiers, ... which are going to be used? why? and why chose those over other approaches?
- Are there any settings needed (eg. in a KNN model, what is the number of the nearest nodes being used in the application of the classifier?)

Detailed Data descriptions:

Data being used, data being generated, data being stored, as well as any summaries and/or reports.

Model evaluation:

- how will any data / outputs be compared / tested / evaluated for correctness and accuracy?
- If you are choosing between models, how will the models be compared / contrasted to see which one has a better performance (e.g. if you are comparing different classifier models, on what basis are you comparing them? Detail each comparison.

Performance evaluation results:

what are the results of any tests run so far and what are the future planned tests for future iterations? (e.g. if this is Deliverable 3, what is planned for Deliverable 4? IF this is Deliverable 4, what is planned before the handover?)

Refer to the rubric / marking scheme ("All Teams: Deliverable 3 and 4 marking schemes") for sections to be included in the design and testing documentation.

Deliverable 4 (Only) stream-specific document: for Data Science Teams

Scripts / Model Execution (2 marks)

The *Scripts and Model Execution Documentation* should not exceed **twenty (20) A4 sides.** This is essentially a user or training manual for the one or more users of your MVP. It might include

model parameters, how to set up the models, data sources/format, file locations, etc. The intended user/s may differ according to the project and there could be more than one type of user with different documentation needs.

The content of this document is something that you MUST discuss before **Del 3** when you talk to your client about what they expect at handover at the end of the project (see point 1 under updated project plan for Deliverable 3), ask them what their requirements are regarding support/maintenance. Find out what they need regarding training, help, installation, configuration, etc of your product/output.

Like all technical writing, the document should have a clear Table of Contents (TOC) and well-organised content. In addition to documenting your models and how to run them, this document could involve sections for different types of users (e.g. Admin and end users), installation guide, configuration settings, screen shots with example data, APIs, training, steps describing usage troubleshooting/where to get help, etc. Check you have page numbers, or your TOC will have a limited value.

Analysis, Design + Testing Documentation for CYBERSECURITY projects

Discuss with your client what they want to receive from you – that will be your MVP. It could be a *Penetration Testing Document* as in the example provided on iLearn. It could be a list of recommendations and vulnerabilities discovered. It could be an overview of attacks or defences tried (if you are doing red team / blue team approaches). Refer to the rubric for content and sections to be included in the analysis, design and testing documentation.

User stories describing the security / privacy attacks:

- What are the points of interest in the system and what attacks were considered / attempted?
- Be sure to include the sections / areas that these were applied.

Detection of intrusion / vulnerabilities:

- What did you find and how were you able to determine that the intrusion was successful / a vulnerability was discovered?

Defending solutions architecture:

- What was the architecture of the system under focus?
- Also, describe the mechanisms / process flow for your own investigations / attacks / defences (depending on the mode of your project). This could be represented as some form of “pen-testing architecture”.

Non-technical concerns/approaches:

- Describe other cybersecurity aspects such as governance, human-factors (e.g. social engineering), processes, staff training, etc, that will impact on the security of IT infrastructure for the organisation.

Algorithms / security rules:

- Describe the pen-testing algorithms / approaches used (or the defending equivalent if you are focusing on that approach).

Data description / feature engineering:

- What data have been collected / used / provided?

Security test plan:

- This is a detailed overview of the entire testing intent for the project including: Timelines, approaches, resources required, times, people involved, approvals, reporting needs, overview and list of test cases. Refer to sample documentation in iLearn for general test plans that could be modified for security projects.

Testing reports:

- Detail the specifics for each test case what / when / where / how …
- What results have been found so far and what remains to be tested / followed up? This can be a comparison to your initial planned scope.
- What tests are yet to be completed / attempted?
- Are there any new tests / approaches being considered after having discovered more about the system under focus?

Be careful not to focus only on the technology side of the matter, rather than taking a holistic view. It may work well if the context and client can support this approach. However, in cybersecurity projects where the client is needing cybersecurity support but not wanting the team to conduct penetration testing or other technical changes to their current system, the team may need to think more broadly about cybersecurity and how to assist their client to make their systems/business more secure, now or in the future.

Note that it is quite common that sometimes clients don't even know what they want. Clients may have a security concern in mind, so they come with a security project, however, the reality is that they need a holistic IT solution instead. For example, if the organisation was having issues with their current web hosting service and are looking for a new solution provider, there wouldn't be much value in conducting pen-testing. Instead the team should first focus on requirements engineering and help the client to reconsider the focus of their proposed project. Having said that, cybersecurity can still be the focus but there is a need of a holistic view. What are the major security concerns in this service? Such as users (admins without strong IT background, general public, school kids).

If you are not doing a pen-testing focused project, ensure in the testing section to provide explanation of any changes and discussions on alternatives, rather than pen-testing specifics.

What can be tested? For example, the team can focus on the development of suitable metrics for measuring effectiveness of policies. - looking for
evidence of some structured approach to making the recommendations, such
as use of frameworks and standards, methodologies like threat modeling
and attack trees, threat catalogues like BSI Grundschutz, etc. which
would put the recommendations on a more formal footing than just "a list
of good ideas we brainstormed". ;)

Deliverable 4 (Only) stream-specific document: for Cybersecurity Teams

Recommendations / Monitoring Plan: Provide an early draft of recommendations / results reported so far based on findings. This should not exceed **twenty (20) A4 sides.** The intended recipients will differ according to the project and who is the target recipient of your MVP (e.g. the sponsor's organisation or one of their clients). The document should have a clear Table of Contents (TOC) and well-organised content. Check you have page numbers, or your TOC will have limited value.

The content of this document is something that you should discuss when you talk to your client about what they expect at handover at the end of the project (see point 1 above under updated project plan for Deliverable 3), ask them what documentation they require to support the MVP such as executing, monitoring, training, help, installation, configuration, etc of your product/output.

Analysis, Design + Testing Documentation for GAMES projects

Design deliverables could include:

Game design – for in-game functionality

* Storyboards: A power-point 'flip-book' showing frame-by-frame the core game mechanics, illustrating how they achieve the experience goals
* Art-book: concept art and/or images from other games or media that represent the art style you have in mind for the game
* UI mockups: showing the important information elements and how they are communicated to the player
* Paper prototypes: playable versions of the core gameplay implemented using cards, tokens, dice etc
* Software prototypes: lo-fi implementations of the core gameplay to demonstrate to "find the fun" — i.e. see whether the gameplay achieves the target experience. Ideally these should be the simplest thing needed to convey the experience. i.e. no art, no polish
* Design document including the following in-game and out-of-game aspects:

Game design: including storyboard, art-book, UI mockups with window navigation diagram to show game screen flow.

Game mechanics: how they function doing play, and how they are intended to achieve the experience goals.

Game scaffolding – for out-of-game functionality
Design of out-of-game features like leaderboards, setting up permissions, reporting, adding in new challenges / quiz questions.
To document these software features you may use activity diagrams for intended flows/algorithms, use case diagram (for different out-of-game features). You can refer to the software development section to see more about documenting software.

<u>System Design Document</u>
This will include the basic architecture of the system and the high-level strategy decisions. Use the following headings and explain if and why it is not relevant for your project, if that is the case. You need to include a description of the:

a) system architecture – including package diagram with description that is consistent with the system architecture
b) storage/persistent data strategy
c) any concurrent processes or data and how they will be handled if any exist
d) user interface strategy e.g. tracker, spoken dialogue, phone input, form/menu-based, GUI, etc
e) design decision choices and trade-offs

Data Definitions: Create a table showing what data is needed to be stored in files/database. Include data field name, type and example. For relational databases, for each table/file show the name of the field, the primary key (if applicable), the field type and an example of data in this field. E-R diagrams are recommended.

Game Testing deliverables:
* Playtesting should be done at all stages of development (hence the need for rapid prototyping).
* Playtesting should target specific questions about behaviour and experience: Does the player do what you expect them to do? Does the player feel what you want them to feel?
* Qualitative testing: observe play and take notes about what players do, interview or survey players to ask them what they feel.
* Quantitative testing: instrument the game to track relevant statistics: time of play, where do players succeed/fail at the game.
* Testing: other forms of testing to ensure testing of all functions e.g. adding players, updating scenarios, etc.

<u>Test Spec (This can resemble the test plan and tests case templates from Software projects)</u>
Acceptable documentation for games projects should include:

1. Test plans, including testing strategy (white/black box) and types (unit, system, penetration, regression, etc), detailed test schedule, testing tools and resources assigned, testing milestones and test deliverables and covering scheduling and resourcing of all testing processes.
2. Playtesting questions and results; qualitative testing; quantitative testing
3. For non-game elements include test case specifications:
    a) Identifier
    b) Test description
    c) Input specifications (e.g. actual input to be tested or link to be tested) (INCLUDE range of test values – possible valid and invalid input)
    d) Output specifications (e.g. expected outputs)

Deliverable 4 (Only) stream-specific document: for Games Teams

Help / Training Documentation (2 marks)

The *Help / Training Documentation* should not exceed **twenty (20) A4 sides.** The intended user/s will differ according to the project. For most games or software projects, the document should enable a moderately computer-literate user, initially completely unfamiliar with the system, to understand and fully utilise its functionality. Note there may be more than one type of user with different documentation needs. Training to use the game could also be provided as part of the game. This could involve special keys and screens with help and/training videos. Screenshots of help/training or the training video could be provided in place of or in addition to creation of a document. Please confirm alternatives to documents with the unit convenor well before any deadline. It is likely you need a combination of both documented (for installation, management and maintenance) and in-game support.

The document should have a clear Table of Contents (TOC) and well-organised content. Content will depend on the project but could involve sections for different types of users (e.g. Admin and end users), installation guide, configuration settings, screen shots with example data, APIs, training, steps describing usage troubleshooting/where to get help, etc. Check you have page numbers, or your TOC will have limited value.

As part of the discussion that you should have already had with your client about handover to them at the end of the project (see point 1 above under updated project plan for Deliverable 3), ask them what their requirements are regarding support/maintenance. Find out what they need regarding training, help, installation, configuration, etc of your product/output.

Analysis, Design + Testing Documentation for SOFTWARE / WEBSITE projects

For teams working on applications and websites follow these instructions.

ANALYSIS DOCUMENTATION

Analysis Documentation MUST include each of the following:

- Use Case Diagram
- Use Case Description (for each use case on the diagram)
- User Stories

Use Case Diagram

This is a graphic model showing the actors, the use cases and the relationships between them. One page should be sufficient. As a rule of thumb – if the description of a use case is very short (e.g. one or two steps only) or very similar to another use case, then consider combining the use cases and describing the alternate courses of action within that use case. Structure the use cases into logical groupings. Remember it's from the users' point of view – not the developers. See resource on iLearn on how to draw Use Case diagrams and common misunderstandings and errors.

Use Case Descriptions (one for each use case in the use case diagram)

A use case is a chunk of functionality. You must have a use case description for each use case. This will elaborate all the ways (uses and the steps to achieve them) in which people will achieve them. If you have many use cases, possibly some of your use cases are actually steps in a use case, not a chunk of functionality. e.g. if they just have one step (like "save", "submit", "authenticate"), they are probably a step in another use case and not their own use case.

When you start to write the use case steps, it will become more obvious whether you have identified separate chunks. Also, if you are repeating steps, then this is a candidate for reuse and should be an "includes" use case. Check you are not joining use cases together in sequences. Only includes, extends and generalisation are valid lines between use cases. E.g. "login" should not be joined to other use cases since you could login and then do nothing, or choose from a range of functions/use cases.

If you have 10 chunks of functionality then you need 10 use case descriptions. How will you know what to implement, if you don't know what triggers each use case, which user initiates, what the steps are, etc?

Please use the template on iLearn. Sub-use cases can be combined into one use case description. For example, if you have a use case Maintain Client with sub-use cases Adding Client, Deleting Client, Viewing Client or Modifying Client, you can just write one use case description for Maintain Client and describe the differences by branches in the use case steps.

User Stories

In line with agile practices, write user stories for your product. See the following link to get some tips: http://en.wikipedia.org/wiki/User_story. You can list these in a separate section or add the appropriate user story in your use case description.

DESIGN DOCUMENTATION

Design Documentation MUST include each of the following headings. If you choose not to include any of these items, provide an explanation under these heading.

1. System Design Document
2. User interface layouts / Report layouts (if needed)
3. Window Navigation Diagram (be sure to show backtracks / alternate navigation paths)
4. Data definitions
5. For every use case description create:

Activity Diagrams

OR

Sequence Diagrams (if OO programming language being used)

If classes/objects are being used:

6. Class Diagram
7. State Diagrams for objects with interesting behaviour

System Design Document

This will include the basic architecture of the system and the high level strategy decisions. Use the following headings and explain if and why it is not relevant for your project, if that is the case. You need to include a description of the:

f) system architecture – including package diagram with description that is consistent with the system architecture
g) storage/persistent data strategy
h) any concurrent processes or data and how they will be handled if any exist
i) user interface strategy e.g. tracker, spoken dialogue, phone input, form/menu-based, GUI, etc
j) design decision choices and trade-offs

User Interface Layouts

Show the actual screens or use a drawing package, system builder like e.g. Jbuilder, interface designer tool.

Report Layouts

If you are producing any reports (e.g. weekly sales report, daily summary, transaction listing), you need to design them. You need to consider things like report title, column/page headers,

report totals/subtotals, fields, how many per page, etc). Different types of reports have different real estate (e.g. paper size or screen size). For example, if you needed to print a sales docket you need to plan what goes at top and bottom (e.g. name of company, ABN, store location, sales person's name, payment method, etc) how many spaces you have across and down and how to layout the docket so it will fit on the paper or screen (e.g. computer/mobile) to be used.

Window Navigation Diagram

Show how the different screens will link together (screen flow) and what triggers a transition from one screen to another. Activity diagrams are recommended.

Data Definitions

Create a table showing what data is needed to be stored in files/database. Include data field name, type and example. For relational databases, for each table/file show the name of the field, the primary key (if applicable), the field type and an example of data in this field. E-R diagrams are recommended.

Class Diagram

If you are using object-oriented programming please include a class diagram. Make sure any classes or methods on your sequence diagrams have been included on the class diagram. Method signatures should be given. The diagram must include all of the following:

- classes
- attributes
- associations
- inheritance and/or aggregation (if applicable)
- traversals
- multiplicities

Sequence Diagram

If you are using object-oriented programming please include one sequence diagram for each use case description. It should be possible to read the description and follow what is happening on the diagram. The objects and messages must be valid and shown on the Class Diagram.

Activity Diagrams

If you are NOT using object-oriented programming please include one activity diagram for each use case description. It should be possible to read the description and follow what is happening on the diagram.

State Diagrams

If you are using object-oriented programming include state diagrams for any objects that have interesting states or complex behaviours. A state diagram shows the life cycle of an object and

thus all important objects can each be represented in their own State Diagram. However, you are required to consider the life cycle of each object in your system and to submit diagrams for those that have interesting states or complex behaviour. One way to measure if a state is interesting is to consider whether you need to test that state before performing a particular action or if the state changes after an action is performed. What is interesting will depend on the application. In most cases when an object is updated or printed (updated and printed can be states themselves but are generally not very meaningful) that will not change more interesting states such as paid/unpaid, married/single or for sale/sold.

Testing Documentation

Test Specifications (3 marks)

Acceptable documentation for software projects should include:

4. Test plans, including testing strategy (white/black box) and types (unit, system, penetration, regression, etc), detailed test schedule, testing tools and resources assigned, testing milestones and test deliverables and covering scheduling and resourcing of all testing processes.
5. Test case specifications:
    a) Identifier.
    b) Test description.
    c) Input specifications (e.g. actual input to be tested or link to be tested) (INCLUDE range of test values – possible valid and invalid input)
    d) Output specifications (e.g. expected outputs)

See templates for Test Plans and Test Cases on iLearn under resources on iLearn. Use of these templates is optional and suited to projects delivering a software product. If you prefer, you can use testing documents from previous units or other source.

Testing documents for games, cybersecurity and data science projects should contain content in accordance with the marking rubric.

Deliverable 4 stream-specific document: for Software Teams

User Manual Documentation (2 marks)

The *User Documentation* should not exceed **twenty (20) A4 sides.** The intended user/s will differ according to the project. For most games or software projects, the document should enable a moderately computer-literate user, initially completely unfamiliar with the system, to understand and fully utilise its functionality. Note there may be more than one type of user with different documentation needs.

The document should have a clear Table of Contents (TOC) and well-organised content. Content will depend on the project but could involve sections for different types of users (e.g. Admin and end users), installation guide, configuration settings, screen shots with example data, APIs, training, steps describing usage troubleshooting/where to get help, etc. Check you have page numbers, or your TOC will have limited value.

As part of the discussion that you should have already had with your client about handover to them at the end of the project (see point 1 above under updated project plan for Deliverable 3), ask them what their requirements are regarding support/maintenance. Find out what they need regarding training, help, installation, configuration, etc of your product/output.

All Teams: Deliverable 3 and 4 marking schemes

Revised Project Plan	D3: /0.50 D4: /0.25
Revised Scoping/Requirements	D3: /0.25 D4: /0.10
Not well-presented, spelling mistakes, language not businesslike	PENALTY
Deliverables certificate not included	PENALTY
List of Assumptions	D3: /0.25 D4: /0.15
Prototype/MVP/Project Output Document	D3: /4.50 D4: /5.50

Analysis and Design Documents D3 and D4 marking scheme							
Software	**D3 D4**	**Games**	**D3 D4**	**Data Science**	**D3 D4**	**Cyber**	**D3 D4**
Use case diagram	D3: /0.75 D4: /0.30	Art book	D3: /0.75 D4: /0.40	Feature Engineering (selection, construction)	D3: /1.25 D4: /0.80	User stories describing security/privacy attacks	D3: /1.00 D4: /0.50
Use Case descriptions and User Stories (Well-structured, sensible, complete)	D3: /1.00 D4: /0.40	Storyboards	D3: /1.00 D4: /0.40			Detection of intrusions / vulnerabilities	D3: /0.75 D4: /0.30

System Design Document	D3: /1.00 D4: /0.40	Design Document	D3: /1.00 D4: /0.50	Solution Architecture	D3: /1.25 D4: /0.40	Defending Solutions Architecture	D3: /0.75 D4: /0.35
Design – includes UML models, structure charts, Report Layouts	D3: /0.75 D4: /0.50	Functional Design - – includes diagrams and Report Layouts	D3: /0.75 D4: /0.40			Non-technical concerns/approaches	D3: /1.00 D4: /0.35
User Interface Layouts, Screen Navigation	D3: /0.50 D4: /0.20	UI mock-ups	D3: /0.80 D4: /0.20	Algorithm/Models/Methods	D3: /0.75 D4: /0.40	Algorithms/Security Rules	D3: /0.80 D4: /0.40
Data Definitions/Schemas/ER	D3: /0.50 D4: /0.20	Data Definition	D3: /0.20 D4: /0.10	Detailed Data descriptions	D3: /1.25 D4: /0.40	Data Description / Feature Engineering	D3: /0.20 D4: /0.10
User Manual **(D4 only)**	D3: /0.00 D4: /2.00	Help/Training **(D4 only)**	D3: /0.00 D4: /2.00	Scripts/Model Execution **(D4 only)**	D3: /0.00 D4: /2.00	Recommendations / Monitoring Plan **(D4 only)**	D3: /0.00 D4: /2.00
D3 Subtotal: /4.50 D4 Subtotal: /4.00		**D3 Subtotal: /4.50 D4 Subtotal: /4.00**		**D3 Subtotal: /4.50 D4 Subtotal: /4.00**		**D3 Subtotal: /4.50 D4 Subtotal: /4.00**	
Test Specification							
Test plan	D3: /1.75 D4: /1.75	Playtesting & Functional Test Plan	D3: /1.75 D4: /1.75	Model Evaluation	D3: /1.50 D4: /1.50	Security Test Plan	D3: /1.25 D4: /1.25
Test-case specifications	D3: /1.25 D4: /1.25	Functional Test-Cases & Playtesting results	D3: /1.25 D4: /1.25	Performance Evaluation results	D3: /1.50 D4: /1.50	Testing Reports	D3: /1.75 D4: /1.75
D3 Subtotal: /3.00 D4 Subtotal: /3.00		**D3 Subtotal: /3.00 D4 Subtotal: /3.00**		**D3 Subtotal: /3.00 D4 Subtotal: /3.00**		**D3 Subtotal: /3.00 D4 Subtotal: /3.00**	

Deliverable 5 Details

Deliverables Certificate
Final Group Reflective Report (8 marks), approx. 7-20 pages

The sections in the Final Report are up to you but you may like to follow the system development life cycles and reflect on what your team learnt in each of the phases. Possible section headings could be:

1. Introduction
2. Project Planning
3. Requirements and Analysis
4. Design
5. Implementation
7. Learning Outcomes
8. Conclusion

This document is NOT a rehash or resubmission of the documents submitted previously. You should review all aspects of your project (information, outputs, documents, weekly reports, meeting minutes, etc) and reflect on what you did, why you did it, what worked and what should have been done differently.

You should aim to make this document self-contained. You may include information / diagrams / snippets from other documents to make your discussion more informative. Assume that someone was reading the document to discover what you had been up to for the past 3 months and could also see why you drew the conclusions that you did. Don't assume they have read all the other documents.

There is no rubric or set structure. The key is reflection. Including individual reflections are also encouraged.

Deliverable 6 Details

Project Presentation (10 marks)

Each group will be given a time slot to present their project and demonstrate their system to the class, sponsors and academics in one of the lecture theatres. You must attend the whole session you are assigned to, not just your project presentation. Each team member must participate in the presentation, but not necessarily for the same length of time. The presentation mark will be based on your individual and group mark. Put your name on the bottom of each slide you present.

Your presentation should describe the problem being addressed, what you did to address the problem and how you went about coming up with solutions. Your presentation should include, assessment of (reflection on) the project and software process and future/outstanding work. Demonstrations should contain: Functionality of software, Non-functional qualities of software; examples of how to the system would be used, by whom and for what purpose.

Make sure that your system will run in the lecture room that will be used. Check with the convenor beforehand if in doubt. You may use the internet and computer in the lecture theatre, but if special software is needed you should demonstrate on your own machine.

As shown in the rubric below, a group mark will be given for presentation structure, communication of content and visual aspects of the presentation. An individual mark will be given for speaking/presentation skills. The four measures all overlap somewhat to make a coherent presentation. Presentation structure concerns the sequence in which the content was delivered and the flow of concepts. Communication of content is how is it delivered - via a demo, video, slides, just talking, how well do we understand what the project was about - you might have a nice structure but then it is not well described, the points are not relevant or they contradict. How understandable, logical and informative was the content communicated.

Grades/10 **(U)nsatisfactory (<5)** **(F)unctional (5-6)** **(P)roficient (7-8)** **(A)dvanced (9-10)**	Presentation Structure	Communication Of Content	Visual Aspects of Presentation	Presentation Skills
Group Number				
Individual Name				

Each item is marked out of 10. The group marks are averaged. The average of the group mark and individual mark will be the mark you receive.

Deliverable 7 Details

Handover to Client (10 marks from Client)

SOFTWARE & SUPPORTING MATERIALS
You must discuss with your client what they need to receive from you. Do they need a manual, install wizard (which platforms/OS)? What files/database needs to be set up, how can they run and maintain your system? This discussion should have started before submission of Deliverable 3 and continue.

When you have handed over what has been agreed, the sponsor must check it works on their setup (or you can check that for them). After they can confirm it is working, then they will provide a mark out of 10 for each individual student.

It is not sufficient to hand over a zip file and wish them all the best. You must make sure they can run your system – unless they agreed that they have technical people who will handle this. So you need to make sure your system is fully tested, well documented and bug free before handover. You should arrange a date for handover and allow your sponsor up to week to confirm they can follow your instructions and use the system. Latest date for handover is final week of exam period. Leaving handover to the latest time may mean you receive an incomplete for the current semester as there may not be enough time for the sponsor to provide a mark and for it to be processed in your end of semester results.

Additionally for Deliverable 7, on iLearn upload a zip file that includes the latest version of all documents created over the semester. This is primarily an archive for the unit. You should update all documents that you will be handing over to your sponsor based on any feedback since the last deliverable. What is submitted will not be assessed, so teams may submit the latest previously submitted version but preferably the documents updated for the sponsor, i.e. the latest version, should be uploaded.

If your code or other files are too large or the structure is too complex to zip and upload, then you do not need to include them in this upload. For all projects, but particularly if you can't upload to iLearn, please ensure the client has been able to access the files and use them since there will be no other record of them that they can access in the future.

[1] For your Gantt chart, you can get MS Project through MQ site here: https://staff.mq.edu.au/intranet/science-and-engineering/services-and-resources/it-support-services/miscellaneous/microsoft-imagine/_nocache

COMP3850 Computing Industry Project

Read this document carefully and by the due date/time. All team members should be familiar with and refer to this Project Deliverables Definition Document. The team Quality Manager should ensure the team follow the instructions correctly.

As part of COMP3850 you will be formed into groups of 4-6 students. You will work together to understand what your client requires and go through a number of phases from assessing the feasibility, requirements, design, implementation and testing of a prototype or other artefact with associated documentation, as appropriate for your project. You can expect requirements and client needs to evolve over the period of the project.

The project is worth 75% of the course assessment.
The other assessment for the course is the **final exam** worth 25%.

The final marks for Deliverables 1-5 may differ for each member of a group if a certain student does not contribute equally to the group's deliverables according to the contributions submitted by team members. At the two checkpoints identified in the deliverables schedule, each individual must fill in an online individual contribution form to state the percentage contribution of each team member, including themselves, to the deliverables included in that contribution form (see Deliverables Schedule).

The project will follow an agile life cycle and is made up of a number of deliverables that will be iteratively updated and expanded.

Contents

Summary of Group / Individual Assessment Expectations

1) List of assessment items
2) List of individual contribution form submission checkpoints
3) List of communication and involvement expectations
4) Semester 1 2023 – Deliverables Schedule

Deliverable 1 Details

Deliverable 1 marking scheme

Deliverable 2 Details

Project Plan including Quality Manual
Marking scheme for project plan and quality manual sections
Requirements Document/ Scoping Document
A note to all groups
Details for CYBERSECURITY projects
Details for DATA SCIENCE projects
Details for GAME projects
Details for SOFTWARE projects
Marking scheme for scoping / requirements sections

Deliverables 3 & 4 (Increments 1 and 2)

All Teams: Deliverable 3 & 4 submission instructions
Common sections for all groups
Updated Project Plan and Quality Manual
Updated Requirements / Scoping Document (0.25 marks Del 3 and 0.1 marks for Del 4)
List of Assumptions
Prototype / MVP
Prototype Feedback Adjustment
Analysis, Design + Testing Documentation for DATA SCIENCE projects
Analysis, Design + Testing Documentation for CYBERSECURITY projects
Analysis, Design + Testing Documentation for GAMES projects
Analysis, Design + Testing Documentation for SOFTWARE / WEBSITE projects
All Teams: Deliverable 3 and 4 marking schemes

Deliverable 5 Details

Deliverable 6 Details

Deliverable 7 Details

Summary of Group / Individual Assessment Expectations

1) List of assessment items

(D1 to D5 must include a deliverables certificate starting from the first page)

Deliverable 1 (D1):
Feasibility Study

Deliverable 2 (D2):
Project Plan + requirements/scoping document

Deliverable 3 (D3):
Updated D2 documents + design documentation + test documentation + Prototype/MVP

Deliverable 4 (D4):
Updated D2 documents + updated D3 documents + user / training manual

Deliverable 5 (D5):
Final Group Reflective Report

Deliverable 6 (D6):
Project presentation and demonstrations

Deliverable 7 (D7):
Delivery of product to sponsor

Deliverable 8 (D8):
Final exam

2) List of individual contribution form submission checkpoints

(marks may be deducted if the following are not completed)

Individual Contribution form 1: *covers deliverable D1 and D2*

Individual Contribution form 2*: covers deliverable D3, D4, and D5*

3) List of communication and involvement expectations

(marks may be deducted if the following are not completed)

Induction in week 1 (C1)

Team training in week 2 (C2)

Weekly reports on the group forum (C3)

Meetings Register (for off campus meetings ONLY) (C4)

Deliverables Certificate and Time/worksheets (C5)

Essential Sponsor Communications (C6)

4) Semester 1 2023 – Schedule

Wk	Deliverable	Product	Marks	Due - Date and Time
2	**C2**	Attend and participate in Week 2 team training	**1**	**5-7pm, Thursday 02/03/2023**
3	**D1**	Feasibility Study*	7	Thursday 09/03/23
6	**D2**	Project Plan and Requirements/Scoping Document*	13	Thursday 30/03/23
6	**Individual Contribution Form 1 For (D1 + D2)**	**Checkpoint to complete first half Individual Contribution form on iLearn		Saturday 1/04/23
8	**D3 (Inc1)**	Updated Deliverable 2 documents, plus Design, Test Cases, Prototype/MVP*	13	Thursday 27/04/23
11	**D4 (Inc2)**	Updated Increment 1 deliverables, plus user/training manual *	13	Thursday 18/05/23
13	**D5**	Final Group Reflective Report*	8	Thursday 1/06/23

13	**D6**	Project Presentation / Demonstration	10	Thursday 1/06/23
13	**Individual Contribution Form 1 For (D3 + D4 + D5)**	**Checkpoint to complete second half Individual Contribution form on iLearn		Saturday 3/06/23
13+	**D7**	Final Delivery of the Product to the sponsor. This should occur by Week 13 but can be extended to end of exam period or Team grade will be (I) Incomplete. Upload of all project documentation (including other files are optional)	10 (from sponsor)	Thursday 1st June up to Thursday of Week 16, 22nd June.
14+	**D8**	Final Exam	25	Week 14/15 Time/ location TBA

*For deliverables 1-5:
One team member in the group should **submit a single pdf named 'Group[num]_Del[num].pdf'** that includes the Deliverable Certificate (which includes timesheets) at the start of the document. Place content in the order used in this document.
For Deliverable 7, on iLearn upload a zip file that includes the latest version of all documents created over the semester.

Additional Activities – penalty may apply for non-compliance		
C1 Induction	Attend and participate in Week 1 to learn about unit structures and meet team and sponsor.	Week 1 lecture/ class

C3 Weekly reports	Start a new thread for each week with Group # Week# in the thread title. One per team. Stating – weekly activities, progress, any meetings (date, time, location, attendees), issues, work to be done in coming week by each individual. Marks for the group may be deducted from the overall grade for < 8 satisfactory weekly reports.	Via iLearn Forum – post to team only
C4 Meetings Register	Each time you meet off campus with your client, before the meeting record the upcoming event. Create/Add a post to Meeting Register thread in forum with the following information: Date, time, location, 2 phone numbers, names of all students attending, sponsor name	Via iLearn Forum – post to team only
C5 *Deliverables Certificate and Time/work sheets	Each student from each group will have to sign the front page of the deliverables certificate. One person from each team will upload the deliverables submission to iLearn by the due date/time. On the reverse side of the deliverables certificate include time/worksheet · Group by student, · Task · Duration (Hours) · Complexity (high-medium-low) · ADD a total row for each individual that shows their TOTAL and grand total that shows the GRAND TOTAL of hours for the whole team. Include ALL tasks/activities (e.g. meeting attendance, research/reading/training, etc) not just those spent on the deliverable. Marks for a submission (i.e. deliverable) may be deducted for missing/incomplete/incredible and far-fetched timesheets	Template on iLearn under Project in [Deliverables Certificate/Cover sheet]

All email correspondence MUST be via the MQ student account. other formats **may** be ignored. Please **include group number** in all correspondence.

C6 ESSENTIAL SPONSOR COMMUNICATION

In order to deliver valuable outputs within the semester, you are expected to follow agile processes including regular and frequent communication with your sponsor to obtain their feedback on your ideas and work. Your sponsor is not responsible for editing or assessing your documents. Make sure what is presented to them is professional and businesslike.

Deliverable 1: Discuss the Feasibility Report with your sponsor to ensure any misunderstandings are clarified, particularly concerning **their** problem, opportunity, mandates and assumptions. **After** updating based on your improved understanding and feedback from marking, discuss with the sponsor what parts of the Feasibility Report they are interested in and provide them with either an Executive Summary or the full report, as described in the Deliverable 1 section.

Deliverable 2: **Before** Deliverable 2 due date, you must obtain sponsor feedback to the Requirements/scoping document. This feedback should be included in the submission.

Deliverables 3 and 4: **Before** Increment 1 and 2 due dates, you should have demonstrated your prototype/minimum viable product (MVP) to your sponsor. In the Prototype section, your submission should include when/how the demonstration was done, the feedback you received and your response to it. You will lose marks for your prototype/MVP if you do not include information about the prototype in your Increment 1 and 2 documentation and regarding the meeting with your client, the feedback you received from them and how you will respond.

Deliverable 1 Details

Deliverables Certificate
Feasibility Report (4.5 marks) approx. 8-15 pages
Team Manual (2.5 marks) approx. 3-6 pages

Please liaise with your industry sponsor about their needs, problem or opportunity description and the solution sought. There may have been supporting documentation you received. The team should do further research to understand better the organisation, domain, problem, task, solution options, etc. Based on the information given and your research, discuss the problem, the activity/task, the domain and issues with your team so that you get a bigger picture of the context the problem fits in. Particularly consider the IT needs related to that activity and develop an initial proposed IT solution to one or more of the problems. You will need to describe the problem, alternative solutions (even if a specific solution has already been requested) and justify the solution you recommend.

Use the template on iLearn under resources for your Feasibility Report. In your Problem Identification section, explicitly identify the nature of your project: software, game, data science or cybersecurity. Please explain why it is that type of project. In the rare case that a project might cover more than one category, please provide a longer explanation. Assumptions –It is important to include assumptions. If those assumptions are wrong, then your plans and project will be compromised. Assumptions could include access to certain resources, scope of the system, support that will be provided, platform that will be used, etc. It is good to elaborate these so the sponsor can see them and correct any false assumptions.

After you receive your feedback, update your submission and discuss with the sponsor what sections of the Feasibility Report are of interest to them. As a minimum you need to provide them with a 1-page Executive Summary stating the sponsor's problem and opportunity.

Use the headings in the rubric below for your Team Manual. Discuss team structure and roles. Review together the ACS Code of Professional Conduct (see iLearn). Describe the team's values and how you will adhere to the Code. Clarify and justify the approaches and/or tools that will be/are being used for project management, team communication and to manage change or conflict.

Deliverable 1 marking scheme

Team _____

Problem Identification & Project Classification	0.5
Opportunities	0.5
Mandates	0.1
Current Situation	0.25
Tangible & Intanbile Benefits	0.5
Alternative Solutions	1.75
Recommended Solution	0.75
References/Abbreviations/Assumptions	0.15
TOTAL	**4.5**
Team Manual	
Team Organisation and Structure	0.5
Team Values & ACS Code of Professional Conduct	0.5

Project Management Approach and Tools	0.75
Communication Plan and Meeting Schedule	0.5
Change Management and Conflict Resolution	0.25
TOTAL	**2.5**

Deliverable 2 Details

Deliverables Certificate
Updated Team Manual (0.75 mark)
Project Plan (6.25 marks)
Requirements Document / Scoping Document (6 marks)

Project Plan

The Project Plan should not exceed (15) A4 sides, excluding appendices. See the marking rubric for content to be included and follow these headings and order.

Include the following in the *Project Plan*, in this order:
Statement of purpose [for plan] and scope [of project]
Risk Management:

- Development risks and their management, including a risk matrix/table that includes the risk, description, probability/likelihood, effect/consequence and risk level. Mitigation strategies must also be included.

Resource Management

- Project resources - people, hardware, software, other resources

Change Management

- Changes to requirements and scope
- Change to documents, code and data (i.e.version control).

Quality Management

- Discuss how quality will be monitored and assured

Schedule

- Using the project management tool of your choice (e.g. Trello, MS Project[1], ClickUp) include chart/screens that clearly show planned tasks, deliverables, time-line, resource/task allocations for the entire project.

Assumptions

- Assumptions around resources, availabilities, tools, techniques, standards, communication, expectations and anything else your project is based on and assumed to be true.

NOTE: For software development and games projects, we are using an agile software development life cycle (SDLC) because the requirements are often not fully known and will emerge as the project progresses based on the client's feedback. Data science projects should follow the intent of an agile data science process (http://www.datascience-pm.com/what-is-agile-data-science/).

Agile approaches are widely used in industry because agile methods allow all stakeholders to communicate and validate their ideas sooner. Examples of agile methodologies include SCRUM and Kanban. You will want to research what agile methodology best suits your project. Your project plan must be consistent with the deadlines and process life cycle you are using.

Marking scheme for project plan

Updated Team Manual	**0.75**
Team Organisation and Structure	0.25
Team Values & ACS Code of Professional Conduct	0.25
Communication Plan and Meeting Schedule	0.15
Conflict Resolution/Negotiation	0.1
Project Plan	
Statement of Purpose/Scope of Project/Description	0.5
Risk Management	1
Resource Management	0.75
Change Management	
Managing requirements and scope change	0.5
Version Control	0.5

Quality Management	0.75
Project Schedule	
Tasks/Deliverables/Process	1
Timeline	0.5
Resources Allocated	0.25
Assumptions	0.25
Standards/Templates/Appendices/Forms	0.25
Plan TOTAL	**6.25**

***** Continue on with the following pages for the Requirements / Scoping Document specifications and rubrics for this deliverable.***

Requirements Document/ Scoping Document

(6-20 pages)

A note to all groups

Overview: you must provide the requirements/scoping document to your client before the due date for their feedback. I will assume that what I receive has been confirmed by the client. You can expect to make changes to your document based on the client's review. It is best, though not essential, to send your draft document to the clients AND also arrange to meet them to obtain face-to-face comments. Your client may also like to see your project plan. Ask them. Make sure you clarify and confirm with them any resources you identified in the project plan that you expect from the client.

Include at the END of your Requirements/Scoping Document a page that describes (1 mark loss if not done):

Meeting date and time:
Feedback received
Team response/action points

Choose which of the following rubric sections to follow for your scoping document / requirements document:

- Cybersecurity
- Data Science
- Game Development
- Software (Default if unsure / not specified otherwise)

Details for CYBERSECURITY projects

Cybersecurity majors should have gained experience with a range of methods, tools and documentation in your previous units. Your client is also likely to have examples and preferred methods, tools and documentation to be used in your project. Please discuss these with your client and describe them in the section on "Deliverables Due" in your project plan. You will need to extend the project plan to a more comprehensive scoping document. Please refer to the marking rubric for content/sections in your scoping document.

The project plan put together the timeline and expected deliverables, now it is time to focus on what your project is about. This scoping document is intended as a way of identifying your analysis of the problem space.

The scoping document for Cybersecurity projects is aimed at outlining / communicating the following things for your team, the staff in The PACE unit, and your sponsors / stakeholders:

Part 1: analysis overview (intent / motivating factors)

1. The purpose and scope of activities being considered
    - What systems are being investigated / profiled / attached / guarded and why?
    - What kind of project are you undertaking?
        - An attacking team? (red team)
        - A defensive team? (blue team)
        - An overall security assessment analysis with recommendations?
        - …
2. The understanding (high level overview) of attacks and vulnerabilities that are going to be investigated
    - What attack vectors are being considered / reviewed / researched / attempted?
3. The types of adversaries that will be considered / profiled as part of any defensive or offensive strategy the team will be undertaking

- This is akin to grouping adversary profiles into different groupings (e.g. what are the different kinds of bad actors and what would their goals be?). This can help to consider how to guard against certain types of attackers and preventing their intended outcomes.
- What resources would each of these different adversary groups have at their disposal to achieve their outcomes?

4. The detection methods that are being considered / are available for investigation. You need to understand the important role of threat detection in any security system/solution. Even if your project is a simple application/plugin, if you don't pay attention to detection, then if an incident happens, you will have no clue how it happened. And even worse, your system/solution has no embedded mechanism to respond to threats before they become incidents. If this has not been part of your considerations in the final product, it is highly recommended to include Threat Detection function. For instance, in providing the proposed end-to-end encryption and storage solution, is there any chance that someone would be able to intercept or tamper the data transmission? If so, how would your solution respond to that? What do you do to prevent these kind of threats from happening again?
5. Monitoring of assumptions. Description of how the identified assumptions will be monitored in your project to ensure they have not been invalidated.
6. Document conventions being followed in this document (and what each of the sections cover – this is useful for anyone who doesn't know what to expect in this document)

Part 2: analysis activity details / specifics (mechanisms for investigation being considered)

1. Security feature extraction
   - This should expand on the points in (Part 1: 1). and (Part 1: 2).
   - What security features / weaknesses are initially being considered?

Part 2.1 focuses on your solution, i.e. what security features you will provide in the final product.

2. Attack / vulnerability analysis
   - This should expand on the points outlines in (Part 1: 2).
   - In Part 1, you outline attacks / vulnerabilities being investigated
   - Here in Part 2, is a chance to answer questions like
     - "What does this look like in the systems we are focusing on"?
     - Are there any indicators that might be a sign that this kind of attack / intrusion is currently occurring?
     - Are there any impacts / results which can show that such an attack / intrusion has occurred?

Part 2: 2 answers why you want to provide these security features by outlining the attacks/vulnerabilities that you have investigated in the current system (or in other solutions that you have investigated but didn't choose).

3. Defence countermeasures

- Identifying those points in Part 2: 2. … now it is time to consider what defence countermeasures that could be used / developed as part of any strategy.
    - If you are a red team project, then consider how these countermeasures might impact your approach.
    - If you are doing an overall security assessment / providing recommendations, then consider how any current defence mechanisms may impact your investigations / obscure other issues / limit the investigation capabilities.
- This is also a chance to have more details on the points outlined in Part 1: 4.

4. Architecture, Algorithms, and models
    - This is an expansion / elaboration on (Part 1: 4). In that section, you've outlined the potential detection methods being considered, here is where you elaborate on these in more detail.
5. Security testing
    - What are the different kinds of tests / assessments being considered? (this may change over time as more is discovered about the systems under focus).
6. Deployment / monitoring / reporting
    - Intended approach for deploying / running any tests or analysis scripts
    - How the monitoring of progress and results will be undertaken
    - What reporting / results / recommendations are being provided to the client.

<u>Details for DATA SCIENCE projects</u>

Data science projects will need to extend the above project plan to a more comprehensive scoping document by adding the extra sections as described below. Also, see the marking rubric. For Data Science projects, you will need to follow a project management method such as CRISP http://www.datascience-pm.com/crisp-dm-2/. For phase 1 (Business Understanding) you have already been asked to create a feasibility report and a project plan.

In the scoping document, your team is aiming to communicate the intended scope of work. However, these might change or evolve as you go through your project… and that is ok, because this is a living document and should be kept up to date / reviewed with stakeholders. In the scoping document, include each of the following sections:

1. Data Understanding: collect initial data; describe data; explore data; verify data quality
    - What are the initial data sources being used / provided?
    - What data collection methods are being used?
    - What is the assumed data quality / structure / completeness?

- What mechanisms will be undertaken to evaluate the above points? Will you have all the data that you need or will there need to be more data collection methods done during the project?

2. Data Preparation (generally, the most time-consuming phase): select data; clean data; synthesize data; integrate data; format data
   - Describe the intended data preparation activities for each of the data sources listed above
   - Describe any data processing pipelines that will need to be done. E.g. noise filtering, smoothing, running averages, error corrections, removing outliers, anonymising any datasets, feature engineering procedures…
   - If there needs to be different preparation pipelines for different models, then describe the processes needed for each model.
   - Also, describe why these steps would make the data more usable for the intended purposes.
   - What data storage mechanisms will need to be used during these processes?
3. Modelling: select modelling technique; generate test design; build model; assess model
   - Which modelling technique / techniques are being used / compared? Why? What is the intended outcome / question being considered and how does it relate to the chosen model?
   - How will the model effectiveness / validity be checked?
   - What test data needs to be provisioned / used for the above points?
4. Evaluation: evaluate results; review process; determine next steps
   - How will the results be reviewed? Are there expectations to be compared to, or is this more of an open exploration / unknown result? If so, how will these results be reviewed / validated?
   - What next steps or questions are being considered depending on the results / outcomes?
5. Deployment: plan deployment; plan monitoring and maintenance; produce final report; review project.
   - What is being given to the client? What training / documentation will be provided?

<u>Details for GAME projects</u>

A requirements document would have:

* High concept: A concise statement of what kind of a game it is, focusing on what the player does and what they experience.

* Core experience: What are the key experiential goals? Is it challenging and exciting? Relaxed and creative?

* Learning outcomes: For a serious game, what are the practical outcomes it is designed to produce? How does it change the player's understanding or ability? Does it motivate the player to do some task?

* Target audience: Usually expressed as a persona, with specific preferences, needs and requirements. E.g. "Malcolm plays puzzle games on his iPhone while travelling on the train to work". This gives you some constraints on who your game appeals to, where/when they are going to play it, for how long, etc.

* Target platform/interface: The device it is going to be played on, including considerations such as screen-size, controllers (mouse/keyboard/touchscreen/joypad), graphical capabilities, etc, and why these are appropriate for the kind of game you want to make. These can change as you develop, but you should always be working with a target platform/interface in mind.

Please follow the following format:

1…...Introduction
1.1 Document Convention/Intended audience
1.2 Game High Concept and Scope
1.3 Definitions, Acronyms, and Abbreviations
1.4 References
2 .....Overall Description
2.1 Core Experience
2.2 Target Platform
2.3 Target Audience and other User Classes and Characteristics
2.4 User/Training Documentation
3 .....Requirements
3.1 Intended Learning Outcomes
3.2 Core Gameplay Features
3.3 Other non-gameplay Functional Requirements
3.4 Design and Implementation Requirements/Constraints
3.5 Usability Requirements
3.6 Other Non-functional Requirements

<u>Details for SOFTWARE projects</u>

If your project concerns software (e.g. desktop or mobile application, website, game) you will need to create a Requirements Specification. If the product already exists (from a previous team or off the shelf product) you should reverse engineer your own requirements document to understand the product specifications.

System/Software Requirements Specification (SRS)/Requirements Document (6 marks)
The SRS should follow the IEEE standard below and use these headings:
1…...Introduction
1.1 Overview/Document Convention/Intended audience
1.2 Purpose (of software)
1.3 Scope – including a context diagram (Level 0 Data Flow Diagram)
1.4 Definitions, Acronyms, and Abbreviations
1.5 References
2 .....Overall Description
2.1 Product Perspective
2.2 Product Functions
2.3 User Classes and Characteristics
2.4 Operating Environment
2.5 User Documentation
3 .....Requirements
3.1 Functional Requirements
3.2 Design and Implementation Requirements/Constraints
3.3 Usability Requirements
3.4 Other Non-functional Requirements

Follow the above headings. For guidance refer to the IEEE SRS Standard. Please do not pay for the IEEE standard which you can access for free if you are logged on to Macquarie University Networks. You must be signed into the MQ campus network for it to recognise the institution. (this does not mean you can login to IEEE with your MQ OneID).
To access off campus:
https://login.simsrad.net.ocs.mq.edu.au/login?qurl=https://ieeexplore.ieee.org%2fdocument%2f6146379
The unit website includes some SRS examples. If your client has a format they would like you to use, please use that one and check that it covers the above sections.

Marking scheme for scoping / requirements sections

Marking Scheme for Scoping Document (Data Science and Cybersecurity Only)

For Data Science projects	For Cybersecurity projects	
Document convention/intended audience	Document convention/intended audience	0.25
Purpose/Project Scope & Context Diagram/	Purpose/Project Scope	0.25
Data Understanding – describe the data	Attacks/Vulnerability Understanding	0.5
Sources	Adversaries	0.5
Capture	Attacks/Vulnerability Detection	0.5
Storage/Environment	Monitoring of Assumptions	0.25
Data quality	Data quality	0.5
Follows Format Specified/Well-presented	Follows Format Specified/Well-presented	0.25
Data Preparation - cleaning needed	Data Preparation	0.25
Integration needed	Data (packages, logs, …) Collection	0.25
Formats needed	Security Feature Extraction	0.5
Data Exploration – Modelling Modelling technique selection	Attacks/Vulnerability Analysis	0.5
Model design constraints	Defence countermeasures	0.25
Model assessments needed	Architecture, Algorithms, Models,	0.25
Evaluation Requirements & Process	Security Testing (e.g., penetration testing)	0.5
Deployment – monitoring, reporting, review, training/support)	Deployment – monitoring, reporting, review, training/support, security polices)	0.5
Customer Feedback	Customer Feedback	PENALTY
TOTAL REQUIREMENTS		6

Marking Scheme for Requirements Document (Software / Web and Games Streams Only)

For Application/Website	For Games	
Document convention/intended audience	Document convention/intended audience	0.25
Purpose/Project Scope & Context Diagram/	High Concept	0.5
Definitions/References/Overview	Definitions/References/Overview	0.25

Follows Format Specified/Well-presented	Follows Format Specified/Well-presented	0.25
Product Perspective/Features/	Core Experience	0.25
User Classes and Characteristics/	Target Audience & other User Classes	0.5
Operating Environment/	Target Platform	0.25
User Documentation & Help	User Documentation/Training & Help	0.25
Functional Requirements (FR)"The system shall …"	Learning Outcomes: description and measurement	0.5
FR Uniquely identified	Other non-gameplay Functional Requirements	0.25
FR Organized/structured/grouped		0.25
FR Covers functionality	Core Gameplay Features	0.5
FR Unambiguous/testable/consistent		0.5
NonFR (security, performance, access, etc)	NonFR (security, performance, access, etc)	1
Design/Implementation Req (platform, language, etc)	Design/Implementation Req (platform, language, etc)	0.25
Usability Requirements (ease of use, training, etc)	Usability Requirements (ease of use, training, etc)	0.25
Customer Feedback	Customer Feedback	PENALTY
TOTAL REQUIREMENTS		6

Deliverables 3 & 4 (Increments 1 and 2)

Common sections for all groups:

Deliverables Certificate

Updated Project plan (with revision history + new "Handover Requirements" section)

Updated Requirements/ Scoping Document (with revision history)

Stream-specific sections (with different criteria):

Design Documentation

Testing Documentation

Prototype / Product (MVP)

+ deliverable 4 stream-specific document
(13 marks for Deliverable 3, 13 marks for Deliverable 4)

All Teams: Deliverable 3 & 4 submission instructions

PLEASE INCLUDE YOUR TEAM NUMBER in the file name.
The next submission contains multiple documents. However, for marking I would like one .pdf document in this order:

1. Revised Project Plan/Quality (complete plan, with your changes and revision history).
2. Revised Requirements/Scoping document (include complete document with your changes and revision history).
3. Analysis & Design Document including Assumptions at the end of this document.
4. MVP/Prototype Document
5. Testing Document
6. User Manual (**for Deliverable 4 only**)

Each of these documents will have their own Table of Contents - or if it suits your formatting better, you could make one TOC at the start.
You could create these pdf documents separately and then do a merge - if you have the right software. Include Deliverables Certificate as the first page.

Warning- Submission of documentation should be done by the due date via iLearn. Documentation must be consistent and relevant for the prototype demonstrated. The client typically is interested in the product, not the documentation. A poor prototype or redundant/inconsistent documentation will bring down the documentation mark.

Common sections for all groups

Add revision tables to the start of all documents just after the TOC to indicate the revision number, date, person and change to the previous version (be specific about what changed / why).

Updated Project Plan

Revised Project Plan (0.5 mark **Del 3 and 0.1 marks for Del 4**)

The following is required:

1. **Include the heading in the project plan: Handover requirements.** This section can be added after your schedule to explain in detail what the client needs. Handover will be different according to what the project is. It's about making sure the client gets what they were expecting and need before the team disappear. So any data, software, algorithms, scripts, test cases, testing tools/environments, outputs and documentation would be handed over together with training/manuals that lets them go forward with it - including reports or test cases showing current output and/or what is fully functional or not.
   For all projects, the point is to discuss with the client what they need at the end of the project. Make sure they have everything they need and know how to use what you have given them. There is no set format or requirement for the marker.
      a. For games and software projects, think about who will do play testing and/or acceptance testing, asking questions such as when will testing start, when and what devices/sites does the final product need to be deployed on, who will maintain the system, how will the system be maintained, what are the user manual/training manual/installation/installation wizard/administration parameters requirements, what database/file setup is required, what documentation is required.
2. **A revised and updated version of your project *schedule*,** that takes into account any schedule revisions, covering the balance of the project, that have arisen since you submitted the previous schedule. Anything else that has changed such as team structure, resource allocation, risk evaluation, change management processes, communication strategy, etc should also updated in the plan and quality manual and included in the revision history. Resubmit the entire updated document.

Updated Requirements / Scoping Document **(0.25 marks Del 3 and 0.1 marks for Del 4)**

Add a revision history table at the start of your document to clarify all changes made from Deliverable 2. No new sections are required. Please review feedback to Deliverable 2 submission and consider any project changes or sponsor feedback and update the document accordingly.

List of Assumptions

(0.25 marks Del 3 and 0.15 marks for Del 4)

Provide a list of assumptions (relevant to design, requirements, project plan or any of the documentation). This will help you think about them and whether they are reasonable and also help the marker/review team to understand why you have done certain things. Please review the

assumptions as a group before submission. A poor assumption will not be a valid reason for poor design decisions.

Prototype / MVP Document

(4.5 marks for increment 1-**Del 3** and 5.5 marks for increment 2-**Del 4**)

Before the due date/time you need to arrange a time with the client to demonstrate your system and receive feedback. The team must organise that meeting. All team members are expected to attend. A valid reason for non-attendance by a team member must be arranged before the meeting. When the demonstration/meeting is arranged please let the convenor know of the arrangement via the meetings thread on iLearn.

Under the heading "Prototype", include at least a page in your increment 1 and 2 documentation to present and discuss your prototype. Screen shots and how the product looks and works should be included.

Also include a subheading "Sponsor meeting, feedback and response to feedback" in the "Prototype section". Include details of the sponsor meeting and feedback. All sponsor feedback on any aspects of your project should be included in this section. Also clarify how the team have or will respond to that feedback.

If you have not been able to arrange a meeting before submission due to sponsor unavailability state that under this heading and when the meeting is scheduled. After the meeting, include the feedback and response in your next weekly report submitted on iLearn.

I will also be contacting the sponsor. The client will be asked to confirm the meeting and their feedback via a survey sent to them.

Prototype Feedback Adjustment

The marks for your prototype are made up of what you present in your document, the feedback reported and the client's feedback to me. (For Increment 1 marks are allocated as follows: 0 if no evidence or mention of a prototype, 1 if some evidence identified, 2 if evidence provided, 2.5 if brief, just screens and description or just feedback, 3 if shown with screens and feedback provided. 3.5 if shown in detail with feedback and responses to feedback, 4 if sponsor is happy 4.5 if sponsor is very happy.)

Analysis, Design + Testing Documentation for DATA SCIENCE projects

The main objective with the iterations is to build up as much of the data processing pipeline and describe the decisions / choices made - as well as how any choices and outputs will be evaluated. Someone else coming in late to the project should be able to use this deliverable to understand what the team is doing and where they are up to with their deliverables - including the rationale behind any design decisions.

Students choosing the data science stream can leverage what was built up in data science units (like COMP257 / COMP2200) and discuss what has been done in their project. Much like in the project portfolio and project presentation from COMP257 / COMP2200, the aim here is to communicate different aspects of the project implementation including the data, the goals, the options, the decisions, the choice of models and starting configurations, and the resources available.

For DATA SCIENCE projects, you should seek to establish a baseline model for the project – the simplest model you can think of for the problem. This would be a good deliverable as a MVP since it encourages you to have an end-to-end processing pipeline and to establish a baseline that you can then try to beat with their refined model in the subsequent increments.

Your documentation should include the data manipulation process with clear instructions on how to run them to turn the raw data into the input for modelling. One possible approach could be to load / maintain them in a GitHub / Bitbucket / Firebase repository with appropriate permissions shares. However, this is something that should be discussed with your sponsor and the convenor.

You should present the results of the modelling via evaluations in the form of a report. The report should outline the different models used and the results observed. Interpreting the results and discussing implications for the business questions that prompted the study are also a key part of this. In many cases this can be structured like a traditional scientific paper – material, methods, results, discussion.

Feature engineering: (deciding what you are looking for and how to go about it)

- From the data, are there any trends / ranges to look for?

e.g. if you are analysing time-based movement data, what accelerometer and gyroscope ranges / trends do you want to look for to identify someone that has fallen / collapsed?
e.g. if you are looking at financial data, what does an "upward sale trend" look like?
e.g. if you are looking at population data, or education rates, what "characteristics" would you want to look at and why?

- Describe what data characteristics are being looked for, and how your data pipeline is being processed to generate these features.
- Give each different "feature" or "characteristic" a name and then attribute some form of data ranges / statistical definition.

Solution Architecture: (choice of macro architecture / pipeline)

- Provide an overall description of each section of the pipeline including the data in and out of each "stage". The data details can be properly described in the "Detailed data descriptions" section.
- This would most likely be a more detailed version of the overall pipeline presented in the team's scoping document as some of the stages would now be implemented / finalised.
- Include any resources available / resource processing constraints to each of the sections in the pipeline (e.g. processing / timing limits)

Algorithms / models methods: (detail what is in each part of the solution architecture, including models used and initial conditions / config settings)

- selection of model... there are many different approaches: predictive, supervised / unsupervised, classifiers, ... which are going to be used? why? and why chose those over other approaches?
- Are there any settings needed (eg. in a KNN model, what is the number of the nearest nodes being used in the application of the classifier?)

Detailed Data descriptions:
Data being used, data being generated, data being stored, as well as any summaries and/or reports.

Model evaluation:

- how will any data / outputs be compared / tested / evaluated for correctness and accuracy?
- if you are choosing between models, how will the models be compared / contrasted to see which one has a better performance (e.g. if you are comparing different classifier models, on what basis are you comparing them? Detail each comparison.

Performance evaluation results:
what are the results of any tests run so far and what are the future planned tests for future iterations? (e.g. if this is Deliverable 3, what is planned for Deliverable 4? IF this is Deliverable 4, what is planned before the handover?)

Refer to the rubric / marking scheme ("All Teams: Deliverable 3 and 4 marking schemes") for sections to be included in the design and testing documentation.

Deliverable 4 (Only) stream-specific document: for Data Science Teams

Scripts / Model Execution (2 marks)

The *Scripts and Model Execution Documentation* should not exceed **twenty (20) A4 sides.** This is essentially a user or training manual for the one or more users of your MVP. It might include model parameters, how to set up the models, data sources/format, file locations, etc. The intended user/s may differ according to the project and there could be more than one type of user with different documentation needs.

The content of this document is something that you MUST discuss before **Del 3** when you talk to your client about what they expect at handover at the end of the project (see point 1 under updated project plan for Deliverable 3), ask them what their requirements are regarding support/maintenance. Find out what they need regarding training, help, installation, configuration, etc of your product/output.

Like all technical writing, the document should have a clear Table of Contents (TOC) and well-organised content. In addition to documenting your models and how to run them, this document could involve sections for different types of users (e.g. Admin and end users), installation guide, configuration settings, screen shots with example data, APIs, training, steps describing usage troubleshooting/where to get help, etc. Check you have page numbers, or your TOC will have a limited value.

Analysis, Design + Testing Documentation for CYBERSECURITY projects

Discuss with your client what they want to receive from you – that will be your MVP. It could be a *Penetration Testing Document* as in the example provided on iLearn. It could be a list of recommendations and vulnerabilities discovered. It could be an overview of attacks or defences tried (if you are doing red team / blue team approaches). Refer to the rubric for content and sections to be included in the analysis, design and testing documentation.

User stories describing the security / privacy attacks:

- What are the points of interest in the system and what attacks were considered / attempted?
- Be sure to include the sections / areas that these were applied.

Detection of intrusion / vulnerabilities:

- What did you find and how were you able to determine that the intrusion was successful / a vulnerability was discovered?

Defending solutions architecture:

- What was the architecture of the system under focus?
- Also, describe the mechanisms / process flow for your own investigations / attacks / defences (depending on the mode of your project). This could be represented as some form of "pen-testing architecture".

Non-technical concerns/approaches:

- Describe other cybersecurity aspects such as governance, human-factors (e.g. social engineering), processes, staff training, etc, that will impact on the security of IT infrastructure for the organisation.

Algorithms / security rules:

- Describe the pen-testing algorithms / approaches used (or the defending equivalent if you are focusing on that approach).

Data description / feature engineering:

- What data have been collected / used / provided?

Security test plan:

- This is a detailed overview of the entire testing intent for the project including: Timelines, approaches, resources required, times, people involved, approvals, reporting needs, overview and list of test cases. Refer to sample documentation in iLearn for general test plans that could be modified for security projects.

Testing reports:

- Detail the specifics for each test case what / when / where / how …
- What results have been found so far and what remains to be tested / followed up? This can be a comparison to your initial planned scope.
- What tests are yet to be completed / attempted?
- Are there any new tests / approaches being considered after having discovered more about the system under focus?

Be careful not to focus only on the technology side of the matter, rather than taking a holistic view. It may work well if the context and client can support this approach. However, in cybersecurity projects where the client is needing cybersecurity support but not wanting the team to conduct penetration testing or other technical changes to their current system, the team may need to think more broadly about cybersecurity and how to assist their client to make their systems/business more secure, now or in the future.

Note that it is quite common that sometimes clients don't even know what they want. Clients may have a security concern in mind, so they come with a security project, however, the reality is that they need a holistic IT solution instead. For example, if the organisation was having issues with their current web hosting service and are looking for a new solution provider, there

wouldn't be much value in conducting pen-testing. Instead the team should first focus on requirements engineering and help the client to reconsider the focus of their proposed project. Having said that, cybersecurity can still be the focus but there is a need of a holistic view. What are the major security concerns in this service? Such as users (admins without strong IT background, general public, school kids).

If you are not doing a pen-testing focused project, ensure in the testing section to provide explanation of any changes and discussions on alternatives, rather than pen-testing specifics. What can be tested? For example, the team can focus on the development of suitable metrics for measuring effectiveness of policies. - looking for
evidence of some structured approach to making the recommendations, such
as use of frameworks and standards, methodologies like threat modeling
and attack trees, threat catalogues like BSI Grundschutz, etc. which
would put the recommendations on a more formal footing than just "a list
of good ideas we brainstormed". ;)

Deliverable 4 (Only) stream-specific document: for Cybersecurity Teams
Recommendations / Monitoring Plan: Provide an early draft of recommendations / results reported so far based on findings. This should not exceed **twenty (20) A4 sides.** The intended recipients will differ according to the project and who is the target recipient of your MVP (e.g. the sponsor's organisation or one of their clients). The document should have a clear Table of Contents (TOC) and well-organised content. Check you have page numbers, or your TOC will have limited value.
The content of this document is something that you should discuss when you talk to your client about what they expect at handover at the end of the project (see point 1 above under updated project plan for Deliverable 3), ask them what documentation they require to support the MVP such as executing, monitoring, training, help, installation, configuration, etc of your product/output.

Analysis, Design + Testing Documentation for GAMES projects

Design deliverables could include:
Game design – for in-game functionality
* Storyboards: A power-point 'flip-book' showing frame-by-frame the core game mechanics, illustrating how they achieve the experience goals
* Art-book: concept art and/or images from other games or media that represent the art style you have in mind for the game
* UI mockups: showing the important information elements and how they are communicated to the player

* Paper prototypes: playable versions of the core gameplay implemented using cards, tokens, dice etc
* Software prototypes: lo-fi implementations of the core gameplay to demonstrate to "find the fun" — i.e. see whether the gameplay achieves the target experience. Ideally these should be the simplest thing needed to convey the experience. i.e. no art, no polish
* Design document including the following in-game and out-of-game aspects:
Game design: including storyboard, art-book, UI mockups with window navigation diagram to show game screen flow.
Game mechanics: how they function doing play, and how they are intended to achieve the experience goals.

Game scaffolding – for out-of-game functionality
Design of out-of-game features like leaderboards, setting up permissions, reporting, adding in new challenges / quiz questions.
To document these software features you may use activity diagrams for intended flows/algorithms, use case diagram (for different out-of-game features). You can refer to the software development section to see more about documenting software.

System Design Document
This will include the basic architecture of the system and the high-level strategy decisions. Use the following headings and explain if and why it is not relevant for your project, if that is the case. You need to include a description of the:

a) system architecture – including package diagram with description that is consistent with the system architecture
b) storage/persistent data strategy
c) any concurrent processes or data and how they will be handled if any exist
d) user interface strategy e.g. tracker, spoken dialogue, phone input, form/menu-based, GUI, etc
e) design decision choices and trade-offs

Data Definitions: Create a table showing what data is needed to be stored in files/database. Include data field name, type and example. For relational databases, for each table/file show the name of the field, the primary key (if applicable), the field type and an example of data in this field. E-R diagrams are recommended.

Game Testing deliverables:
* Playtesting should be done at all stages of development (hence the need for rapid prototyping).
* Playtesting should target specific questions about behaviour and experience: Does the player do what you expect them to do? Does the player feel what you want them to feel?

* Qualitative testing: observe play and take notes about what players do, interview or survey players to ask them what they feel.
* Quantitative testing: instrument the game to track relevant statistics: time of play, where do players succeed/fail at the game.
* Testing: other forms of testing to ensure testing of all functions e.g. adding players, updating scenarios, etc.

Test Spec (This can resemble the test plan and tests case templates from Software projects)
Acceptable documentation for games projects should include:

1. Test plans, including testing strategy (white/black box) and types (unit, system, penetration, regression, etc), detailed test schedule, testing tools and resources assigned, testing milestones and test deliverables and covering scheduling and resourcing of all testing processes.
2. Playtesting questions and results; qualitative testing; quantitative testing
3. For non-game elements include test case specifications:
    a) Identifier
    b) Test description
    c) Input specifications (e.g. actual input to be tested or link to be tested) (INCLUDE range of test values – possible valid and invalid input)
    d) Output specifications (e.g. expected outputs)

Deliverable 4 (Only) stream-specific document: for Games Teams

Help / Training Documentation (2 marks)

The *Help / Training Documentation* should not exceed **twenty (20) A4 sides.** The intended user/s will differ according to the project. For most games or software projects, the document should enable a moderately computer-literate user, initially completely unfamiliar with the system, to understand and fully utilise its functionality. Note there may be more than one type of user with different documentation needs. Training to use the game could also be provided as part of the game. This could involve special keys and screens with help and/training videos. Screenshots of help/training or the training video could be provided in place of or in addition to creation of a document. Please confirm alternatives to documents with the unit convenor well before any deadline. It is likely you need a combination of both documented (for installation, management and maintenance) and in-game support.

The document should have a clear Table of Contents (TOC) and well-organised content. Content will depend on the project but could involve sections for different types of users (e.g. Admin and end users), installation guide, configuration settings, screen shots with example data, APIs, training, steps describing usage troubleshooting/where to get help, etc. Check you have page numbers, or your TOC will have limited value.

As part of the discussion that you should have already had with your client about handover to them at the end of the project (see point 1 above under updated project plan for Deliverable 3), ask them what their requirements are regarding support/maintenance. Find out what they need regarding training, help, installation, configuration, etc of your product/output.

Analysis, Design + Testing Documentation for SOFTWARE / WEBSITE projects

For teams working on applications and websites follow these instructions.
ANALYSIS DOCUMENTATION
Analysis Documentation MUST include each of the following:

- Use Case Diagram
- Use Case Description (for each use case on the diagram)
- User Stories

Use Case Diagram
This is a graphic model showing the actors, the use cases and the relationships between them. One page should be sufficient. As a rule of thumb – if the description of a use case is very short (e.g. one or two steps only) or very similar to another use case, then consider combining the use cases and describing the alternate courses of action within that use case. Structure the use cases into logical groupings. Remember it's from the users' point of view – not the developers. See resource on iLearn on how to draw Use Case diagrams and common misunderstandings and errors.

Use Case Descriptions (one for each use case in the use case diagram)
A use case is a chunk of functionality. You must have a use case description for each use case. This will elaborate all the ways (uses and the steps to achieve them) in which people will achieve them. If you have many use cases, possibly some of your use cases are actually steps in a use case, not a chunk of functionality. e.g. if they just have one step (like "save", "submit", "authenticate"), they are probably a step in another use case and not their own use case.
When you start to write the use case steps, it will become more obvious whether you have identified separate chunks. Also, if you are repeating steps, then this is a candidate for reuse and should be an "includes" use case. Check you are not joining use cases together in sequences. Only includes, extends and generalisation are valid lines between use cases. E.g. "login" should not be joined to other use cases since you could login and then do nothing, or choose from a range of functions/use cases.
If you have 10 chunks of functionality then you need 10 use case descriptions. How will you know what to implement, if you don't know what triggers each use case, which user initiates, what the steps are, etc?

Please use the template on iLearn. Sub-use cases can be combined into one use case description. For example, if you have a use case Maintain Client with sub-use cases Adding Client, Deleting Client, Viewing Client or Modifying Client, you can just write one use case description for Maintain Client and describe the differences by branches in the use case steps.

User Stories

In line with agile practices, write user stories for your product. See the following link to get some tips: http://en.wikipedia.org/wiki/User_story. You can list these in a separate section or add the appropriate user story in your use case description.

DESIGN DOCUMENTATION

Design Documentation MUST include each of the following headings. If you choose not to include any of these items, provide an explanation under these heading.

1. System Design Document
2. User interface layouts / Report layouts (if needed)
3. Window Navigation Diagram (be sure to show backtracks / alternate navigation paths)
4. Data definitions
5. For every use case description create:

Activity Diagrams

OR

Sequence Diagrams (if OO programming language being used)

If classes/objects are being used:

6. Class Diagram
7. State Diagrams for objects with interesting behaviour

System Design Document

This will include the basic architecture of the system and the high level strategy decisions. Use the following headings and explain if and why it is not relevant for your project, if that is the case. You need to include a description of the:

f) system architecture – including package diagram with description that is consistent with the system architecture
g) storage/persistent data strategy
h) any concurrent processes or data and how they will be handled if any exist
i) user interface strategy e.g. tracker, spoken dialogue, phone input, form/menu-based, GUI, etc
j) design decision choices and trade-offs

User Interface Layouts

Show the actual screens or use a drawing package, system builder like e.g. Jbuilder, interface designer tool.

Report Layouts

If you are producing any reports (e.g. weekly sales report, daily summary, transaction listing), you need to design them. You need to consider things like report title, column/page headers, report totals/subtotals, fields, how many per page, etc). Different types of reports have different real estate (e.g. paper size or screen size). For example, if you needed to print a sales docket you need to plan what goes at top and bottom (e.g. name of company, ABN, store location, sales person's name, payment method, etc) how many spaces you have across and down and how to layout the docket so it will fit on the paper or screen (e.g. computer/mobile) to be used.

Window Navigation Diagram

Show how the different screens will link together (screen flow) and what triggers a transition from one screen to another. Activity diagrams are recommended.

Data Definitions

Create a table showing what data is needed to be stored in files/database. Include data field name, type and example. For relational databases, for each table/file show the name of the field, the primary key (if applicable), the field type and an example of data in this field. E-R diagrams are recommended.

Class Diagram

If you are using object-oriented programming please include a class diagram. Make sure any classes or methods on your sequence diagrams have been included on the class diagram. Method signatures should be given. The diagram must include all of the following:

- classes
- attributes
- associations
- inheritance and/or aggregation (if applicable)
- traversals
- multiplicities

Sequence Diagram

If you are using object-oriented programming please include one sequence diagram for each use case description. It should be possible to read the description and follow what is happening on the diagram. The objects and messages must be valid and shown on the Class Diagram.

Activity Diagrams

If you are NOT using object-oriented programming please include one activity diagram for each use case description. It should be possible to read the description and follow what is happening on the diagram.

State Diagrams

If you are using object-oriented programming include state diagrams for any objects that have interesting states or complex behaviours. A state diagram shows the life cycle of an object and thus all important objects can each be represented in their own State Diagram. However, you are required to consider the life cycle of each object in your system and to submit diagrams for those that have interesting states or complex behaviour. One way to measure if a state is interesting is to consider whether you need to test that state before performing a particular action or if the state changes after an action is performed. What is interesting will depend on the application. In most cases when an object is updated or printed (updated and printed can be states themselves but are generally not very meaningful) that will not change more interesting states such as paid/unpaid, married/single or for sale/sold.

Testing Documentation

Test Specifications (3 marks)

Acceptable documentation for software projects should include:

4. Test plans, including testing strategy (white/black box) and types (unit, system, penetration, regression, etc), detailed test schedule, testing tools and resources assigned, testing milestones and test deliverables and covering scheduling and resourcing of all testing processes.
5. Test case specifications:
    a) Identifier.
    b) Test description.
    c) Input specifications (e.g. actual input to be tested or link to be tested) (INCLUDE range of test values – possible valid and invalid input)
    d) Output specifications (e.g. expected outputs)

See templates for Test Plans and Test Cases on iLearn under resources on iLearn. Use of these templates is optional and suited to projects delivering a software product. If you prefer, you can use testing documents from previous units or other source.

Testing documents for games, cybersecurity and data science projects should contain content in accordance with the marking rubric.

Deliverable 4 stream-specific document: for Software Teams

User Manual Documentation (2 marks)

The *User Documentation* should not exceed **twenty (20) A4 sides.** The intended user/s will differ according to the project. For most games or software projects, the document should enable a moderately computer-literate user, initially completely unfamiliar with the system, to understand and fully utilise its functionality. Note there may be more than one type of user with different documentation needs.

The document should have a clear Table of Contents (TOC) and well-organised content. Content will depend on the project but could involve sections for different types of users (e.g. Admin and end users), installation guide, configuration settings, screen shots with example data, APIs, training, steps describing usage troubleshooting/where to get help, etc. Check you have page numbers, or your TOC will have limited value.

As part of the discussion that you should have already had with your client about handover to them at the end of the project (see point 1 above under updated project plan for Deliverable 3), ask them what their requirements are regarding support/maintenance. Find out what they need regarding training, help, installation, configuration, etc of your product/output.

All Teams: Deliverable 3 and 4 marking schemes

Revised Project Plan	D3: /0.50 D4: /0.25
Revised Scoping/Requirements	D3: /0.25 D4: /0.10
Not well-presented, spelling mistakes, language not businesslike	PENALTY
Deliverables certificate not included	PENALTY
List of Assumptions	D3: /0.25 D4: /0.15
Prototype/MVP/Project Output Document	D3: /4.50 D4: /5.50

Analysis and Design Documents D3 and D4 marking scheme							
Software	**D3 D4**	**Games**	**D3 D4**	**Data Science**	**D3 D4**	**Cyber**	**D3 D4**

Use case diagram	D3: /0.75 D4: /0.30	Art book	D3: /0.75 D4: /0.40	Feature Engineering (selection, construction)	D3: /1.25 D4: /0.80	User stories describing security/privacy attacks	D3: /1.00 D4: /0.50
Use Case descriptions and User Stories (Well-structured, sensible, complete)	D3: /1.00 D4: /0.40	Storyboards	D3: /1.00 D4: /0.40			Detection of intrusions / vulnerabilities	D3: /0.75 D4: /0.30
System Design Document	D3: /1.00 D4: /0.40	Design Document	D3: /1.00 D4: /0.50	Solution Architecture	D3: /1.25 D4: /0.40	Defending Solutions Architecture	D3: /0.75 D4: /0.35
Design – includes UML models, structure charts, Report Layouts	D3: /0.75 D4: /0.50	Functional Design - – includes diagrams and Report Layouts	D3: /0.75 D4: /0.40			Non-technical concerns/approaches	D3: /1.00 D4: /0.35
User Interface Layouts, Screen Navigation	D3: /0.50 D4: /0.20	UI mock-ups	D3: /0.80 D4: /0.20	Algorithm/Models/Methods	D3: /0.75 D4: /0.40	Algorithms/Security Rules	D3: /0.80 D4: /0.40
Data Definitions/Schemas/ER	D3: /0.50 D4: /0.20	Data Definition	D3: /0.20 D4: /0.10	Detailed Data descriptions	D3: /1.25 D4: /0.40	Data Description / Feature Engineering	D3: /0.20 D4: /0.10
User Manual **(D4 only)**	D3: /0.00 D4: /2.00	Help/Training **(D4 only)**	D3: /0.00 D4: /2.00	Scripts/ Model Execution **(D4 only)**	D3: /0.00 D4: /2.00	Recommendations / Monitoring Plan **(D4 only)**	D3: /0.00 D4: /2.00
D3 Subtotal: /4.50 D4 Subtotal: /4.00		**D3 Subtotal: /4.50 D4 Subtotal: /4.00**		**D3 Subtotal: /4.50 D4 Subtotal: /4.00**		**D3 Subtotal: /4.50 D4 Subtotal: /4.00**	
Test Specification							
Test plan	D3: /1.75 D4: /1.75	Playtesting & Functional	D3: /1.75 D4: /1.75	Model Evaluation	D3: /1.50 D4: /1.50	Security Test Plan	D3: /1.25 D4: /1.25

		Test Plan					
Test-case specifications	D3: /1.25 D4: /1.25	Functional Test-Cases & Playtesting results	D3: /1.25 D4: /1.25	Performance Evaluation results	D3: /1.50 D4: /1.50	Testing Reports	D3: /1.75 D4: /1.75
D3 Subtotal: /3.00 **D4 Subtotal: /3.00**		**D3 Subtotal: /3.00** **D4 Subtotal: /3.00**		**D3 Subtotal: /3.00** **D4 Subtotal: /3.00**		**D3 Subtotal: /3.00** **D4 Subtotal: /3.00**	

Deliverable 5 Details

Deliverables Certificate
Final Group Reflective Report (8 marks), approx. 7-20 pages

The sections in the Final Report are up to you but you may like to follow the system development life cycles and reflect on what your team learnt in each of the phases. Possible section headings could be:

1. Introduction
2. Project Planning
3. Requirements and Analysis
4. Design
5. Implementation
7. Learning Outcomes
8. Conclusion

This document is NOT a rehash or resubmission of the documents submitted previously. You should review all aspects of your project (information, outputs, documents, weekly reports, meeting minutes, etc) and reflect on what you did, why you did it, what worked and what should have been done differently.

You should aim to make this document self-contained. You may include information / diagrams / snippets from other documents to make your discussion more informative. Assume that someone was reading the document to discover what you had been up to for the past 3 months and could

also see why you drew the conclusions that you did. Don't assume they have read all the other documents.

There is no rubric or set structure. The key is reflection. Including individual reflections are also encouraged.

Deliverable 6 Details

Project Presentation (10 marks)

Each group will be given a time slot to present their project and demonstrate their system to the class, sponsors and academics in one of the lecture theatres. You must attend the whole session you are assigned to, not just your project presentation. Each team member must participate in the presentation, but not necessarily for the same length of time. The presentation mark will be based on your individual and group mark. Put your name on the bottom of each slide you present.

Your presentation should describe the problem being addressed, what you did to address the problem and how you went about coming up with solutions. Your presentation should include, assessment of (reflection on) the project and software process and future/outstanding work. Demonstrations should contain: Functionality of software, Non-functional qualities of software; examples of how to the system would be used, by whom and for what purpose.

Make sure that your system will run in the lecture room that will be used. Check with the convenor beforehand if in doubt. You may use the internet and computer in the lecture theatre, but if special software is needed you should demonstrate on your own machine.

As shown in the rubric below, a group mark will be given for presentation structure, communication of content and visual aspects of the presentation. An individual mark will be given for speaking/presentation skills. The four measures all overlap somewhat to make a coherent presentation. Presentation structure concerns the sequence in which the content was delivered and the flow of concepts. Communication of content is how is it delivered - via a demo, video, slides, just talking, how well do we understand what the project was about - you might have a nice structure but then it is not well described, the points are not relevant or they contradict. How understandable, logical and informative was the content communicated.

Grades/10 **(U)nsatisfactory (<5)** **(F)unctional (5-6)** **(P)roficient (7-8)** **(A)dvanced (9-10)**	Presentation Structure	Communication Of Content	Visual Aspects of Presentation	Presentation Skills
Group Number				
Individual Name				

Each item is marked out of 10. The group marks are averaged. The average of the group mark and individual mark will be the mark you receive.

Deliverable 7 Details

Handover to Client (10 marks from Client)

SOFTWARE & SUPPORTING MATERIALS
You must discuss with your client what they need to receive from you. Do they need a manual, install wizard (which platforms/OS)? What files/database needs to be set up, how can they run and maintain your system? This discussion should have started before submission of Deliverable 3 and continue.

When you have handed over what has been agreed, the sponsor must check it works on their setup (or you can check that for them). After they can confirm it is working, then they will provide a mark out of 10 for each individual student.

It is not sufficient to hand over a zip file and wish them all the best. You must make sure they can run your system – unless they agreed that they have technical people who will handle this. So you need to make sure your system is fully tested, well documented and bug free before handover. You should arrange a date for handover and allow your sponsor up to week to confirm

they can follow your instructions and use the system. Latest date for handover is final week of exam period. Leaving handover to the latest time may mean you receive an incomplete for the current semester as there may not be enough time for the sponsor to provide a mark and for it to be processed in your end of semester results.

Additionally for Deliverable 7, on iLearn upload a zip file that includes the latest version of all documents created over the semester. This is primarily an archive for the unit. You should update all documents that you will be handing over to your sponsor based on any feedback since the last deliverable. What is submitted will not be assessed, so teams may submit the latest previously submitted version but preferably the documents updated for the sponsor, i.e. the latest version, should be uploaded.

If your code or other files are too large or the structure is too complex to zip and upload, then you do not need to include them in this upload. For all projects, but particularly if you can't upload to iLearn, please ensure the client has been able to access the files and use them since there will be no other record of them that they can access in the future.

---

[1] For your Gantt chart, you can get MS Project through MQ site here: https://staff.mq.edu.au/intranet/science-and-engineering/services-and-resources/it-support-services/miscellaneous/microsoft-imagine/_nocache