SOFTENG 761 Project Version 4.4

Dr. Seyed Reza Shahamiri
Senior Lecturer in Software Engineering
Department of Electrical, Computer, and Software Engineering
Faculty of Engineering
The University of Auckland
Acknowledgment: We would like to thank Dr. Rashina Hoda for designing the initial version of this project.

Table of Contents

This document provides guidelines and best practice approaches to applying Agile and Lean practices to your SOFTENG761 project given time and academic constraints. It is imperative that you have understood the fundamentals of Agile, Scrum, XP, and Lean concepts from lectures and related resources (e.g., Scrum Primer, Scrum Guide, and extremeprogramming.org) before applying these guidelines and the SOFTENG761 Agile development model.

In this project, you will be working in teams of **seven** developers with a client that plays the role of **Product Owner (PO)** in which you will go through a **Scrum** software development process to produce a **software tool** for your PO. You will receive a list of available projects along with a short description of each project for you to bid. The teaching team will then allocate the projects to teams based on their interests, capability match, etc. Please note that not all teams get assigned to the project(s) of their choice and the teaching team makes the final allocation decisions.

During the project, you are required to carefully document all meetings, attendance, standups, member participation, task allocations and completion, development activities, git commits, etc. We may ask you to submit evidence of your and other team members' participation in the project and process.

The assessment of this project is holistic in which we consider both the product that you produce and the Scrum process you follow, as well as your individual contributions and team working skills. It is expected that you spend ten hours on this project per week. Your final product and weekly progress should be consistent with this ten hours of work per week, and overall, 70 hours per developer. We assess your skills to deliver **high-quality** software products while following a specific process in which you are expected to meet frequent deadlines among other deadlines of your other courses and, potentially, your other work commitments. Thus, you must not delay the work until the last weeks of this project and follow your plan precisely.

Please ensure that you read the Scrum Primer document before starting the project.

1 GETTING STARTED

1.1 Appoint Scrum Master

You need to appoint one of the developers to play the role of Scrum Master. If needed, other developers can step in to assist the Scrum Master. We recommend that you select a Scrum Master who ideally has more experience practicing agile since the Scrum Master is the team's agile coach. Otherwise, the Scrum Master is to quickly obtain the agile knowledge required and assist the team and PO in applying Scrum.

The Scrum Master needs to facilitate estimating story points factoring complexity, risk and uncertainty, and volume of each Product Backlog item/story, and maintain velocity (the number of story points the team marks as Done in each sprint, and average velocity for forecasting) and sprint burndown charts. **The velocity and burndown charts need to be reported during each sprint demo** explained in the following sections.

In addition, Scrum Masters **communicates the sprint forecast tables to the PO** after each sprint planning meeting. This table includes the Sprint Backlog items, their story points, and how many points per item are forecasted for completion per sprint day. These forecast tables are needed to be stored, documented, and submitted to the teaching team upon request.

Finlay, Scrum Masters are required to keep all necessary member participation evidence, including keeping an attendance sheet for all team meetings and standups per sprint, and complete the *Scrum Master's Member Participation Report* using this template indicating members participation in each sprint and hours spent by each member. This requires each team member to also document the hours spent on the project in each sprint, and indicate the items worked on via the e-tool (section 1.4), and supply the information to the Scrum Master. During each sprint demo, teams will present attendance and participation of each member, and Scrum Masters will submit the participation report to the teaching team towards the end of the project.
In order to compensate for the additional workload imposed on the Scrum Master, the Scrum Master's development participation should be 30% less than the other team members.
The Scrum Master can be changed during the project if the team decides to do so. However, this is not recommended as the change of Scrum Master may confuse the team and your PO.

1.2 Select a set of Agile Practices

There are several great practices to choose from, but you may not have the time and conditions to apply them fully, given the short time frame and the academic boundaries. We recommend the following core practices, which typically work well for SOFTENG761 teams:

- Release Planning
- Sprint Planning
- Standup
- Self-assignment
- Cross-functionality
- Work-in-Progress Limits
- Pair Programming
- Group Programming
- Sprint Review (including Demo)
- Retrospective
- Test-first development

1.3 Select a set of Artefacts

We recommend using these most common and highly useful artefacts:

- Product Backlog
- Sprint Backlog
- User Stories
- Tasks
- Release Plan
- Design Documents
- Scrum Board
- Burndown/burn-up charts

1.4 Setup Trello Board

All teams are required to use Trello as the Agile project management e-tool. Your Trello board must keep updated and show the updated progress of the project, and team members' task allocations and responsibilities at all times.
You may wish to maintain a physical scrum board **in addition** to the Trello board, or use other e-boards in addition to Trello.

1.5 Sprint Duration

Each sprint is one week (fixed).

1.6 Select Locations

- *Project work location*: Decide on where you plan to work together on the project. You are encouraged to use the lab in the timings that have been booked for SOFTENG761 and at other times when it is free. However, you may use other labs or spaces available to you.
- *PO meeting location*: Decide with your PO where they would like to conduct the sprint planning and review meetings. Some POs are happy to come over to the university, and you can book a room in advance. Others may invite you to visit them at their office or even attend virtual meetings. You may wish to use this opportunity to experience their work environment if convenient for you or request them to come over to the UoA instead (for local POs).

1.7 The Product Owner

Due to the academic nature of this project, some POs may also play the role of the client, end-user, or even one of the developers. Nevertheless, POs should not play the Scrum Master role.
A *Product Owner Guide* has been sent to all POs.

1.8 Continuous Scoping

Some of your POs may not be software engineers and hence not be familiar with the complexity of developing software systems. It is the team's responsibility to ensure the overall complexity of the product is sufficient for a 700 level software engineering course at the UoA and ten hours of weekly work. **You must help your PO scope the project down to ensure it can be done and tested properly, or scale it up to ensure it is complex enough and features rich**. Failing to do so will result in receiving a lower holistic score that affects all deliverables of this project. Your requirements engineering and negotiation skills are verified here.
The teaching team may also give you feedback during your weekly demos if the complexity of the product is lacking or you are over-committing.

1.9 User Interface and User Experience

Your product is expected to be developed professionally. If your product has a GUI, you need to ensure its aesthetic is professional and consistent, responsive, has proper user input validations and verifications, proper use of animations and transitions, etc.

1.10 Setup GitHub Classroom

All developers need to use GitHub Classroom to work on the project collaboratively. **The Scrum Master** to follow GitHub Classroom to setup your team and repo. Your team name must be the same as the name used on Canvas. Once you create your team, kindly inform your other team members to follow the link and join your Classroom team. Please do not follow the GitHub Classroom invitation link ***before*** your Scrum Master invites you to do so, and then ensure that you join the correct team.
You must use your GitHub account associated with your UoA UID. Please do not use your personal GitHub account unless it is linked to your UoA UDI, or you may face a penalty, or even a zero mark if we cannot confirm your project participation.
You are given admin access to your repo, so it is your responsibility to maintain it, and keep it private until the course is finished. You can add your PO to your repo, but no one outside the team, the teaching team, or from your PO's team should have access to the repo.

2 GitHub Usage Policy

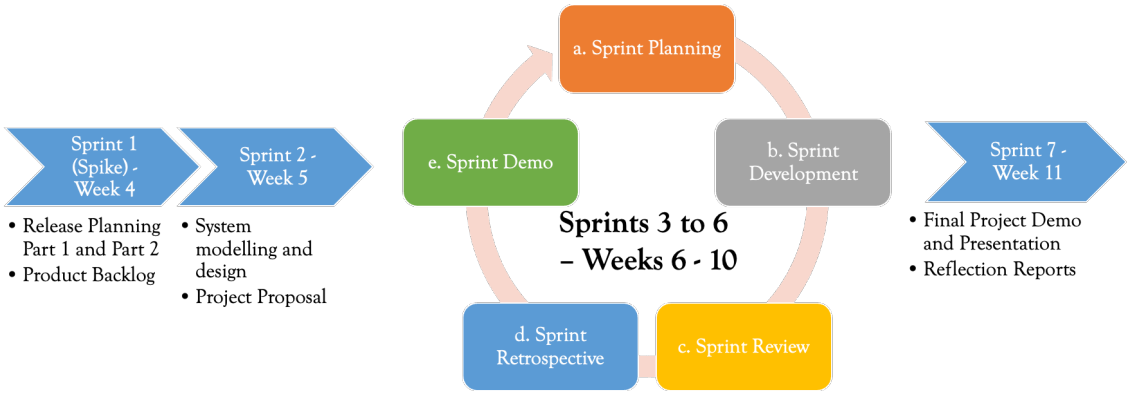
You are required to properly set GitHub Classroom and commit changes as you go through the project and sprints. We refer to each team member's commit and code participation history as part of the project assessment, meeting deadlines, team commitment, etc. However, your GitHub performance is not the only indication of your commitment and participation as we rely on other factors to identify your participation, such as your demo performances and peer reviews. Kindly pay attention to the followings:

- Do not commit all your changes at the end of the project, or you may receive a zero mark if you cannot provide us evidence of your ongoing commitment to the project and team.
- Do not push fake commits or add and then delete bulk code to increase your commit and code participation metrics. This may result in a zero mark.
- Lack of continuous and meaningful commit history from the beginning of the project or any attempt to manipulate your participation results in, at least, 30% deduction of your overall project mark if you fail to supply a proper justification.

Please note that the team repo is not an indication of submission and all deliverables must be submitted via Canvas since GitHub is an external system to our organisation.
Your repo must be private and only be shared with the teaching team plus your PO and no other participants. If your project is confidential, **you must not share anything about this project with individuals outside your team without written consent from us and your PO even after the course completion.**

3 Process

The figure below shows the process that you need to follow for this project. There are seven sprints with a fixed duration of one week each, as explained below. **We highly encourage you to adjust the start day of the sprints with the first lecture session of the week to ensure sprint demos are properly synchronized.**



3.1 Sprint 1

This sprint is the beginning of the Release Planning event and includes two parts:

3.1.1 Release Plan Part 1:

Description: A meeting (or more) to plan for the project scope and overall requirements. The team and customer (i.e., the PO) meet and discuss the overall vision and define 3 or 4 significant capabilities or modules of the system to be developed. These are then broken down into features and used to create a *Product Backlog*.
When: sometime in week 3 or early week 4
Who: Whole team + Product Owner
Outcomes:

- Clarification of project scope
- Definition of 3-4 major capabilities
- Product Backlog (prioritized list of all desired features described as user stories)

Success Tips:

- This will likely be the first or second meeting with your PO. Take a few minutes for everyone to introduce themselves briefly. Thanking them for proposing an awesome project does not hurt!
- Briefly describe your project approach and course deliverables, and schedule.
- Decide on things like:
- Regular meeting time and location;
- Setup your Trello board;
- Project platform and technologies,
- If hosting services are needed,
- How your PO can see and execute the code.

Note: it is preferred that your PO provides any servers needed in the project early, especially if your PO/client is from the UoA, as servers may typically take very long to setup by the university.

- Expectation management is at the heart of customer collaboration. It is a good idea to suggest a flexible approach where the team promises to do its best to accommodate as many features as possible during the seven sprints of this project (effectively five-six coding weeks) but cannot guarantee a complete solution. On the flipside, if the team finishes all requirements early, indicate that they will ask for more. The same has been conveyed to POs in the *Product Owner Guide*.

- The Product Backlog and user stories to be provided by you based on the information PO provides, as most POs are likely to have other full-time commitments. In our invitation to POs to propose a project and play this role, they were asked to dedicate one hour per sprint if they can – use this one hour wisely. However, your PO may be willing to spend more time on the project.

3.1.2 Release Plan Part 2:

Description: A meeting to discuss the Product Backlog and estimate effort to create a *Release Plan*. The team should discuss and estimate the size of all user stories roughly and the top few user stories with more accuracy (see DEEP acronym for creating a good product backlog.) Once all stories are estimated, divide them into sprint cycles of one week each. This initial schedule of which user stories will be implemented, tested, and delivered in which sprint cycle is the *Release Plan*.
When: sometime in week 3 or early week 4
Who: Whole team (+ PO)
Outcomes: Release Plan
Success Tips:

- Use planning poker to estimate user stories. If a story seems too big, divide it into 2-3 smaller ones (see INVEST acronym for creating good user stories).
- Depending on the number and size of stories, you may find it impossible to fit everything in this project's scope. If so, show that some lower priority stories are out of the project scope and will be attempted if the team gets to them. Similarly, if the scope looks too small and everything fits in 4-5 weeks, for example, indicate in the Release Plan that you'll ask the PO for more requirements.
- Time taken to set up the development environment, learning a new language or technology etc., should also be considered in the release plan.
- Remember, not all of your course commitment weekly hours are available for development work. Account for meetings and demos. Effectively each student should spend around 6-8 hours per week on development activities.
- Avoid planning too far ahead. The release plan is meant to give you a better idea of the project scope but is still meant to be flexible to change.

By the end of this sprint, you should complete both release planning events, which means the following should be ready:

1. The overview of your product backlog, including the major functionalities of the system to be developed, and their initial user stories draft.
2. Confirmed technology stack. **We highly recommend that you adopt an object-oriented approach and tools/languages that support it when possible.** During our holistic assessment, we evaluate how well object-oriented principles are adopted and realized.
3. Your Trello board.
4. Agreement on when, where, and how the team meets to work on the system and meet with your PO each week. We recommend that the team sets aside two days each week to work on the project full-time (to the total of approximately 10 hours a week inclusive of all SE761 activities). You can start these days with a brief virtual stand-up first thing in the morning and then work collaboratively on the project.
5. Confirming which agile practices the team adopts.

6. All required accounts (like GitHub, the e-management tool, etc.) are created, and the team (plus PO) has access. **All team members should have joined the GitHub repo by the end of sprint 1 or face late submission penalties.**

3.2 Sprint 2

In this sprint, you are expected to:

- design the architecture and domain model of your product,
- finalize the Sprint Planning event,
- setup the dev environment,
- setup the database and initial data models (if any),
- design mockups and wireframes, and the initial theme of the product (for GUI projects),
- initiate the development activities,
- develop a prototype to help you define the product requirements and scale the project,
- and submit the Project Proposal as explained in the deliverable section of this document.

By the end of the second sprint, you should:

1. Finalize and prioritize the product backlog.
2. Finalize all high-level system blueprints (class diagrams, use case diagrams, activity diagrams, system architecture, or anything else applicable to your project).
3. If your project needs GUI, the overall look and feel of the system should be confirmed and implemented, the navigation mechanism and map to be in place, and primary wireframes/mockups of the major items in product backlog done.
4. If your project requires a prototype, this has to be developed during this sprint. In this case, make sure to throw away the prototype before starting to develop the main system.
5. Submit the first deliverable of the project, *Project Proposal*.

3.3 Sprints 3 to 6 (both inclusive)

During these four sprints, you will follow the following events and continue implementing and verifying the software. Note that depending on the availability of the PO and the team, you may mix and modify the order of these events. For example, you can merge Sprint Review of the current sprint with Sprint Planning of the next sprint to make it easier for the PO to attend.

3.3.1 Sprint Planning

Description: Clarifying, prioritizing, and estimating user stories for development in the upcoming sprint.
When: Beginning of a sprint
Who: Whole team + Product Owner

3.3.2 Sprint Development

Description: Actual development and testing of user stories, including unit, functional and acceptance testing.

Note: commit individually to the repository, add useful comments, maintain continuous integration, and document individual member contributions per sprint. You will present what you did in Sprint Demos.
When: During a sprint
Who: Whole team (+ Product Owner)

3.3.3 Sprint Review (and Customer Demo)

Description: Reviewing the product with the PO at the end of the sprint along with a demo of the user stories implemented in the previous sprint. This is the time for the customer to conduct acceptance tests and provide feedback. Depending on the availability of the PO, teams often tend to have one meeting where the first half is used as a sprint review for the spring that just finished, and the second half is used as the Sprint Planning session for the upcoming sprint.
When: Towards the end of the sprint.
Who: Whole team + Product Owner

3.3.4 Sprint Retrospective

Description: Discussing how the sprint went for the team and ideas for improvement. This is the 'check' of the plan-do-check cycle.
Note: document planning meeting details.
When: At the end of the sprint
Who: Whole team

3.3.5 Sprint Demo

Description: During sprints 3 to 6, you will demo the Sprint Plan of the current sprint that just finished (not the upcoming sprint) and user stories/functionalities implemented in lecture time in front of the rest of the class. Prepare a short PowerPoint slide-deck if needed. Sprint Demos are marked and calculated as part of the second deliverable of the project. **You need to ensure all team members present at least once during these four sprints.**
Note: test that your code works in the lecture theatre and/or that the laptop/device you are using for the demo connects to the connectors in the lecture room in advance. If you need an HDMI adaptor, ensure that you have one.
When: As per class schedule
Duration: 5 minutes per project
Who: Two (rotating) from the team present to the teaching team and the class (if the project is not confidential). Can invite the PO. All team members must demo at least once during sprints 3 to 6; otherwise, no sprint demo mark will be given to the developers who did not present.

3.4 Sprint 7

This is the last week of the project, where teams wrap up any remaining refinements and/or fix defects and polish their documentations in preparation for final submission and presentation. You are required to deliver *Final Project Demo and Presentation* (all team members to participate) and submit *Reflection Report,* as stated in the deliverable section. **You must be prepared to have your presentation on the Monday of week 11.**

Please be advised that your Part IV project final report is due during this sprint; hence it is very important that you follow the process properly and do not use Sprint 7 as a full development sprint. You may only have time to make some minor changes before you submit the final deliverables.

4 DELIVERABLES

The tasks you perform in this project are **holistically assessed** via three sets of deliverables (and the four Sprint Demos) that totals to 80% of the course mark, a *Team Portfolio* that is marked as part of *Project Proposal* but needs to be submitted in advance, *Scrum Master's Member Participation Report*, and peer-reviews during your final demo (explained below).

To perform the holistic assessment, each deliverable is first assessed based on the criteria presented in this document. Then, our holistic assessment involves considering the entire agile process you adopted, the quality of the product you produced, its feature richness and level of complexity, whether you applied the best software engineering principles and practices, your team working skills, etc. Considering these factors, a holistic score for each team will be calculated and reflected back on all project deliverables so you should view all Project marks connected and as one mark. In other words, although each deliverable has its weight, your performance in one deliverable may still affect the others as part of our holistic assessment. We will also try to respect the peer-reviews (explained later) as much as possible when calculating the holistic scores.

All deliverables to be submitted by the Scrum Master on behalf of the team via Canvas. The deliverables include 40 marks allocated individually to each developer and 40 team marks. However, your final mark for the project may be affected by:

- your individual participation factor, as explained in section *4.4*, and,
- the overall quality of your product, and
- the complexity and feature richness of your product as explained in section 1.8, and
- if you fail to follow the process and weekly deadlines properly.

In this case, all your project deliverables' marks may be adjusted despite your performance in each deliverable. Since our holistic assessment requires all project deliverables submitted, project deliverable marks will be released at once.

If you feel you may not meet a deadline or face issues with your team or PO, or the project, please get in touch with us well in advance when we can help rectify the issue. We may not be able to effectively assist you if the project is concluded, which means your mark will be affected.

4.1 Team Portfolio (marked as part of Project Proposal)

This is a document you submit on week 2 of the course (and again as part of your *Project Proposal*) to introduce the team to your PO. It includes the following information:

- A group photo of the team
- Team name
- List of team members and their contact details
- Scrum Master

- A one-paragraph introduction for each team member covering what you study, any related work experience and technical/programming skills, and member's portrait.

4.2 Project Proposal (15 marks)

This document should provide the following information:

- Team Portfolio
- Who is the client?
- What the project is and its overall goal
- Explain the software functionalities via a high-level Use Case Diagram
- The high-Level architectural design of the system featuring your application, data (data storage/communication), and UI/presentation layers and how they relate to each other. Consider using standard design patterns for your software architecture, e.g., Model-View-Controller, where applicable. For more backend or API focused projects, consider modelling how the various components are meant to interact with each other, the messages being passed, and expected outcomes. The architecture should show the logical entities of the system inside each component/package rather than the technology stack.
- Conceptual Class Diagram featuring the classes in your application and their relationships. Strictly for non-object-oriented systems, you can supply activity diagrams of the top three primary user stories and use cases of the system, or a UML Component Diagram. Nevertheless, object-oriented systems must supply the class diagram. Consider applying design patterns and SOLID principles, and refer to SOFTENG306 instructions for class diagram.
- Prioritized Product Backlog (consider DEEP – Detailed, Estimated, Emergent, and Prioritized) with estimated story points for each item
- The definition of Done
- Which Agile practices you will use
- When and where to hold the Scrum Meetings
- When and where to meet with your PO
- The sprints start weekday
- The link to your Trello board, with access granted to the teaching team and your PO

Any hand-drawn models should be neat, legible, and scanned. No camera photos or screenshots will be accepted.

For this deliverable, we will consider the following criteria in addition to our holistic evaluation mentioned before:

Team Portfolio is complete as stated in section 4.1
Client information is supplied
Project goals are identified and presented
The use case diagram is properly shown the major software functionalities
The use case diagram does not have any syntax and semantic errors
Architectural design includes all major components of the system and makes sense
Product Backlog is presented considering DEEP and prioritized
Information about agile principles, scrum meetings location and time, etc. are presented and accurate
For Object-Oriented Systems
Class diagram properly captures all major software entities, and its syntax and semantic are accurate
Class associations are identified properly, with no unreachable class
For Non-Object-Oriented System
Activity diagrams for the top three user stories and use cases are supplied (in addition to the use case

diagram), and their syntax and semantic are accurate. OR A UML Component diagram is supplied, and its syntax and semantic are accurate

4.3 Project Demos and Presentations (35 marks)

This deliverable includes i) the Sprint Demos, ii) the team's final presentation and demo of the project and iii) the quality of the source code:

4.3.1 Sprint Demos (8 marks)

During sprints 3 to 6, the team is required to present (per sprint):

1. Sprint plan of the current sprint
2. Demo the items/stories marked Done as defined in the Project Proposal
3. Velocity history
4. The sprint burndown chart
5. Any significant changes in the velocity average should be explained.
6. Each member's sprint attendance and participation using the template.

Depending on which day of the week you start your sprints, you may be in different stages of the current sprint on the demo days. If your demo is scheduled towards the end of your current sprint, your demo should report on the current sprint progress and demo the items/stories marked Done during this sprint. However, if your demo is scheduled at the beginning of the sprint and you are yet to make sufficient demonstrable progress, you can demo the progress during the previous sprint and only present the current sprint plan with any progress made so far. We understand that your sprint start and end days may not match the lecture times; hence we are flexible as long as all sprints progress and Done items are demoed by the end of sprint 6.
For velocity history, you need to provide the current and previous velocities, and for each sprint provide the planned and actual velocities. Use a bar chart to present Velocity History similar to the chart below:

Only for Sprint 3 Demo, please spend an initial 2-3 minutes to introduce your PO, product, and explain the expected functionalities of your product via the Use Case Diagram you submit in the Project Proposal. You can also refer to the other diagrams and information included in the proposal if it helps to better introduce your project. This is because Sprint 3 is the first time other teams hear about your projects. Once the project is introduced, you can carry on with the typical 5 minutes sprint demo as explained above.

Note that each Sprint Demo must be done by two-three students only and other students to conduct the future demos. Each team is given **5 minutes** for each demo session.
You need to book the Sprint Demos each week in advance using the Sprint Demo Booking Sheet. Otherwise, late submission penalty will be applied.
By the end of sprint 6, all team members must demo at least once. For each sprint, ensure that both presenters are active and speak equally.
Teams are required to prepare few slides for each sprint demo. Toward the end of Sprint 6, Scrum Masters are required to zip all sprint demo slides and submit them on Canvas.

4.3.2 Final Demo and Presentation (12 marks where 10 marks to be allocated individually)

In this session, you are required to present and demo the complete project. Every team member must present equally in this session; otherwise, the individual presentation criteria (explained below) for that individual may be affected. Each team is given a 20 minute presentation slot in which you should present for 15-17 minutes and the remaining time for Q&A. You are encouraged to invite your PO to attend your final presentation.
You are required to provide the following information during your final presentation:

1. Introducing the team and the client
2. What the project is
3. The product backlog with estimated and actual story points
4. Final system blueprints (such as class diagrams, use case diagrams, architecture, or other diagrams that you see fit)
5. Complete demo of the software produced
6. What items from product and sprint backlogs are not completed and why
7. Explain the agile process and practices you adopted
8. The velocity of each sprint
9. Show how the team spent a minimum of (team size x 10 x 7) hours on the project
10. Show how each team member's minimum 70 hours were spent (minus lecture times if attended)
11. Present the Scrum Master's Member Participation Report. Items 9, 10, and 11 should be consistent.

You will be assessed based on both your team presentation performance and individual performance, and the code demo as follows (in addition to the holistic assessment of the project):

Team Presentation and Code Demo Criteria
All of the required information is supplied, and the system is demoed successfully.
Individual Presentation Criteria
Speaker maintains good eye contact with the audience and is appropriately animated (e.g., gestures, moving around, etc.), and is properly dressed.
Speaker uses a clear, audible voice.
The speaker's presentation time is similar to other team members.
Information was well communicated, and good language skills and pronunciation are used.

Note your Individual Presentation Mark will be based on your performance in both "Sprint Demos" and "Final Demo and Presentation".

You are required to provide a PowerPoint during your presentation, and the Scrum Master to submit the PowerPoint on Canvas as well. You may receive a zero for this component if the PowerPoint slides are not submitted on Canvas on time.
Judging and Peer-Review: All developers are required to judge all final demos during the official lecture hours, and complete a Final Demo Peer-Review Form for each team separately (one form for each team). The form must be completed immediately after each demo is done and submissions outside the lecture hours will be deleted. Your UoA UID and timestamp are captured when you submit the form so please ensure that you logged in using your uid@aucklanduni.ac.nz Google account. We are expecting at least five submissions per developer. Your attendance during the final demos during the official lecture times will be recorded. Lack of participation or attendance in judging your peers results in 30% penalty to your Project mark.
Please ensure to book your final demo using Final Demo and Presentation Booking Sheet. It is pertinent to note that all students' attendance is expected for the demos that happen during the formal lecture hours. However, only teams presenting outside the lecture hours are required to attend and complete the peer reviews.

4.3.3 Code Quality and Professionalism (15 marks)

The project code will be assessed based on how professionally it is written, the GUI designed (if applicable), the UX of the system, and whether the best software engineering practices (such as design patterns, design principles, clean code, etc.) were adopted when implementing the software. For machine learning-based projects, best ML practices (such as cross-validation, using validation data during training, check for overfitting, capturing proper performance criteria, etc.) can be considered as well.
Please ensure that you supply a document with proper instructions to help us set our testing platform and run your code. You may receive a zero for this deliverable if we cannot execute your software. We may also come back to you if we need further assistance.
For this deliverable, we will identify proper criteria based on the nature of the project. For example, the following criteria will be considered for a typical project with proper GUI (in addition to the holistic assessments):

The GUI. For example, well-designed GUI, appropriate use of colours respective to the nature of the project, UI Control sizes, navigation mechanism, UI responsiveness, user interactions and user input exception handlings, animations and transitions, etc.
Code quality. For example, components/files/classes are structured properly, using descriptive identifiers, SOLID design principles and design patterns are considered properly, proper usage of class interfaces, code properly commented, automated tests are designed, exception handling, etc.
Professionalism. For example, readme file is provided, readme file has sufficient information to setup and run the application, deployment information is provided, user manual is provided (if applicable).

While code quality has its own weight, low code quality will have a major effect on the entire Project mark as it has a profound impact on compiling the holistic score for each team.
Scrum Masters, please ensure that you submit your **final code** on Canvas. Compress everything as one file.

4.4 Reflection Report (30 individual mark)

This report consists of two main components: 1) individual reflection on the agile process and tools you adopted in the project, and 2) a participation report compiled by each team member**. This deliverable is submitted individually by each member and will not be shared with other team members ensuring the confidentiality of the peer**

reviews. However, we may ask your PO to verify the participation report and ratings you provide.
Each team member, including Scrum Masters, to complete the review forms (links are provided below). You can edit your responses before the due date (check Canvas). However, any submission or modification after the due date will impose a late submission penalty. Additionally, Scrum Master will submit the Scrum Master's Member Participation Report too.
You must use your UoA account to complete the forms. Any submission via private or non-UoA accounts will be deleted.

4.4.1 Individual Reflection (5 marks) – Form Link

Each team member to complete Individual Reflection Form explaining what worked well, what did not work well, what you did to rectify the issues, and what you would do differently. You are required to explain your agile experience during the project. All questions must be answered.

4.4.2 Participation Report and Peer-Review (25 marks) – Form Link

Each team member to complete one Participation Report and Peer-Review for **each team member** explaining:

1. the roles of the member played during the project and the tasks completed,
2. attendance during each meeting and how much the team member participated in those meetings and standups,
3. whether each team member completed the tasks allocated and met the sprint deadlines, and
4. the attitude and professionalism of the member.

You are required to complete one form per team member. For example, if your team size is seven, every team member must supply six peer-review forms, one for each member, excluding yourself. Using the form, you will rate each other concerning the criteria mentioned above and performance during the project. From the peer-reviews we receive, we will compile each developer's participation factor (0-100%) and will verify it against your demo performances, git contributions, the quality and complexity level of your product, etc.
If the participation factor of a developer is considerably lower than others, the developer's holistic assessment may be affected, as explained before, and may receive a fraction of the overall project mark. Please be advised that the peer-review ratings are not the only indication that we consider, and our assessment may be different from your peers to ensure the integrity and fairness of our evaluations.
Important: For teams that indicate excellent participation for all team members, we expect to see agile processes were followed entirely and correctly, all due dates and deadlines were met, and the product produced exceeds expectations in terms of product features, code quality, and GUI (if applicable), and other process and product related factors. Otherwise, the participation reports may be considered invalid, in which case we will entirely rely on other factors to compile each members participation factor. Thus, the team is required to carefully document all evidence and supply them if requested to the teaching team. **Scrum Masters are required to keep all necessary evidence, including the member participation report.** Scrum Masters need to submit the Scrum Master's Member Participation Report using the template on Canvas. Failure to supply the report (or the evidence if needed) may impose a significant mark penalty

on every team member. It is worth noting that Scrum Masters also need to complete the Participation Report and Peer-Review form for each member in addition to submitting the Scrum Master's Member Participation Report on Canvas.

5 PROJECT SCHEDULE

The project schedule is provided below. The due dates are supplied on Canvas.

	Date	To Do	Deliverables
Week 1	21 July	Formation and confirmation of teams (consider enrolment changes too)	Teams
Week 2	28 July	Bid for projects, submit the team portfolios with photos, and confirmation of Scrum Masters – this will be sent to your Product Owners	Team Portfolios – Friday
Week 3	4 August	Project Allocation (by the teaching team) Submission of team portfolios to Product Owners (by the teaching team) Students to sign the Project Agreement form and submit it on Canvas	Project Agreements
Week 4	5 August	Sprint 1	
Week 5	12 August	Sprint 2	Project Proposal (Due Friday)
Week 6	19 August	Sprint 3	Sprint 3 Demo
Study Break			
Week 7	Sep 9	Sprint 4	Sprint 4 Demo
Week 8	Sep 16	Sprint 5	Sprint 5 Demo
Week 9	Sep 23	System's Week – No Course work	
Week 10	Sep 30	Sprint 6	Sprint 6 Demo Submit all Sprint Demo slides
Week 11	October 7	Sprint 7	Final Project Demo and Presentation (and code submission on Sunday) Reflection Report – Sunday Scrum Master's Member Participation Report - Sunday

6 LINKS

activity diagrams
Component Diagram
extremeprogramming.org
Final Demo and Presentation Booking Sheet
Final Demo Peer-Review Form
GitHub Classroom
Individual Reflection Form
Participation Report and Peer-Review
Scrum Guide
Scrum Primer
Scrum Master's Member Participation Report Template
sprint burndown charts

Sprint Demo Booking Sheet
story points
Trello

7 FEEDBACK

Please submit your feedback via email to reza.shahamiri@auckland.ac.nz

8 FAQ

1. Should we keep our GitHub repo updated with the final version of the code?
   Yes, your repo and the code you submit on Canvas must be identical.
2. What if we make changes to the product and the final code submitted on Canvas?
   It is fine that your demoed product and final submission to be slightly different as you may use the feedback received during the demo to polish your code.
3. How detailed should the use case diagram be?
   Use cases capture the primary functionalities of the system. The level of detail you provide is with respect to the complexity and the context of the system. There also multiple levels of use case diagrams in which detailed use case diagrams can capture more detailed operations of the system, as actors see them. Here you only need to provide the high-level use case diagram. You need to act as actors with the help of your PO to decide what functionalities are major and important.
4. I missed a due date. Do any penalties apply?
   Unless you have a reasonable and acceptable reason for missing due dates, yes, penalties will apply. This applies to all deliverable due dates and other due dates such as team formations, git repo setups, etc. Being able to deliver on time and follow instructions are the primary objective of this course in which we closely monitor in our holistic assessment approach as both product's features and quality, and the process in which you follow will be assessed.

   The late penalty submission is up to 30%.
5. Can I change my team?
   Changing your team after teams are setup creates a ripple effect. We can only accept changes to teams before Team Portfolios are submitted, *and* providing that both teams agree to the change. Otherwise, changes in teams are not possible. It is important that you develop skills to work with others who you do not know closely since you usually do not get to select your partner in a real professional environment. Your team working skills will be assessed in this project as well.
6. What is the role of our assigned POs?
   POs were given a guide in which we asked them to set aside one hour each week for your projects. As such, they probably don't have the resources to both attend the planning sessions of each sprint and design the product backlogs and user stories. Furthermore, product owners may not be software engineers so the terminologies that you use may not be familiar for them. Their primary responsibility is to prioritise the product backlog, in consultation with the team, and to bring requirements. We expect that you capture and compile product backlog yourselves based on the information supplied by your PO.

Additionally, POs can be end users, clients, and/or stakeholders. For some projects they may want to be more involved but for others you may not even get the one hour weekly we originally requested. In this case, it's better that you think of them as your client and the dev team as the contractor. Hence, you need to adjust the Scrum events and practices around your product owner's availability.

7. How do we allocate the tasks between the team members?
   Teams are self-managed. There is no project manager role and it is everyone's responsibility to get involve, volunteer, and complete the project.
8. Can we change the Sprint Demo schedules?
   It is strictly not possible to change the project schedule as it goes against primary Scrum principles with respect to fixed sprints.
9. Can we change the release plan and user stories after the release plan is finalized in Sprint 1?
   Initially, your release plan includes your overall plan in which you estimate, roughly, when user stories will be implemented. However, as you progress in your project and more information becomes available, and changes to user stories and product backlog occur, you will adjust your sprint plans accordingly. Product backlog keeps changing during the SDLC so does your sprint plan.
10. Are we following Scrum as it is theorized?
    In some extend yes and other extend no. In practice, agile practitioners adjust agile principles the way best work for them and their organization. In this course we cannot completely simulate these principles the way they are prescribed due to the academic nature of the paper, other commitments you and your PO have, and some POs may not be software engineers. Nonetheless, we highly advise that you adopt the agile principles mentioned in section 1.
11. What are the compulsory Scrum practices we must follow?
    You are free to customize the Scrum process to suit the team's (and your POs) context and preferences. You must however follow some basic Scrum practices which will be evaluated. These include:
    - weekly sprints
    - sprint planning meetings
    - daily/frequent standup meetings
    - composing user stories and associated acceptance criteria
    - creating and maintaining product and sprint backlogs
    - sprint review and retrospective meetings.
12. What kind of client information is expected to be added to the Project Proposal?
    For UoA POs, include information about their department and their research that can be find via their profile. For external POs, information about their organization could be helpful, if it's available.
13. Do we include UI mockups in the Project Proposal?
    For GUI based projects, yes, you need to include them in the proposal. The purpose of the proposal is to provide a clear picture of the project so any information that better helps us to understand your project is welcome. These mockups will help us to better understand the presentation/UI components of your project. Also, the mockups should reflect the final product including the theme and colours you intend to use. However, it is acceptable if the team and PO decide to change the

design afterwards.

14. What agile practices should we mention in the proposal?
    It is required to mention which practices the team's planning to apply. We have a list of recommendations in this document in section 1.2, but you can add/remove any practice to the list to customize the process alongside your team and PO. Having said that, the practices you apply should be consistent with the project and its process. For example, all teams should adopt release and sprint planning but pair programming and WIP are optional.

15. What tools can we use to design the diagrams?
    You are free to use any tool you prefer. Some recommendations are below:

    - UML Use Case Diagrams - Lucidchart
    - What is a Use Case Diagram - Visual Paradigm
    - UML Use Case Diagram - Java At Point
    - UML Class Diagram Tutorial - Visual Paradigm
    - UML Class Diagrams - Java At Point
    - UML Class Diagrams - Lucidchart
    - Regarding GUI design mocks, you can use Figma, inside which you can create a Prototype': an interactive wireframe that simulates how it will flow on the app. There is also a plugin for Material Design icons.

16. Should we proceed with sprints during the system's week?
    During the System's Week, all activities with respect to any Part IV course are paused. You should only concentrate on System's Week.

17. Do we need to submit evidence of workload?
    You don't need to supply any evidence unless we come back to you and ask you specifically.

18. Do we need to introduce the team and project in each sprint?
    During the first demo please use two minutes to introduce the project, team, and product owner. Then, the remaining three minutes to briefly walkthrough your progress from the beginning of your project, and finally your Sprint 3 plan. However, for the following Sprints demos, detailed introductions will no longer be needed.

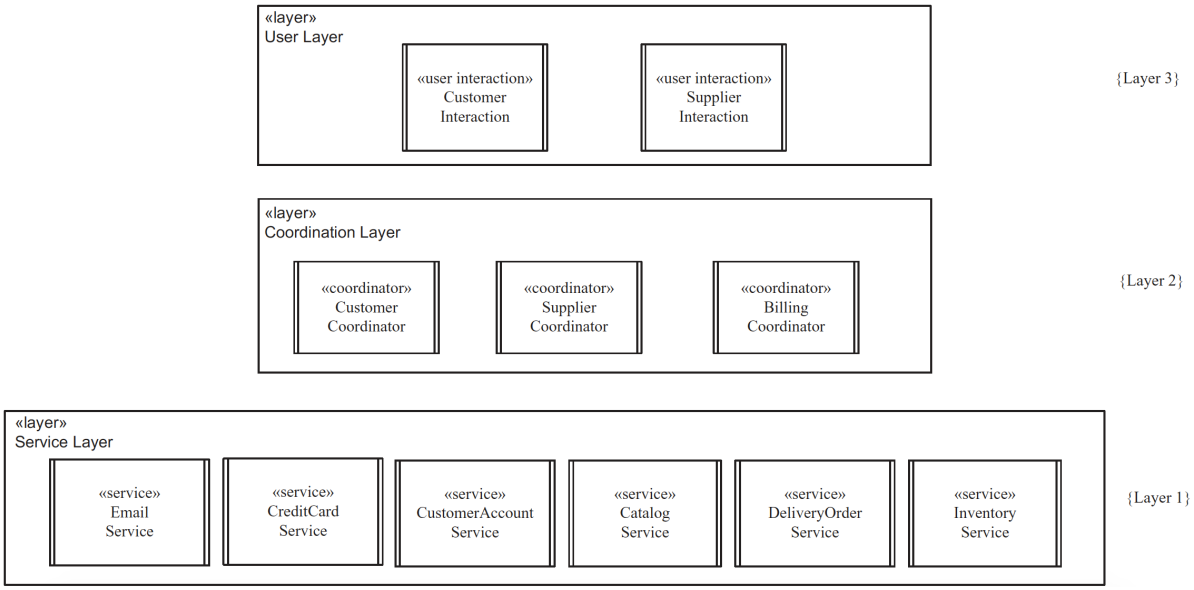
19. What if the study break start date is in the middle of our sprint?
    Please ensure that your teams complete all sprint 3 events and typical closure activities as usual before the study break starts. You need to ensure that the sprint 3 backlog is cleared before the start of the study break.

    In case you usually merge sprint reviews and planning in one session, it would be a good idea that you hold separate events for sprint 3 closure activities in week 6 with your PO, and sprint 4 planning after the break.

20. Should we continue with the project during the study break?
    There is no official requirement for teams to work on the project during the study break. It is up to the teams to decide if they want to progress the project during the break. If your project is on track, we recommend that you have some reset to be ready for weeks 11 and 12 in which you will have a heavy workload. If your project is not on track or not many features are identified/implemented, it makes sense to utilize the break to catch up. The study break period is especially helpful to assist team members who spent a good portion of their time to learn new languages or

technologies to catch up with their project commitments and be more productive.

21. Can we get Maintainer access to our GitHub repo:
    The answer is “it is complicated". You all singed the Project Agreement that is a legally binding document. Under the "Publication" clause, the University and you are not allowed to circulate any client's background material (and confidential information) without the client's consent. As such, the university needs to control access for three years, and granting Maintainer access is a risk to that clause for both you and the university. However, we may consider elevating the Scrum Master's role to Maintainer providing: 1) The client (who signed the agreement) emails us directly confirming there's no background material in the code repository and the client is happy to accept any risks involved with elevating the access, and 2) Scrum Masters email us confirming they will not add any new member to the repo and do not authorize any extra tool that may make the repo readable to anyone beyond the team, and 3) The team accepts any legal implications if the agreement is knowingly or not-knowingly breached.
22. How can we add/install GitHub tools now that we don’t have Maintainer access:
    You can send us your access requests to the tools that you need wrt your GitHub repo, and we authorize them if they don’t impose any risks to the Project Agreement. Please note these requests must be submitted by your Scrum Master to ensure proper discussions within the team has happened.
23. Can we add our PO to our GitHub repo?
    Yes, you can. You can send us their GitHub id and email address and we will add them to your repo.
24. Is it possible to get wikis for our repo?
    Yes, you can. You can place a “wiki” folder at the top level of your repo, and inside it include whatever Markdown files you want. You might need to call the default page README.md so that GitHub automatically renders it. You can then link to other Markdown files part of the wiki using standard Markdown notation.
25. Are there any samples of the previous years submissions available?
    We don't provide modal answers for project assessments. Your creativity is one of the primary items we assess.
26. The architecture requires to show application entities rather than technology stack. What does it mean?
    What we are interested is what the architectural components of your application are, how they are grouped/packaged, and their associations. For example, instead of using generic components like “React Components” or “Front End APIs”, you should indicate what logical entities you will implement in your “React Web App”. These entities are usually high-level. Ideally, you first need to identify your architectural pattern, for example, MVC, the State-Display-Logic-, etc., then show what the Model components/entities, views, etc. (if MVC selected) are inside the Model package. If you follow a full stack setup, then you can design separate architectures for your front-end (like your React Web App) and your backend (your Web APIs), and then link both architectures. For example, look at the architecture shown below for a typical online shopping system using a three-layered pattern. Notice that it shows high-level application components like Email components, Billing components, etc., rather than demonstrating the tech stack:



27. Can I use generative AI, such as ChatGPT, in this project?
    It is important that your work is original and you can explain it completely. You can use tools to help you complete the project better. However, if you are using code or other materials you did not produce, you must disclose them and explain from where and to what extent you have used them in your project. You are still required to be able to explain any material you submitted, even if inspired from tools or other individuals, and prove you understand the project and course materials well. Failure to explain your submissions and demonstrate sufficient technical understating or disclose the use of generative AI or other tools and sources that helped you in your project may be considered plagiarism.