IS212 (AY 2025-26, T1) – Scrum Project Instructions

Executive Summary

Overall Goal	***Deliver a quality software increment for a customer using scrum***
Weightage	35% of all course assessments
Deliverables & Due Dates	
Scrum Process Consultation (ungraded)	**During the Week 9 class:** ● Each team will have an informal (ungraded) ***conversation*** with their instructors about how they are conducting their sprints ● We recommend that teams start sprinting in Week 5 to get their 'hands dirty'. Figure out what's working and what's not, then use the Week 9 conversation to improve your process for the final sprint(s)
Software Release & Scrum Deliverables (graded)	**By 23:59 on Friday Week 12 (all sections):** ● You are to submit a working software release based on the customer briefing that (i) implements the **six core features** and (ii) was built by your team by following a scrum process ● Submit all **evidence of your scrum process** and **all code**, including **recordings of your final sprint meetings** ● Submissions must be done via eLearn; late penalties apply
Interactive Q&A Sessions with Instructors (graded)	**During your Week 13 Class:** ● Your team will meet your instructors for a closed-door 20-minute **interactive Q&A session** (i.e. no formal presentation) ● Your instructors will ask you questions about your scrum process and software increment ● You are encouraged to show evidence from your deliverables when you answer your instructors' questions
Peer Review (graded)	By **Friday Week 14, 11:59pm**: ● Complete your intra-team **peer evaluation** ● Non-participation in the project / peer review will result in penalties

Project Goal

The overall goal of this project is to give you hands-on experience of running an agile software project. It is designed to give you a sense of how real-world software developers collaborate both within a team and with an external customer (*the good, the bad, and the ugly*).

Based on the customer briefing provided to you in Week 1, and the subsequent clarification sessions, you are to nurture a product backlog of user stories, design a software system, and ultimately build a first release that increments **six core functionalities** (specified below).

You are to manage your coding activities using the **scrum process**[1] and are to document evidence of your various scrum activities and meetings throughout the entire project. Unlike other SCIS projects, our focus is on ***the process you are following***, so please do not hack a system together in Week 12 and expect to score well. We want to see a **sustainable** scrum process that is calibrated to the velocity of your team's developers *(and we understand that a 5 pax team typically has a lower velocity than a 6 pax team!)*. For example, it would be reasonable to conduct three 2-week sprints with around 6hrs/week coding work per team member. Exceeding 10hrs/week per person suggests something is going wrong.

Note that your final sprint **<u>must</u>** be scheduled/planned to end by Friday Week 12 **<u>at the latest</u>**, as your deliverables are all due that evening by 23:59.

Finally, we want to see evidence of the steps you have taken to ensure the quality of your code: steps such as test case design, continuous integration, and/or refactoring. It may be the case that your team starts sprinting before we teach these topics. If that's the case, it's fine to incorporate them only into your later sprint(s) when your sprinting process should be at its most refined.

First Release – Core Functionality

The customer briefing describes an entire system that is beyond the scope of a single-term project.

The customer has identified **some core feature areas and the expectations** that they would like to see addressed in your first release (i.e., your submission in Week 12). In particular:

Functionality Area	**Description**
User Authorisation and Authentication	Allows users to securely access the system using their login credentials. Depending on their role (e.g., staff, manager, director, HR), they will be granted access to different features and information. This ensures that data is protected and accessible only to the appropriate people.
Task Management	Users can create new tasks/subtasks (can be in a project or standalone), view their current tasks/subtasks, update task/subtask details, and update

[1] Base your scrum process on the version taught in class. Please consult your instructors before considering variations you might find on the web (e.g. scrum of scrums).

	their statuses. Tasks can include deadlines, notes, invited collaborators, and status tracking. Managers and above can assign tasks/subtasks to their staff, transferring ownership to them. This forms the foundation for personal and team productivity.
Task Grouping and Organisation	Users can create projects to house their tasks and subtasks. Collaborators can be invited to work on projects. This ensures that there are proper organisation and navigation for projects, as most staff work on multiple.
Deadline and Schedule Tracking	Enables users to attach due dates to tasks and view schedules across a timeline. Automated reminders help staff stay on track, and overdue items are highlighted for follow-up. Team members within the same project can also view schedules to monitor team load and prioritization.
Notification system (email or in-app)	The system sends alerts to notify users of approaching deadlines, new comments, task updates, or mentions from other team members. Notifications can be shown in the app or sent via email, helping users stay informed and responsive.
Report Generation and Exporting	Team members in a project can generate and export a report of its overall schedule, which shows at-a-glance the projected tasks, completed tasks, in-progress tasks, and tasks under review. This allows for future planning around deadlines and serves as a good estimate for how busy the team members are.

If you can complete these core features ahead of the Week 12 release, you can incorporate further features from the customer briefing as part of your final sprint(s). But you should not torpedo the quality of your sprinting process simply to fit in more features.

Note that even though the Week 12 release only requires working code for these **six** features, you may still want to consider features beyond the scope of the first release in your product backlog and C4 models.

Details on how to reach the customer for further clarifications have been conveyed separately to your section. DO NOT spam public Slack channels with such questions, as they will not be answered. **Note that each section has a different customer, so clarifications may differ slightly between sections. You should only consider the clarifications offered to your section.**

Deliverables – Working Code & Evidence of Scrum

It is important that you start collecting evidence of your team's sprinting process from the first sprint (Week 5) onwards, as this forms a significant part of your Week 12 submission. The quality of your scrum process (and thus the evidence you collect) weighs heavily when we grade your projects.

We encourage you to decide as a team how to manage your scrum project. You could, for example, manage your product and sprint backlogs using spreadsheets. Alternatively, you could explore tools such as Jira. The specific process you put together will determine the type of evidence you need to collect, e.g., PDF exports of Google Sheets vs. exports/screenshots from Jira.

Note that it is fine for your initial sprint to be a bit 'rough', as long as you show evidence of reflecting on this and taking steps to improve your subsequent sprints. We will view your final sprint as a **"showcase sprint"** that demonstrates your process at its best.

Your deliverables should be **submitted in a single .zip file**. The contents of your zip file should be **organised into numbered sub-folders** as per the following:

#1	Deliverable	Type
1	Your **full, final product backlog**, containing all your completed / pending user stories (if you use a tool like Jira, you need to export your backlog/stories for the submission)	**Document**
2	Your **system design**, conveyed using the C4 model; this should include some C4-Code models for *significant design decisions*	**Document(s)**
3	**Test cases** covering the **six** core features	**Document(s)**
4	Documents / screenshots from your **sprint meetings** (all sprints), for example: ● Sprint backlogs/goals/tasks at the start of each sprint ● Records of estimation activities ● Scrum task boards at the end of each sprint ● Burndown charts at the end of each sprint ● Any other relevant meeting documents, e.g., sprint retrospective boards **IMPORTANT:** for your final sprint only, you must additionally include recordings of your sprint planning, review, retro, and	**Documents & Videos**

	(one) daily standup. As these will be too large for eLearn, please **upload them to YouTube** as unlisted videos, and include links to the videos as part of your eLearn submission. (If you upload all meetings in a single video, please make sure to time-index it.)	
5	A **self-contained slide deck** summarising how your team addressed the rubrics of this course project. (Note that this will not be presented.) **IMPORTANT:** this must follow the slide template we will release in eLearn **by Week 10**.	**PDF Slide deck**
6	Your **software release** that increments the **six** core features ● Include all your working code ● Include a README file with clear instructions on how to run your software release ● You can use any programming language[2] (but no commercial / low code software – *we need to see tests!*) ● Please include any **test classes** containing unit/integration tests for your core features ● Include any **CI pipeline script** that you utilised	**Code**
7	A README file that contains a **link to your Git repository**; you must also ensure that both of your instructors have been granted access to the repository.	**README** **& GitHub access**

The following will take place **during** your Week 13 class. The specific timing will be conveyed to you by your instructors.

Deliverable	**Type**
● Your team will meet your instructors for a closed-door 20-minute **interactive Q&A session** (i.e. no formal presentation)	**Interactive Q&A**

[2] If you are NOT using Python/Flask (as in our labs), please include full installation instructions so that we can recreate your software environment.

● Your instructors will ask you questions about your scrum process and software increment (e.g. demo certain features) ● You are encouraged to show evidence from your deliverables when you answer your instructors' questions	

eLearn Submission Requirements & Late Penalties

1. All deliverables must be submitted as a **single .zip file** and should be organised in numbered sub-folders (1–8) according to the table above.

2. Please submit to your section's project 'assignment' on eLearn by 23:59 on Friday Week 12.

3. An intra-team peer review will be conducted in Week 14. There will be a penalty for those who are deemed to have not contributed after investigation by the teaching team. Having said that, please inform the teaching team **early** when issues arise within your team so that we can try to intervene before it is too late.

4. Late submissions are subject to heavy penalties, as below. Please submit early. We will grade only your latest submission.

within 1 hour (one second late is late)	10% deduction from the marks you deserved
each subsequent hour	Penalty will increase: 20%, 40%, 80%, and then finally 100%

Project Rubrics

Projects will be graded holistically according to the rubrics on the next page:

Unsatisfactory	Fair	Good	Excellent
● Team codes in a largely ad hoc manner; no discernible process or documentation ● Team did not show evidence of having used the estimation techniques covered in class for their user stories ● User stories somewhat follow the prescribed format but they were unclear and did not consider desired attributes of user stories ● System is documented with detailed specifications that might be invalidated by changes in-between sprints ● System is poorly designed or barely functions ● Minimal evidence of ensuring software quality (e.g., test cases, unit tests)	● Evidence of occasionally following a scrum process with limited documentation ● Estimation techniques are used in an arbitrary manner without justification on why a particular technique(s) is/are used ● User stories somewhat follow prescribed format but need further conversations to reflect user needs and desired attributes of user stories ● System is designed beyond necessary significant design decisions with specifications that introduce inflexibility during sprint development ● System exhibits some elements of good design and some features work ● Some testing is involved	● Team largely follows a scrum process supported by adequate documentation ● Estimation techniques are used in a consistent manner with adequate justification on why particular technique(s) is/are used ● User stories follow prescribed format and exhibit desired attributes but need further conversations to confirm user needs ● System is well modelled showing significant design decisions ● System is well-designed and solution works reasonably well ● System is well-covered by test cases and unit tests ● A basic CI pipeline is utilised when integrating code	● Team consistently follows an effective scrum process supported by comprehensive documentation ● Estimation techniques are used in a consistent manner with adequate justification on why particular technique(s) is/are used to produce valid and reasonable estimates ● User stories follow prescribed format and exhibit desired attributes with clear demonstration of user needs and acceptance criteria ● System is well-modelled showing significant design decisions that exhibit good practices of pattern reuse and design principles ● System is well-designed, code is modular, and solution can be demonstrated reliably ● System is comprehensively covered by traceable test cases, as well as code-level unit tests, and/or integration tests ● A comprehensive CI pipeline is used during code integration