Adelaide Damrau, Reagan Mullins, Stephen Maurer

Scrum Master Responsibilities:

First and foremost, from day one it is up to the SM to establish each of the team member's understanding of the process and purpose of Scrum.

- To ensure each Sprint event that takes place is a positive experience.
- Encourage your team members with a positive uplifting attitude.
- Keep up with the conversation, if it begins to drift off topic or to issues that are not related to your team's Sprint Goal, then it is up to you to redirect the conversation back on topic.
- Coach the team members in self-management and cross-functionality.
- Causing the removal of impediments to the Scrum Team's progress.
- Make sure everyone is getting the option to speak their opinion on what they feel should be done (Reminder: there is no "leader" in Scrum. You want this to be a smooth, agile, and an adapting process. Everyone is working at the same level towards the same goal to complete a Sprint one at a time. This requires a lot of communicating between all members: the Scrum Master, Product Owner, and Development Team).

You want your team to have a good understanding of:

- Scrum vocabulary (Agile, User Story, Sprint Goal, Done [Which, you are the team will define the definition of what done will look like for you all], Sprint Backlog, Grooming, etc).
- Along with what the roles entail (Scrum Master, Product Owner, and Development Team). It is best to explain all of these upfront, so you can ensure everyone understands their responsibilities.
- The order of steps to take to complete a Sprint.
- The purpose for each step in the Sprint (why you are doing each step and the purpose it holds, how long it should take, what is to be done in that time frame, etc.).

Adelaide Damrau, Reagan Mullins, Stephen Maurer

Sprint Steps:

Sprint Planning: Time <= 15 Minutes

Sprint Standup: Time <= 15 Minutes

Three Questions for the team: (Make sure everyone can see the answers for each person this is a part of communication. Google Docs are extremely helpful through all of this.)

What has each person completed?

What do they plan to do?

What issues are they running into, if any?

Backlog grooming: <= 30 Minutes

Sprint Review <= 15 Minutes

Sprint Retrospective: Time any time left.

Lessons Learned:

Make sure the group understands the importance of Source Control and are using it. (It's hard to work together if they do not all have access to the same code and resources.)

Make sure your sprint goals and tasks are well defined. Use Smart!

Specific	• Define what you expect • Determine who will do it • Detail accountability • Use action verbs, expressing physical or mental action, as much as possible • Provide enough detail - this depends on the objective but should be enough to be clear
Measurable	• Identify how you will know objective was accomplished – usually this means quantity but can also be quality (for instance, "80% of participants agree or strongly agree on the feedback form")
Attainable	• Make sure you have the time, manpower, resources, and authority to accomplish the objective • Consider if there may be factors beyond your control
Relevant	• The objective helps you meet the purpose of the grant • The objective is aligned with the Community Readiness Assessment scores
Time bound	• Specify when the objective should be completed • Include time-lined benchmarks for long-range goals and all objectives

Adelaide Damrau, Reagan Mullins, Stephen Maurer

Communication is super important and has the tools for it (Trello, Discord, Source Control doesn't matter if the team does not use them).

Make sure to have a "definition of done" for a task that includes making sure it is tested.

It might be a good idea to suggest having people work with each other on tasks that provide testing since two people are doing plus accountability. This is where "pair programming" comes into play. Pair programming is an agile software development technique in which two programmers work together at one workstation. One, the driver, writes code while the other, the observer or navigator, reviews each line of code as it is typed in. The two programmers switch roles frequently.

Make sure to create and follow coding standards (comment all throughout your code, and as you go to make it easier).

Make sure to stick to the sprint goal with your tasks.

<u>For the Sprint Retrospective:</u>

Have each member of the team answer these questions:

What were 3 things we have done well this Sprint?

What are 3 things we can improve on moving into the next Sprint?

Then have the team pick 1 or 2 of these and come up with a plan to improve on them. Understand that you will have multiple things to improve on as you are just now practicing 13the process of Scrum probably for the first time. It is important to collectively choose a couple of the "we could do better" things and improve them for the next Sprint. I recommend keeping an ongoing document that will capture each Sprint Retrospective. This makes it easy to reflect on where you have addressed issues and come next Sprint Retrospective you will also record your improvements.

Adelaide Damrau, Reagan Mullins, Stephen Maurer

Not everyone has the same experience will all technologies, e.g. Trello. When defining a team's standards, ensure everyone has an opportunity to voice their preference and knowledge level of different technologies. Also consider which technologies have better documentation.

Define UI standards
Define coding standards – some past classes provide documents on best coding and commenting practices, this is what we followed throughout our sprints.
Coding Architecture?
Prod vs Dev vs QA environment?
Languages used?
Technology stack?
Model to mobile?

<u>Ways to Exceed Expectations:</u>

- Work more than 6 hours a week
    - Our team met an hour before class every day to ensure we all hit the minimum 6 hours a week but usually we worked a little throughout the week too, exceeding expectations. This was also helpful to the team because we consistently worked on the project for a 3 hour block with all other team members, making it easier to work together in real time. Highly recommend this method, although it sucked getting up an extra hour early on class days, it paid off throughout the sprints.
- Proactively collaborate with each other
    - Working in time blocks with each other made it easier to implement peer programming, code reviews, and ask each other questions. Each team member consistently developed with at least one other team member and was involved in at least one instance of peer programming and a code review as both the reviewer and the reviewee. As a scrum master, encourage team members working on similar or the same task to implement peer programming. If you have extra time during development in class, that's the perfect time to hook a laptop to the tv and do some code reviews or testing.
- Follow the process (agile/scrum) and document thoroughly
    - As scrum master, I created a shared one-note at the beginning of sprint one to maintain documentation, coding/UI standards, and documentation of scrum activities, such as sprint planning tasks with estimated and actual time taken to complete them. A one-note was a great system for us because it allowed lots of tabs for new progress or documentation while allowing each team member permissions to edit and update their work while reviewing each other.
- Code/design documentation/readme & deployment instructions
    - The team consistently documented their code workflows, database diagrams, and web page designs in the one-note to make it easier to follow the development progress. Deployment instructions were made as the server deployment was set up so as not to forget any steps. The readme, however, was made as part of the last sprint using past documentation as a finalized and

simple form of project documentation. We decided to create the readme last because we kept needing to change it after daily progress. By doing it last and referencing the documentation, it saved time and was more efficient.

- Get feedback from people outside of class
    - Various team members have jobs in development and used their jobs as sources to help solve sprint problems. One team member, for instance, met with her mentor at work to discuss how to properly set up source control for the different environments for the team. Another team member reached out to a friend to get an outside option on the web page formatting and features. These are different examples of ways team members can get feedback from people outside of class, even reaching out to other professors is an easy way to get valuable feedback.
- Team has consistent code reviews, document, and testing
    - Throughout the sprints, because our team met early we had extra time for development during class, giving us opportunities to perform code reviews and testing.