# CS 4300/5300 Software Engineering

Common Sprint Requirements

Each sprint is two weeks long. Make sure that you work through both weeks of the sprint to achieve your tasks.

## 1 Sprint Planning

1. At the beginning of the sprint, meet with your group to do **backlog grooming** and **prioritization**. Mark stories that are MVPs (minimum viable product).
2. Begin the Sprint by looking over your backlog. Add stories or divide stories as needed—some of this may include playing Planning Poker in your team to determine points and need for division. Record the changes in GitHub.
3. Meet with your customer to identify the stories they want you to implement. Prioritize and confirm with them (if you can).
4. The number of stories that you will complete each sprint depends on the difficulty of the stories. Select as many stories as you hope to get done—if you don't meet your expectations, that's okay, but each person should have a story/stories to work on. You will balance the workload between each of the sprints. At least two larger stories should be implemented and fully tested each sprint. Pairs of students may work on features that are especially large.
5. Move the selected tasks from Backlog to the current Sprint.

**Note:** Keep the FIRST and SMART principles in mind—it is fine to divide stories and tasks so that they are achievable and measurable. MVPs should be prioritized. We only have 4 iterations, so pick what's really important! It is good to have a few extra MVPs/non-MVPs assigned to the sprint that can be ready to be assigned and moved on as initial tasks are completed.

## 2 Development Requirements

1. All stories selected for a sprint should be assigned to a main person and include unit/integration tests.
2. Follow the **Red-Green-Refactor** process. You don't need to worry about refactoring yet (end of semester).
3. You are responsible for **full stack development**. In other words, you are responsible for building both the front end and the back end of your SaaS application and performing the associated integration and unit level testing.
4. Communication and the use of branches will be key. Push, pull, and merge to main frequently. Protections have been placed on the main branch, so approval must be given for your merge to main to go through. **Work early** so that your branch is submitted in time for approval.
5. As you finish tasks, move them along to completion in your project management tool.

# 3 Demo Expectations

For each sprint demo, you must:

1. Share the URL to your production deployment so we can follow along
2. Tell us about your application and why you're making it
3. Go through GitHub Projects and show us what work you completed based on user story
4. Demo the work you completed on the production environment showing the feature in production and showing us the test cases used to test each requirement
5. Get customer feedback

# 4 Submission Requirements

### On Canvas (1 person)

- Submit your GitHub repository address
- Submit your application's production URL
- Submit a short report of what was implemented and tested. You may have not completed everything—include that in your report discussing what you will do.
- If your plan changed since your last submission, explain why
- If you are having issues with some of your tests, document why
- Record your velocity (from GitHub Projects) for later use

### On GitHub

- Make sure that all project files including your tests and code are committed and pushed to GitHub on the main branch
- **All** students should be pushing to GitHub. Make sure your name is associated with your GitHub pushes (learn more) so that you are not penalized for not contributing
- **Tag your git repository revision** with "sprint$i$" where $i$ is the sprint number
  - To tag the current revision: `git tag -a sprint1`
  - **Warning:** You must explicitly push your tags with the command `git push --tags`. Before you submit, please make sure that we can checkout the tag—you can try this from the command line and also check the GitHub website to double check that the tag is there.

### On GitHub Projects

Ensure project management tools are updated, cards are moved to done, and acceptance criteria has been developed for each card as gherkin scenarios. Be prepared to demo these scenarios during class.

### On Slack

One team member must post to your Dev_Cust Group on Slack with information about what was implemented and your application's URL. Customers will see and can write posts to/from their developers.

### AI Documentation

You must document in your project README where and when you got help from AI and include the transcript URL for each instance of support.

### Peer Evaluations

Complete your peer evaluations! The activity opens immediately after the sprint deadline and has multiple stages to complete.