**Grading Rubric:**

**\*Bolded text inside the rubric is used to show any points off or other comments.\*** For example, if “**\*2 points: description of expectation\***” is bolded, that means 2 points were deducted for that being not present.

1. Start by going through any uncompleted tasks from the prior deliverable, and assign them with new due dates.
2. Then go through the feedback on the prior deliverables and create & assign tasks for any unaddressed comments/feedback, both comments on documents and also in the overall feedback on Brightspace.

* If that’s not available yet, create a task for that by copy-pasting this paragraph.

### **Product Backlog** ( /5)

* The comments from Deliverable 2 must be addressed.
* For any comment that was not addressed, deduct -2 points.
* All unaddressed comments are carried forward by default to the next deliverable as well
* All columns based on the projects’ details document should be in the product backlog and must be filled out.
* For any missing column or missing information deduct -2 points.
* The backlog should have been updated with respect to sprint numbers, story points, etc.
* If there is no update, deduct -2.5 points.

### 

### **Sprint Backlog 3** (/5)

* The comments from Deliverable 2 must be addressed.
* For any comment that was not addressed, deduct -2 points.
* Should reflect the product backlog.
* That is, all the user stories that are assigned to Sprint 2 in the product backlog, should be shown in the Sprint Backlog 2.
* For any missing user story, deduct -2 points.
* If the Sprint Backlog 2 does not reflect the product backlog, deduct -5 points.
* You must have other technical tasks in your backlog otherwise -5.
* You must have other tasks in the backlog, like documentation, requirements, UI mockups, etc.
* As covered in class sessions, these are called non-technical stories. Otherwise -5.
* Sprint backlog must have accurate information about who did what % of the work.
* The sprint retrospective is a good time to fill that in, so you can see the difference between planned and actual effort.

### **Github Contributions** (/5)

* Contributions should be consistent and all students contribute.
* If a student does not contribute to Github, the student will be graded as 0.
* You can each commit different files, or make changes to the README.

### 

### **Kanban** (/5)

* Kanban board should show progress (on your issue/task management platform e.g. Github Issues + Github Projects, Jetbrains Space, etc)
* If no progress is shown, the grade for this part will be 0.
* If no link to the project management platform showing the Kanban is submitted, 0 and it’s not resubmittable for regrade.
* There should be 20-40 tasks for current Deliverable and prior deliverable revisions, each task should be small.
* If < 20 tasks, -2.5 points
* If any tasks are nonsensical or made just to get to 20, -10 points
* If there are more than 1-2 tasks with the entire team assigned to them, -2.5 points for each
* Each task should have the grading feedback/rubric information inside the task, for that task.
* The task assignee should have everything they need to know to do the task well, by just looking at the task. (Also for the quality checking for the task).
* -2 for each task missing this

### 

### **Sprint Review 3** (/10)

* Should follow the [template](https://docs.google.com/document/d/1QkzW3zqLNUdav9vv93DhmefqulpcLdep/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true).
* For any missing item from the template, deduct 2 points.
* Should be written for this past sprint / week (i.e. for Deliverable 1, Deliverable 1 and Revisions for Deliverable 0) but reflect about the whole project experience so far
* Use the template as an agenda for the sprint review meeting. Step through each point as a team, have a discussion and have someone take notes. The team should review the notes during the meeting in a shared Google Doc to make sure everyone is heard, and anyone can edit the notes.
* For any missing item from the template, deduct 2 points.
* Document should contain all prior Sprint Reviews as well, with the most recent at the top, otherwise -5.

### 

### **Architecture Design** (/5)

* Developers lead this
* The comments from Deliverable 2 must be addressed.
* For any comment that was not addressed, deduct -2 points.

### **SRS** ( /5)

* Need to ensure the comments from Deliverable 2 have been addressed.
* For any comment that is not addressed (every single one), deduct -2 points.
* Fill in the rest of the sections, using information from the project description document.
* See [this example](https://docs.google.com/document/d/1WkqFkOZBiUtC19x8w7Zt7qIg8MJishZp/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true) for an example of the minimum level of detail required.
* Unlike this example, your section 4 should be organized into subsections, grouping by general feature area or top-level use cases (i.e. don’t have a sub section for every single use case that is an extension, or instance of)
* If the number of FR and NFR were lower than 15 and 10 in Deliverable 2, you must meet those minimums in this deliverable. If not met, give 0.
* For each wrong requirement deduct -2 points.
* For each conflicting/contradictory requirement, deduct -2 points.
* If the UI is missing, give 0.
* If the UI is not updated, deduct -3 points.

### 

### **Use Case Diagrams/Models and Descriptions** ( /10)

* The comments from Deliverable 2 must be addressed.
* For any comment that was not addressed, deduct -2 points.
* Update Use Case Models and Descriptions document with potential new use cases and based on the feedback given on prior Deliverable(s)
* If the number of UCD were below 10 or the models/diagrams were below 2 in Deliverable 2 and they did not change in Deliverable 3, deduct -5 points for each.
* If the models/diagrams are missing entirely, an additional -5 points
* Use this [use case description](https://docs.google.com/document/d/1jBaLHNLAAEPAqN7BCqZspq04X625YcIY/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true) template. Here is an [example of Use Case Diagrams and Descriptions](https://drive.google.com/file/d/1_r0T_X7xg1zB9Vnt0CAREqIjXAOmi0MV/view?usp=sharing). Here is [another example](https://docs.google.com/document/u/0/d/1HsytILdXk_8f-2vJjSbXwhj9OYWIcvVP/edit?fromCopy=true).
* Make the use case diagrams first, then assign individual people to making each use case’s description.
* About 10 - 20 use case descriptions and 2-4 use case diagrams/models each including 3 - 5 use cases.
* If the number of use case descriptions (UCD) is below 10, deduct -3 points for any one missing. For example, if there are only 8 UCD, then deduct -6 points.
* If the number of use case diagrams/models is less than 2, deduct -10 points.
* Use case diagrams/models should be correct and have the correct links.
* If they have unnecessary arrows, deduct -2 points per each mistake.
* If they are complicated, deduct -5 points per each diagram/model.
* Use cases should start from the actor. If not, deduct -2 points for each mistake.
* Only have include, extend and generalize links between the use cases.
* If the links are not correct or they do not have a type, deduct -2 points for each mistake.
* Use case descriptions should follow the templates/guidelines from slides or the given template.
* If they do not match, deduct -5 points for each mistake.
* The name of the UCD should match with the use cases in the diagram.
* If they do not match, deduct -1 point per each mistake.
* The related use cases should be shown correctly in the UCD
* If they do not show correctly, deduct -1 point per each mistake.
* For every other mistake, deduct -1 points for each.
* If the quality of the UCD or diagrams varies substantially, deduct -10 for the team not internally reviewing/quality checking their work before submission. Not recoverable with regrade.

### 

### **Sequence Diagrams** (/35)

* Pick a few of the main features of your application and draw the sequence diagrams based on those. Prioritize drawing ones with the highest priority.
* For each of the sequence diagrams, start by writing a table describing the steps/messages, with columns for subject, subject action (a verb), parameters, and object acted upon.
* See class/slides for more information and examples.
* About 7 - 15 analysis sequence diagrams plus 3 -7 design sequence diagrams.
* If the number of analysis/design sequence diagrams is below 7 and/or 3 respectively, deduct -3 points per each. For example if there are only 5 analysis SD, deduct -6 points.
* See [this example of sequence diagrams](https://docs.google.com/document/d/16aZE4B08wakokbiZmZOn2_nkfWlb59rq/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true). Analysis sequence diagrams can be less formal (not showing method calls and formal types for arguments). Design sequence diagrams show message arguments and types (e.g. function call arguments and types), as covered in the class lecture.
* Each diagram needs to have correct variables, methods and parameters.
* For any mistake, deduct -2 points.
* For each diagram you need to write a description of the steps.
* For each description that is missing, deduct -5 points.
* For each incomplete description, deduct -2.5 points.

### 

### **New Version of the Application** (/15)

* If there is no progress compared to Deliverable 2, give 0.
* Any comments from Deliverable 2 must be addressed.
* For any comment that was not addressed, deduct -2 points.
* The implementation should have been started.
* Aim to have developed at least 1 of the core features of your app.
* You should use test-driven development.
* +5 for your application if you used TDD. Next deliverable TDD will be required.
* At least 1 feature is started, and there are at least 3 automated tests written for the feature (some of which may be failing and thus commented out)
* If no feature implementation is started, or there are no automated tests, give 0.
* The developers should lead this and are expected to, as needed, to review/complete parts of the [Interactive React Textbook](https://greglnelson.github.io/react-hooks-typescript-tome/) earlier than other team members, do self-directed learning using Google, StackOverflow, other online resources, and ask people or other classmates questions.
* Developers are expected to help any other team members with technical questions.
* If there is no code/HTML/scripts, and no changes from last time, this part will be graded as 0.
* If there is no HTML/CSS UI (this may include React components or just plain static HTML/CSS ) that looks similar to at least one of your UI mockups (or you have no UI mockups in your SRS, even if there is some UI), deduct -10 points
* Your designers will provide wireframes/mockups.
* In general it is best practice to make a simple UI mockup/sketch before implementing any UI more complicated than a single button or HTML element

**Setting up the build environments and the necessary dependencies.**

* Your README file should detail the installation and build process for your application.
* Make sure it works on each person’s machine.
* If there is nothing, deduct -5 points.
* You must upload a video to your team Discord channel in the COS420 Discord, from a synchronous team meeting/screenshare on Discord or Zoom, where each team member shares their screen showing the app working on their machine, to show that this has been set up and works for each team member.
* No video editing, one person on the team should record their screen continuously through the meeting (you can use the Zoom record meeting feature to do this easily).
* If this is not present, -10.
* For each team member not showing it works on their machine, -2.5